

---

# **Environmental Monitoring and Sensor Storage (EnMaSSe) Documentation**

***Release 1.0***

**Nigel Sim, Casey Bajema**

July 28, 2016



<b>1</b>	<b>EnMaSSe User Guide</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Metadata Records (ReDBox) Concepts . . . . .	6
1.3	Data Concepts . . . . .	7
1.4	Project Creation & Configuration . . . . .	9
1.5	Managing & Searching Data . . . . .	27
1.6	Project Lifecycle . . . . .	29
1.7	Permissions & Sharing . . . . .	31
1.8	Dataset Event Logs . . . . .	31
<b>2</b>	<b>EnMaSSe Administrator Guide</b>	<b>33</b>
2.1	Overview . . . . .	33
2.2	Installation & Configuration . . . . .	33
2.3	ReDBox Integration . . . . .	36
2.4	Project Management . . . . .	38
2.5	Editing Page Templates & Text . . . . .	40
2.6	User Roles & Permissions . . . . .	41
2.7	Managing Templates . . . . .	42
<b>3</b>	<b>EnMaSSe Developer Guide</b>	<b>45</b>
3.1	Ingestor API . . . . .	46
3.2	Developing the Provisioning Interface . . . . .	49
3.3	Developing the Ingestor Platform . . . . .	54
3.4	Project Reusability . . . . .	59
3.5	Information on Implementing Specific Functionality . . . . .	61
3.6	Limitations & Future Work . . . . .	62
<b>4</b>	<b>Nomenclature</b>	<b>65</b>



The EnMaSSe documentation is broken into three sections:

- User guide explains the application as a whole, application concepts and how to use it.
- Administrator guide focuses on installation, expected changes that administrators will need to do and general administration during normal usage.
- Developer guide outlines the frameworks and libraries used, the project structure and the technical details behind how it was developed.

*All three guides focus on different areas and it is recommended that they are read in the order explained above.*



---

# EnMaSSe User Guide

---

There have been great strides in making it easier to share and re-use research data throughout Australia, and EnMaSSe is a tool that takes it one step further. EnMaSSe is designed to increase the efficiency of creating and sharing research data, which means quicker progress within Australia as a whole, higher potential for new technologies and a better understanding of our environment.

Think of research data as information scientists & engineers have available for creating new technologies such as mapping how climate change will affect your life in the next 30 years, the latest medical cures or preserving our great Australian environment - almost anything you can think of, the building blocks of our entire modern lifestyle is research!

The EnMaSSe application provides a user-friendly interface for:

- Flexible and scalable research data ingestion (both streamed or manually input).
- High quality, fine grained, project based, metadata creation and export (eg. Enter 1 record, export many).
- Administerable and maintainable project lifecycle and workflows.

## 1.1 Overview

Researchers face many challenges once they decide to share data as there is a large amount of information that must always be attributed to this data in all circumstances of use. The most obvious of these is data collection methods, but there are other pieces of information that **NEED** to be passed on with this data, such as all parties involved in data collection (persons and organisations); who owns the data; who manages the data; version of the data; funding sources; licences; whether the data is derived; etc. EnMaSSe helps researchers to easily attribute this data about data (metadata) to whole projects, whole research methods, and all datasets associated with each project and method (Figure 1).

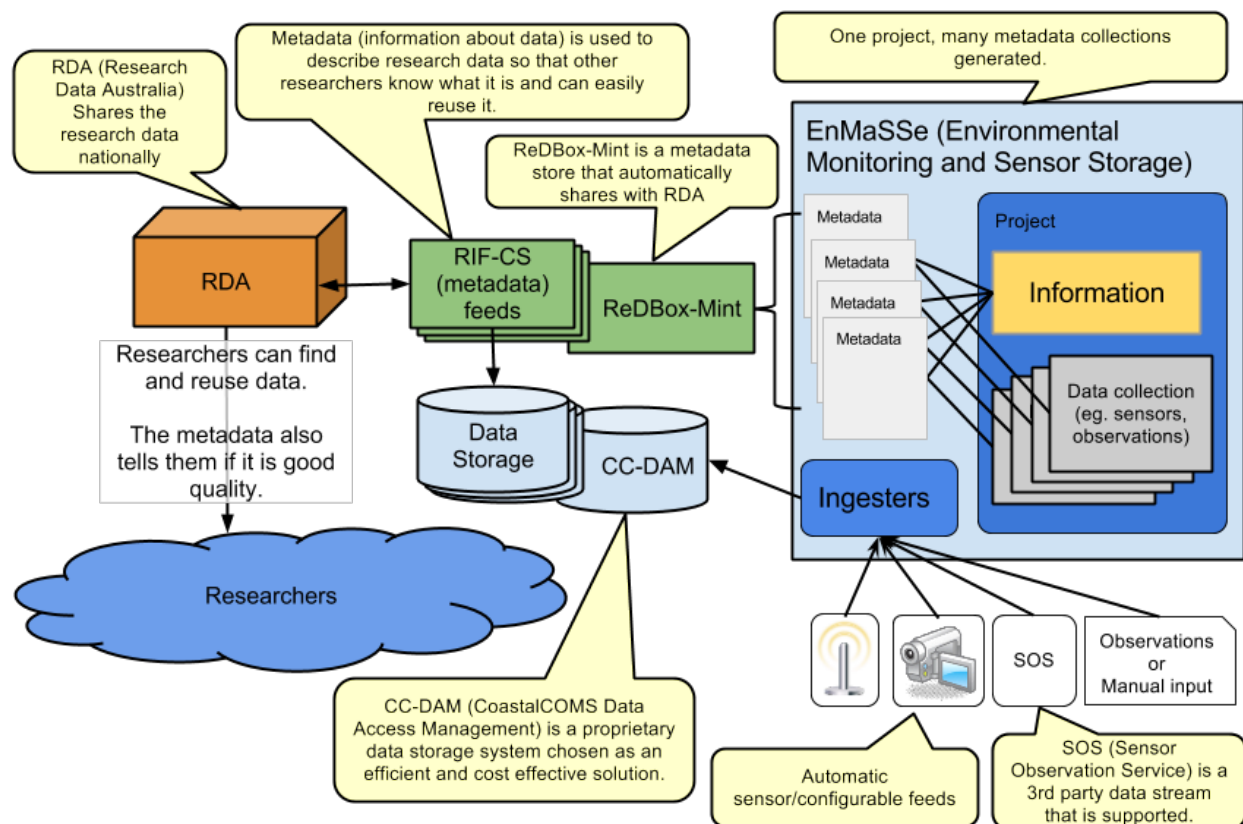


Figure 1: EnMaSSe Overview

The important outcomes of the EnMaSSe system are both:

- High quality information about the data (metadata) describing why it is important and good quality.
- Ingesting the actual data.

Figure 1 above illustrates how external sensors and manual observations are input into EnMaSSe along with the high quality information about the ingested data which is then exported to external systems for persistent storage and publication.

Research Data Australia (RDA) is a national service for publication and sharing of research data, this is the destination for the high quality information about the collected data.

ReDBox-Mint is 3rd party metadata repository (ReDBox) and name authority (Mint) which stores people, grants/activities and services. ReDBox handles curation (quality assurance and linking related information) and exporting to RDA.

CoastalCOMS Digital Asset Manager CC-DAM is the persistent data storage system used, to put it simply it is a database indexed file storage system which makes it highly scalable (for example our first import contained over 8 million files!).



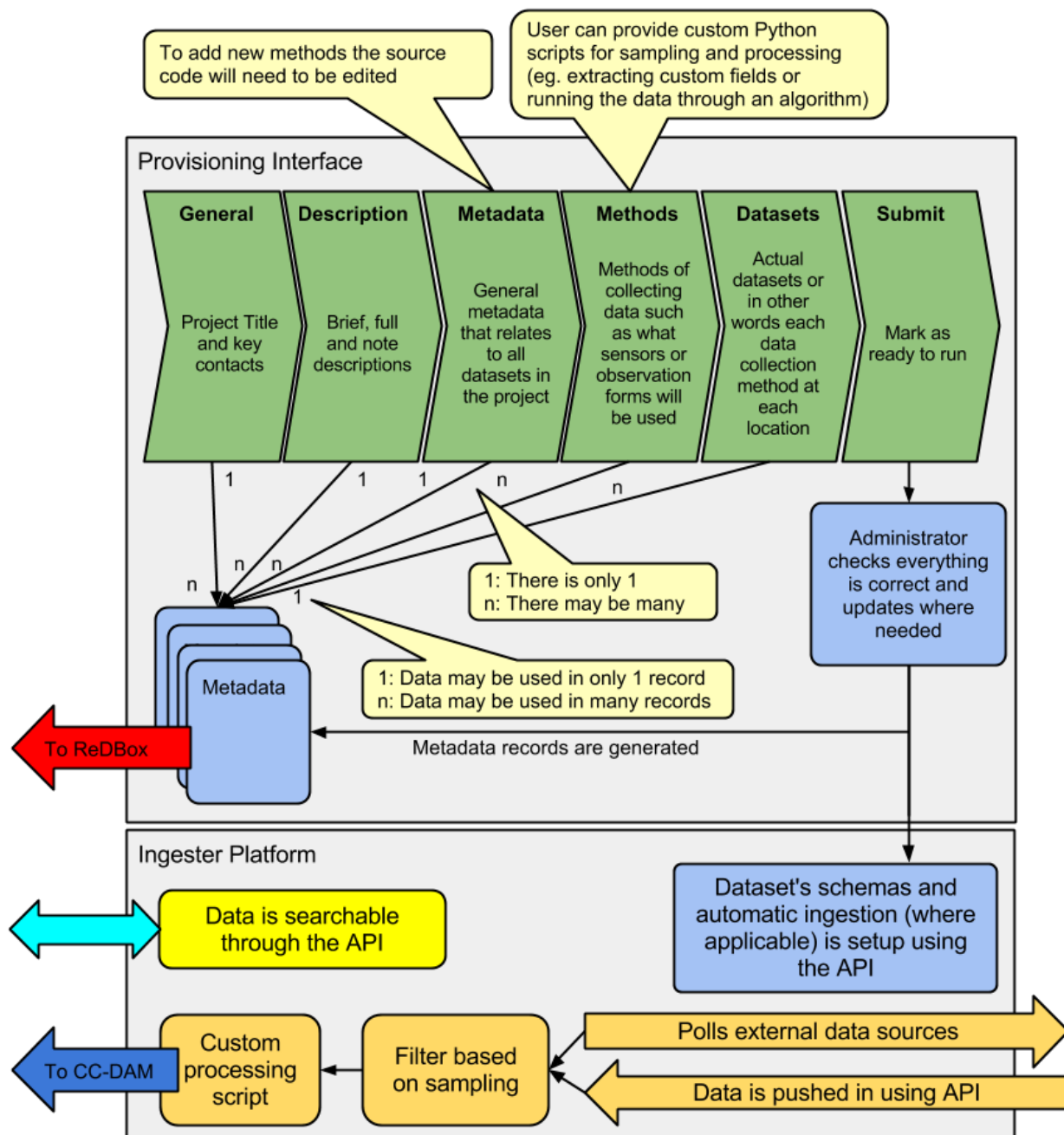


Figure 2: EnMaSSe workflows & components

Sensor data presents the most obvious example of data that is challenging for researchers to share: many datasets are collected at many locations sometimes with multiple sensor types often across multiple projects. Yet almost all metadata for a single sensor for a single project will be the same, and often the metadata between projects will be similar. EnMaSSe recognises this and is structured such that a single unique piece of information should rarely need to be typed more than once (Figure 2).

## 1.2 Metadata Records (ReDBox) Concepts

Metadata describes what your data is, who collected, why it was collected and how it can be used by others. Its purpose is to help those who are unfamiliar with the project understand what they are looking at. Even if you are not intending to share your data outside your project group, a metadata record will be useful to your future self - when you need to revisit the research after a long break.

The EnMaSSe application gathers some basic information from you and will automatically create a metadata record for each of your datasets from this.

### 1.2.1 Templates, grants and pre-fill

A core focus of the EnMaSSe system has been to reduce the amount of information that needs to be entered and pre fill as much as possible.

To achieve this goal a number of features have been implemented:

- **Project templates** allow administrators to provide projects that have any/all fields already filled, while this is a great feature it has the limitation that most projects will be dissimilar.
- **Method templates** allow administrators to provide method+dataset combinations for known data collection methods. This will be great for sensors that are commonly used and it is anticipated that this will both save a lot of time and simplify the use of sensors.
- **Standardised fields** allow administrators to provide common data configurations that the end user can select rather than recreating the way the same data is stored all the time.
- **Project creation wizard** allows selection of the project template as well as collecting the primary contacts and (optionally) the associated grant, data is retrieved and used to pre fill fields such as the brief description and project dates.
- **Record citation** are automatically generated from the entered information (administrators have the option to provide a custom citation).
- **Duplicate Project** allows any user to create a new project which is a copy of the currently viewed project. This is great for similar projects and is more flexible than templates that administrators first need to set up.

### 1.2.2 How metadata records are generated

When an EnMaSSe project is submitted and approved a metadata record is created for the project itself as well as each individual dataset, each dataset metadata record starts as a copy of the project record then the:

- **Title** is set to <project title> at <location name>(<lat, long, height>) collected by <method name>.
- **Location** is set to the dataset location (each dataset requires a valid location).
- Each dataset is associated with a method, the description entered for the method is added as a **note description**.
- **Publish date** is set as specified on datasets page.
- **Citation** is generated when sent to ReDBox.

If a dataset is required as an intermediary processing step and it doesn't make sense to create a metadata record, generation for the dataset can be disabled by unchecking the publish metadata record box.

All generated metadata Records link back to EnMaSSe for previewing of the ingested data, for project records this will redirect to the project pages (contextual sidebar options will allow browsing of associated data) and dataset records will redirect to the relevant manage dataset page.

### 1.2.3 Limitations

There are a couple of metadata record creation limitations with the current system.

**Records can't be edited after creation**, this is by design as published records should be persistent. One use case in support of this is where your data is cited in a publication, if the metadata record was then changed it could invalidate their work.

**Record generation isn't configurable**, this was out of scope - if specific minor changes are required the administrators may be able to update the record generation.

## 1.3 Data Concepts

When we talk about research data what we are really talking about is years of work, it is all too common for researchers to store data on their own computers or laptops and it could all be lost with a single hard drive failure!

But not only is data backups and redundancy important, it is also important to make sure the data is stored in a format that makes sense and can be found and used when needed.

This section introduces a number of complications that occur when working with data and provides more information on the concepts behind how to configure data ingestion.

### 1.3.1 Data Sources

Data sources are ways of getting data into the system and how that data should be processed to fit the data configuration.

EnMaSSe has been designed to be as flexible as possible by providing generic ways of ingesting data and allowing end users to provide custom python scripts to process the data,

Choose the data source that is easiest for you to use:

- **Web form/manual** data sources simply allow for manual data entry (web form) without any streamed data ingestion.
- **Pull from external file system** data sources ingest data from a folder on a web accessible server, this is a very generic and is anticipated to be the most commonly used data source.
- **SOS data sources** (Sensor Observation Service) provide data ingestion from an external SOS server, all data for every sensor is ingested and it is up to the processing script to retrieve and store the relevant data.
- **Push to this website** allows 3rd parties to develop software tools that integrate directly with the EnMaSSe system, allowing them to provide the data for ingestion however they like.
- **Output from other datasets** allows for chained processing and storage. Such as the data for many datasets is ingested as a single large file in a 'parent' dataset and each 'child' dataset then ingests and further processes the data individually.

Data sources aren't conceptually too difficult they just need to get the data from somewhere and process it to fit the provided data configuration.

The complicated part is that a custom Python script is needed to do the processing (Requires a developer/programmer), so most users will need to enter a description of their processing needs for the administrators to help with.

### 1.3.2 Data Configuration

Data configuration is about storing your data as efficiently as possible and identifying which fields potential users will want to search on.

The data configuration also sets up a manual data entry form where the custom fields and standardised fields are the fields on that form.

### 1.3.3 Efficiency & Searching

Think of each data ingestion as being a single file (eg. basic text file that you open in notepad), the custom fields and standardised fields you set up in data configuration are bits of data that you want read from the file and remembered so that they are easy to search.

Basically the less indexes used, the lower the processing and storage overheads, but the whole point is to make the data as reusable as possible so provide indexes on fields that are likely to be searched.

Storage and data configuration has been implemented this way to make EnMaSSe as efficient and scalable as possible, this is necessary as research data grows quite large (eg. our first import was over 8 million data points).

For example, when deciding which fields should be searchable - potential users would almost always search for the location and time (which is why they are compulsory) but they would probably also search on relevant data such as the temperature, it is unlikely that they would search on quality assurance or other minor/associated data though.

### 1.3.4 Process to work out how to store your data

While modelling data we want to **store it in a way that makes sense to the researcher**, so that it makes sense to the researcher when it comes time to reuse the data.

**It is good practice to store the original data** as well as the processed data or results, this is both in case the processing had errors and the original data is needed to recover as well as to allow other researchers to re-process the original data in the way they need.

The first step is to think about what data is being collected, how the data is originally stored and what needs to be searchable.

Now identify the different methods that are being used to collect the data, this may include:

- Different ways of collecting data, such as manual observation forms or sensors.
- Different methodology being used for the same data collection method, such as temperature sensors placed under trees vs temperature sensors placed in the sun for measuring the differences.

Each method just identified is a data collection method in the EnMaSSe system, now we need to work out the data configuration for each method.

Now we need to break the data your methods collect up to fit the EnMaSSe data configurations:

1. In most cases the raw data should be stored as a file of some kind so add a custom field of type file (it is good practice to permanently store the raw data for future needs). One possible exception is when the web form/manual data source is selected.
2. Identify what in your data needs to be searchable. This will typically be the final, processed result and is generally a common measurement (eg. temperature, weight, humidity) rather than associated information such as quality assurance.
3. Where available add fields that need to be searchable as standardised data fields section, if there is no applicable standardised field add them as custom fields.

After following this process you should now have your data logically modelled in a fine-grained manner that promotes efficiency of storage, efficiency of searching and the flexibility for researchers to reprocess and/or reuse your data in ways you don't even anticipate.

### 1.3.5 Limitations

There are some limitations with the current EnMaSSe implementation:

- Data is stored as flat files, so indexing (data configuration) needs to be done right from the start - it is possible to reprocess and re-index data, but this shouldn't be the norm.
- It is likely that users will require new standardised fields and template which requires constant administrator support.
- Standardised fields can only be added once, so if the user has two of the same type of data they will need to add the second as a custom field. We have since rethought this and it would make more sense for standard measurements such as temperature to be added as custom field types and standardised fields be used as a template like system for data configurations.

## 1.4 Project Creation & Configuration

EnMaSSe has been designed around projects that may contain many data collection methods and datasets (data collected by the same method at the same location/identifier). The project structure was chosen to require the minimum amount of data from the user to generate many metadata records and data ingesters (less work for you!).

### 1.4.1 1. Project Creation

The first step is to create a new project (click the New Project item in the main menu), this consists of a creation wizard that pre-fills fields based on the selected project template and the associated research grant as well as collecting the primary contacts.

Project templates allow for pre-filling of any/all fields, and provides the maximum time-savings when there are projects that are similar - equivalent functionality can be achieved using duplicate project in the sidebar.

☒ **Use a project template (only select if your project is similar to a previous one)\***

**? Select a Project Template**

1. First select the category or organisational group on the left hand side.
2. Then select the most relevant template from the list on the right hand side.

Austrian Wet Tropics  
**Blank (No auto-fill)**  
DRO  
TERN Supersite  
The Wallace Initiative  
Tropical Futures

**Blank Template**  
An empty template that allows you to start from scratch (only for advanced users or if no other template is relevant).

☒ **? There is an associated research grant\***

**? Research Grant**

Un-Select 'There is an associated research grant' above if your project isn't associated with a research grant.

Example artificial tree research grant

**? Data Manager (Primary contact)\***

Person A

**? Project Lead (Supervisor)\***

Person B

Create project

Figure 3: New Project page (Templates are hidden by default)

## Description of Fields

**Use a project template (only select if your project is similar to a previous one)** Select this checkbox if you would like to use a project template.

**Note:**

- Project templates are most useful when set up specifically for your department or research group, you can request the administrators to add new project templates.
- In most cases your supervisor or department will let you know in advance if there is an appropriate template for you to use.

**Select a Project Template** Categories are displayed on the left and templates are on the right.

First select your category/research group/department on the left side, then select the most appropriate template on the right.

**Note:**

- A template has been selected when it is outlined.

**There is an associated research grant** Un-select this checkbox if your project doesn't have a research grant.

**Research Grant** Start typing the title of your research grant, there is a short delay when you stop typing before the list of available grants is shown.

**Note:**

- If your grant isn't displayed in the list then please contact the administrators about getting it added to ReDBox-Mint.
- You must select an item from the autocomplete list, directly entered text will not work.

**Data Manager (Primary contact)\*** Start typing the name of the projects data manager, there is a short delay when you stop typing before the list of available people are shown.

**Note:**

- If your primary contact isn't displayed in the list then please ask them to login to the EnMaSSe system.
- You must select an item from the autocomplete list, directly entered text will not work.

**Project Lead (Supervisor)\*** Start typing the name of the project lead, there is a short delay when you stop typing before the list of available people are shown.

**Note:**

- If your project lead isn't displayed in the list then please ask them to login to the EnMaSSe system.
- You must select an item from the autocomplete list, directly entered text will not work.

## 1.4.2 2. General Details

After project creation the general details page is displayed and collects metadata including the title, associated grant and information about all associated people, groups and organisations.

If a research grant was provided in the project creation step:

- Project title is pre-filled with the grant title as a starting point
- Any additional people associated with the research grant are added to the people section.

**? Project Title\***

Micro-environmental change from canopy to forest floor

**? Research Grant**

Example artificial tree research grant

**? People**

✗ **? This project is Person\***

Managed by  [More Info](#)

✗ **? This project is Person\***

Aggregated by  [More Info](#)

[Add Person](#)

**? Collaborators (Organisations, groups or external people)**

Other people, groups or organisations who are associated with this project but cannot be added as a person above.

[Add Collaborator](#)

[Save](#) [Next](#)

Figure 4: General details page

## Description of Fields

**Project Title\*** Enter a descriptive name for this project, someone that sees the project title should get a general understanding of what the project is about.

**Note:**

- Metadata records generated for datasets will use the project title in the following pattern: <project title> at <location name> (<lat>, <long>, <elevation>m above MSL) collected by <method name>
- The project title may be pre-filled with the title of the research grant chosen on the project creation page.

**Research Grant** Start typing the title of your research grant, there is a short delay when you stop typing before the list of available grants is shown.

**Note:**

- If your grant isn't displayed in the list then please contact the administrators about getting it added to ReDBox-Mint.
- You must select an item from the autocomplete list, directly entered text will not work.
- This will be pre-filled if a research grant was provided on the project creation page.

**People** Add all people associated with this project.

**Note:**

- This will be pre-filled with the data manager and project lead.



- Due to internal restrictions the project lead is shown as aggregated by.
- If a research grant is selected all associated people will be pre-filled in this section.

**This project is** The relationship that this person has with the project, select the most relevant relationship.

**Person** Start typing the name of the person, there is a short delay when you stop typing before the list of available people are shown.

**Note:**

- If your person isn't displayed in the list then please ask them to login to the EnMaSSe system.
- You must select an item from the autocomplete list, directly entered text will not work.

**Collaborators (Organisations, groups or external people)** Add any additional collaborators that cannot be added in the people section such as people, groups or organisations.

### 1.4.3 3. Descriptions

The descriptions page provides plenty of space to enter the brief and full descriptions of the project as well as optional notes.

Detailed methods used within the project should not be entered in this section.

?
Brief Description\*

A short description targeted at a general audience.  
This field may be pre-filled with the grant description (as a starting point).

The dataset here represents the temperature and humidity changes that occur throughout a day across an 18m transect going from forest floor through the top of a semi-open canopy. Recording at 5 min intervals since December, 2012, this vertical transect further highlights seasonal changes in both canopy openness and sub-canopy vegetation dynamics that alter the temperature and humidity profile. This research is used to demonstrate the viability and robustness of low-cost sensors to provide invaluable information at

?
Full Description\*

Full description targeted at researchers and scientists

The dataset here represents the temperature and humidity changes that occur throughout a day across an 18m transect going from forest floor through the top of a semi-open canopy. Recording at 5 min intervals since December, 2012, this vertical transect further highlights seasonal changes in both canopy openness and sub-canopy vegetation dynamics that alter the temperature and humidity profile. This research is used to demonstrate the viability and robustness of low-cost sensors to provide invaluable information at spatial and temporal scales relevant to what species observe. This research is used to demonstrate the viability and robustness of low-cost sensors to provide invaluable information at spatial and temporal scales relevant to what species observe.

?
Note(s)

eg. TODO

Add Note

Save Next

Figure 5: Descriptions page

## Description of Fields

**Brief Description** Provide a short description of the research done, why the research was done and the collection and research methods used.

This description should be written in layman’s terms and focus on giving an overview of the whole project without going too far into detail about specific methods or datasets.

**Note:**

- The brief description may be pre-filled with the research grant description, this will need updating.
- The entered brief description will be used for all generate records, so make sure it makes sense for all methods and datasets that your project will use.

**Full Description** Provide a full description of the project targeted at researchers and scientists (technical details allowed!).

**Note:**

- The entered full description will be used for all generate records, so make sure it makes sense for all methods and datasets that your project will use.

**Note(s)** Optionally enter additional notes about the project, this may include things like additional information on funding bodies, high level overview of the project location or any information you want to add to the generated metadata records that doesn't really fit elsewhere.

**Note:**

- Notes will be used for all generate records, so make sure it makes sense for all methods and datasets that your project will use.

#### 1.4.4 4. Information

Collects the bulk of metadata (information about the collected research data) for the ReDBox record such as keywords, research codes, dates, location and other related information.

If a research grant was selected the date from and date to fields will be prefilled (when available).

**- Area of Research****? Keywords**

- ☒ Artificial Tree
- ☒ Temperature profile
- ☒ Humidity profile

[Add Keyword](#)**? Fields of Research\***


- ☒ 050206 Environmental Monitoring

**? Socio-Economic Objectives**


- ☒ 960806 Forest and Woodlands Flora, Fauna and Biodiversity

**Research Themes**

- ☒ Tropical Ecosystems, Conservation and Climate Change
- ☐ Industries and Economies in the Tropics
- ☐ Peoples and Societies in the Tropics
- ☐ Tropical Health, Medicine and Biosecurity

**? Type of Research Activity\***

- ☒ Applied research
- ☐ Experimental development
- ☐ Pure basic research
- ☐ Strategic basic research

**- Project Date and Location****? Time Period (description)**

**? Date data started/will start being collected\***

**? Date data stopped/will stop being collected**

**? Location\***

*Figure: Information page*

## Description of Fields

**Area of Research** Grouping of fields that categorise what type of project this is.

**Keywords** Provide a list of keywords for your project, keywords may be singular words or phrases.

**Fields of Research\*** Select the most appropriate Field of Research (FOR) which is selecting the categories for the methodology used by your project.

To select a Field of Research select the most relevant entry in each of the three dropdown boxes then click the Add Field of Research button on the right side of the last dropdown box.

**Note:**

- It is valid to select a Field of Research after only selecting values for the first two dropdown boxes.
- You may enter 1-3 Field of Research codes.
- Field of Research codes are standardised by the Australian and New Zealand Standard Research Classification (ANZSRC)

**Socio-Economic Objectives** Select the most appropriate Socio-Economic Objective (SEO) codes which is selecting an intended purpose or outcome of the research this project is recording.

To select a Socio-Economic Objective select the most relevant entry in each of the three dropdown boxes then click the Add Field of Research button on the right side of the last dropdown box.

**Note:**

- It is valid to select a Socio-Economic Objective after only selecting values for the first two dropdown boxes.
- You may enter 1-3 Socio-Economic Objective codes.
- Socio-Economic Objective codes are standardised by the Australian and New Zealand Standard Research Classification (ANZSRC)

**Research Themes** Select the most appropriate research theme.

**Note:**

- At least one research theme must be selected.

**Type of Research Activity** Select the most appropriate type of research activity for this project:

- **Pure basic research** is experimental and theoretical work undertaken to acquire new knowledge without looking for long term benefits other than the advancement of knowledge.
- **Strategic basic research** is experimental and theoretical work undertaken to acquire new knowledge directed into specified broad areas in the expectation of useful discoveries. It provides the broad base of knowledge necessary for the solution of recognised practical problems.
- **Applied research** is original work undertaken primarily to acquire new knowledge with a specific application in view. It is undertaken either to determine possible uses for the findings of basic research or to determine new ways of achieving some specific and predetermined objectives.
- **Experimental development** is systematic work, using existing knowledge gained from research or practical experience, that is directed to producing new materials, products or devices, to installing new processes, systems and services, or to improving substantially those already produced or installed.

**Note:**

- 1297.0 Australian Standard Research Classification (ANZSRC) 2008.

**Project Date and Location** Grouping of date and location fields which is sometimes referred to as coverage.

**Time Period (description)** Provide a textual representation of the time period such as 'world war 2' or more information on the time within the dates provided.

Date data started/will start being collected\*

The date that data started being collected.

**Note:**

- This is the actual data date not the finding date, recording date or other date. For example, an old letter may be found in 2013 but it was actually written in 1900 - the date to use is 1900.








**Date data stopped/will stop being collected** The date that data will stop being collected.

**Note:**

- This is the actual data date not the finding date, recording date or other date. For example, an old letter may be found in 2013 but it was actually written in 1900 - the date to use is 1900.

**Location** Provide the locations of this project, many locations may be entered as points, lines or polygons.

Locations can be added, edited or deleted using the controls in the top right corner of the map:

-  Navigate or drag the map to the desired location.
-  Draw a polygon (shape with any number of sides) of any shape.
-  Draw a line which may have multiple line segments.
-  Draw a rectangle (click and drag rather than clicking on each point).
-  Draw a single point.
-  Move points, this may be actual points or vertices of polygons and lines.
-  Delete a location, this has the same effect as pressing the X.

**Name** The name of the entered location, most research projects will have a code or name for each location such as Australian Wet Tropics or CU42A.

**Location** Actual location formatted in the WTK standard.

**Note:**

- If you want to enter a location manually as text it may be easier to add the location using the map first and edit the text that is provided.

**Elevation** Optionally, enter the elevation as meters above mean sea level (MSL).

**Note:**

- The entered elevation won't be used in exported metadata records.

**Licenses & Access Rights** Contains fields associated with licensing, getting access to the data and how the data can be used.

**Access Rights** Select how interested 3rd parties can go about gaining access to the projects data.

**License** Select the most appropriate license from the list, if you require a different license please let the administrators know so they can add it for you when approving the project.

**Retention period** Record the period of time that the data must be kept in line with institutional or funding body policies.

**Related Publications** Provide details on any publications that are related to this project including their title and URL with an optional note.

**Related Websites** Provide details on any websites that are related to this project including their title and URL with an optional note.

**Attachments (Uploading to ReDBox isn't supported at this time)** Optionally provide additional information as attachments.

**Note:**

- This should be added to all generated records but at the time of writing it is a limitation of the EnMaSSe integration with ReDBox.

### 1.4.5 5. Methods

The methods page sets up ways of collecting data (data sources), what the data is or its type (data configuration) as well as collecting the methods name (used to generate record titles of associated datasets) and description of the detailed methodology (added as a note description to records).

Adding methods uses a simple wizard that allows selection of a method template. Method templates pre-fill any/all data in methods and their associated datasets.

The type of data being collected allows configuration of what data is collected and how that data is indexed:

- Most methods will store raw data as a file and index specific information so it is searchable.
- Standardised fields are provided for common data types (eg. temperature, humidity, etc).
- Using the standardised fields will make the indexed data searchable globally within the data storage.
- Data configuration allows full configuration of the data types as well as how to display the fields in a web form.

Selection of the data source specifies how data will be ingested but configuration of the data source is done in the datasets step.

**\* - Method****? Method Name\***

Give the data collection method a name, this will be used in the title of the generated dataset records.

**Description\***

Provide a description of this method, this should include what, why and how the data is being collected but **Don't enter where or when** as this information is relevant to the dataset, not the method.

The Artificial tree captures a vertical, eighteen meter, humidity and temperature transect. Artificial Tree Sensor Boards are located at one meter intervals along the tree, and contain the humidity and temperature sensors. These sensor boards are sampled every five minutes and, if possible, uploaded every hour. The data is stored on a server as a series of time stamped files, each of which contains an aggregation of the data between uploads. These files are broken down into two sections, one for humidity and one for temperature. Each section contains the data aggregated between uploads for the respective sensor type. This method downloads the time stamped files containing the aggregated data from which the individual data points can be extracted and calibrated.

**? Data Source (How the data gets transferred into this system)\***

Additional configurations may be required on the datasets page (eg. each dataset for a pull from external file system method will need its location set on a per dataset basis).

- ☐ Web form/manual only
- ☒ Pull from external file system
- ☐ Sensor Observation Service
- ☐ (Advanced) Push to this website through the API
- ☐ (Advanced) Output from other dataset

**? - Data Configuration**

Configure how collected data should be stored, displayed and searched.

**Data Configuration Preview**

**? Standardised data fields (Recommended where possible)**

Please request additional standardised data fields through the contact form.

[Add Standard Data Field](#)
**? Custom Fields****\* Custom Field  
Field Type\***

**Name\***

**Description**


This preview shows how the web form will look for manual data entry.

**Data File (Text)\***

File that contains the aggregated recordings of all sensors on the tree.

No file chosen



Figure 6: Methods page

## Description of Fields

**Method Name** Provide a short, descriptive name for this method of collecting data.

**The entered name will be used in the generated dataset record as:** <project title> at <location name>(<lat, long, height>) collected by <method name>

The name and description will also be used to identify the method used in the datasets step.

**Description** Provide a description of this method, this should include what, why and how the data is being collected.

**Note:**

- The entered description will be added as note descriptions to the metadata records associated with this method.
- Don't enter where or when as this information is relevant to the dataset, not the method.

**Data Source (How the data gets transferred into this system)** Select the way you would like to ingest data for your project.

'Web form/manual' is the default (other data sources also allow adding data through a web form), 'Output from other dataset' provides advanced processing features and the other three methods allow automatic ingestion from compatible sensors or services:

- **Web form/manual only:** Only use an online form accessible through this interface to manually upload data (No configuration required).
- **Pull from external file system:** Setup automatic polling of an external file system from a URL location, when new files of the correct type and naming convention are found they are ingested (Configuration required on datasets page).
- **(Advanced) Push to this website through the API:** Use the XMLRPC API to directly push data into persistent storage, on project acceptance you will be emailed your API key and instructions (No configuration required).
- **Sensor Observation Service:** Set-up a sensor that implements the Sensor Observation Service (SOS) to push data into this systems SOS server (Configuration required on datasets page).
- **(Advanced) Output from other dataset:** Output from other dataset: This allows for advanced/chained processing of data, where the results of another dataset can be further processed and stored as required (Configuration required on datasets page).

**Note:**

- It will be possible to change the data source once the project has been submitted and approved.
- Refer to the Data Concepts section for a more indepth explanation of data sources.

**Data Configuration** Data configurations setup how ingested data will be stored and what data will be searchable.

**Note:**

- Refer to the Data Concepts section for a more indepth explanation of data configuration.
- Data configuration cannot change once the project is submitted and approved.

**Standardised data fields (Recommended where possible)** Standardised fields allow you to extend commonly used data configurations, this makes it both easier for you and collects more uniform data (which makes it easier to search).

Select the type of data you want to use and click the Add Standard Data Field button.

**Note:**

- It is a current limitation that you can only each type of standardised field once.

**Custom Fields** Each custom field adds an indexed (searchable) field to your data configuration.

Add additional custom fields by clicking the Add Custom Field at the bottom of the list of custom fields.

**Note:**

- It is highly recommended that you refer to the Data Concepts section.

**Name** Provide a name for your field.

**Description** Describe to other users what the purpose of this field is.

**Field Type** Select what type of data this field represents.

**Units (Integer, Decimal)** Enter the units for this field, this will sometimes be not applicable and you can leave it blank.

**Mime Type (File)** Provide a mime type for your file (eg. text/json)

**Example (Single line text, Multi line text)** Provide an example of the sort of text expected.

**Default Value (All)** Enter a default value, this will be the value used if no value is given.

**List of Values (Dropdown box, multiple choice)** Provide a comma-separated list of options (eg. Red, Blue, Green)

**Admin Notes (All)** If you need help from the administrators provide a description of your requirements for this field.

**Attachment (Such as datasheets, collection processes, observation forms)** Attach files that provide more information on your data collection method. For example, this may include data sheets for sensors used or in-depth detail on the methodology or calibration methods used.

To add an attachment:

- Click on the browse button.
- Find the file on your local computer.
- Click the open button.

**Further information website (Such as manufacturers website or supporting web resources)** Provide information on any websites that describe your data collection method, this is similar to attachments but provide website links instead of the file itself.

**Title** Provide a name for the linked website.

**URL** Enter the website address.

**Notes** Optionally add a note about why the website was linked to.

## 1.4.6 6. Datasets

Each dataset represents an individual collection of data with an associated metadata record (metadata record generation can be disabled).

Adding datasets uses a simple wizard where the data collection method is selected as shown in figure 7 below.

The dataset page collects the following data:

- Whether to create a metadata record and when the record should be published.

- Location of the data, the location may be a set location or an offset from a location where that is more relevant. For example it is more relevant that the sensor shown is 1m from the base of the artificial tree.
- Configuration of the data source.

Each data source is configured differently but will usually require the data location, when to sample and how to process the found data.

## Datasets

## ✗ - Dataset for Artificial Sensor Tree

☒ ? Publish Metadata Record (Publicly advertise that this data exists)

? Date to publish\*

? Location\*



✗ ? Name\* ? Location\* ? Elevation

? Location Offset (optional)

Latitude Offset/X (meters) Longitude Offset/Y (meters) Elevation Offset/Z (meters)

## - Pull Data Source

? Address (URL/URI)

? File Field

*This will be empty if the methods, data configuration doesn't have a custom field of type file.*

? (Advanced) Filename Pattern (Regex)

*Unless you know how to use this or that you need this, just leave it blank.*

? Periodic Sampling (How often should new files be looked for)

? - Custom Data Processing (Read data from the found file)

*If you haven't used this system before you will need help to create a processing script, describe your requirements as best you can below and an administrator will contact you.*

Describe custom processing requirements (or describe your script)

Figure X: Datasets page.

## Description of Fields

**Publish Metadata Record (Publicly advertise that this data exists)** Un-select this checkbox if the dataset shouldn't export a metadata record. This option has been provided tentatively as there are some valid reasons to not create records such as:

- Testing or administration purposes.
- Datasets that use a dataset data source for intermediary processing and the created data isn't a usable end result.

**Note:**

- We would like to encourage as many datasets be published as possible (this is the purpose of the EnMaSSe system!).



**Date to publish** When should this metadata record be published? Update the publish date if it shouldn't be published right away.

**Note:**

- The publish date will be pre-filled to today's date.
- If the publish metadata record is un-selected this field will be hidden.

**Location** Provide the location of this dataset, only one point location may be entered.

Controls for adding points or navigating the map are located at the top right corner of the map:

-  Navigate or drag the map to the desired location.
-  Draw a single point.

**Name** The name of the entered location, most research projects will have a code or name for each location such as Australian Wet Tropics or CU42A.

**Location** Actual location formatted in the WTK standard.

**Note:**

- If you want to enter a location manually as text it may be easier to add the location using the map first and edit the text that is provided.

**Elevation** Optionally, enter the elevation as meters above mean sea level (MSL).

**Note:**

- Dataset location will be pre-filled to the project location if the project had a valid point location before the dataset is created.
- The entered elevation won't be used in exported metadata records but it will be used in the record title.

**Location Offset (optional)** Providing a location offset means that the actual location used will be offset from the location entered above.

This may seem odd but it is useful where the important information is the distance from somewhere rather than the actual point on earth.

For example, you may have many sensors spaced around a central point - it would then make sense to enter the project location at the centre and set each dataset to offset from that location.

**Latitude Offset (meters)** How far the location latitude should be offset in meters, this can be positive or negative.

**Longitude Offset (meters)** How far the location longitude should be offset in meters, this can be positive or negative.

**Elevation Offset (meters)** How far the location height above mean sea level should be offset in meters, this can be positive or negative.

### 1.4.7 7. Submit

Submit provides full project validation and an overview of the generated records and data ingesters. The project has four states:

- **Open** - The initial state when a project is created, the creator and administrators have read/write access. The creator can also share permissions with other users.
- **Submitted** - When the project is submitted by the creator it is ready to be reviewed by the administrators and either approved or reopened. A project can only be submitted when there are no validation errors. In the submitted state creators have read access and administrators have read/write access.
- **Approved - When an administrator approves the project:**
  - Metadata records are exported to ReDBox.
  - Data ingesters are configured and started.
  - The project can no longer be modified, the creator and administrators only have read access.
- **Disabled** - This state represents the end of the project, when an administrator disables an approved project it disables all ingesters (no more data will be ingested).

The generated record for each dataset can be viewed, edited or reset. Viewing a dataset record is exactly the same as general details, descriptions and information all on a single form.

**- Validation**

Validation successful.

**- Summary of Datasets & Records**

Dataset Name	Data URI	Record URI	Record
dataset for Artificial Sensor Tree method	Not exported yet	Not exported yet	<a href="#">Create Record</a>
dataset for Artificial Tree Sensor Board method	Not exported yet	Not exported yet	<a href="#">Create Record</a>

?
Project Notes

eg. Please enter all metadata, the supplied processing script has errors, please extend the existing temperature data type so that your data is searchable, etc...

Add Project Note

Save notes

Submit

Figure 7: Submit page.

## Description of Fields

**Validation** The validation section lists all errors that need to be fixed before the project can continue on to the next step.

The submit and approve buttons will be hidden if there are any validation errors.

Validation errors are categorised per page, giving the name of the field and the error message.

**Summary of Datasets & Records** Provides a quick overview of all datasets in the project along with:

- Link to their data management page.
- Link to exported metadata records (if the project has already been approved).
- Link to create and/or view the metadata that will be exported.
- Option to reset all modifications to that specific datasets metadata.

**Project Notes** Add notes to the project, these will be visible to everyone with view access and are a simple way of communicating between administrators and users.

## 1.5 Managing & Searching Data

Projects, datasets and data can be searched and edited through the Browse Data page (accessed through the Browse Data menu item along the top):

- You can specify your search criteria in the left hand sidebar, this includes the search type, ID list, string/keywords, state and dates. Each search criteria further filters the results.
- The buttons along the top of the page allow you to perform actions on multiple search results (eg. enable all selected datasets).
- The dropdown/multi-select boxes along the top-right allow you to order your search results.
- Each result can be selected by clicking the checkbox on it's left.
- Each search result has actions on the right hand side.

**Search**

With selection:

Disable

Enable

View Datasets

Showing 0 to 1 of 1

Order by

ID

▼

Descending

▼

Max Results

20

▼

?

Type

Projects

▼

?

ID List

eg. project\_1, project2

Search String

?

State

☐ Open
 ☐ Submitted
 ☐ Active
 ☐ Disabled

?

Start Date

?

End Date

Search

This page doesn't provide bulk download of ingested data - these features are provided by the [data portal](#).

<input type="checkbox"/>	ID	Type	State	Created	Modified	Description			
<input type="checkbox"/>	project_2	project	Open	2013-06-06	2013-06-06	Test project	<a href="#">Edit Project</a>	<a href="#">Add Dataset</a>	<a href="#">Datasets (click here for data)</a>

Figure 8: Screenshot of the browse data page which provides data management and searching features.



## 1.6 Project Lifecycle

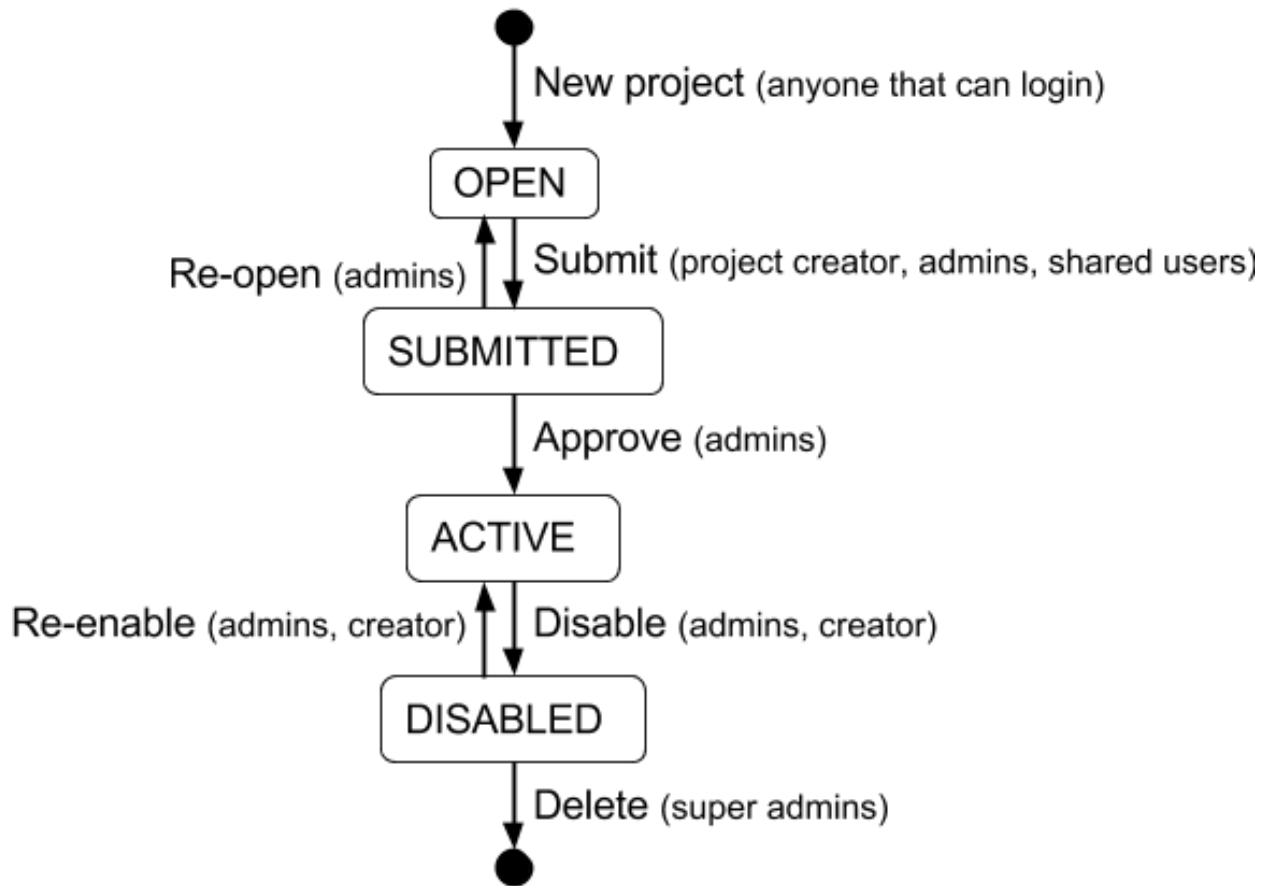


Figure 9: Project life cycle

Projects in the EnMaSSe system follow the life cycle illustrated in figure X above, generally you will only need to submit the project for administrator approval, but the whole process has been illustrated here for completeness.

### 1.6.1 Open

The open state is for initial setup of the project including all configurations ready to create metadata records and setup data ingestion.

Once the project has been fully configured the user should go to the submit page to and click the submit button to indicate that the project is ready for administrator approval.

#### Permitted

Creator, administrators and users that have been given share permissions can edit the full configuration of the project.

#### Not permitted

Nobody can view logs, enter data or access any data management as the project hasn't been activated yet.

### 1.6.2 Submitted

The submitted state is for administrators to check the project configurations are correct and the project is ready to be approved for metadata record generation and data ingestion set up.

Once the administrator has checked that the configurations are correct they should press the approve button on the submit page. Pressing the approve button starts the metadata record generation and export as well as setting up and starting the data ingesters.

If the administrator requires the creator to update the configurations they should press the reopen button on the submit page.

#### Permitted

Administrators can edit the full configuration of the project.

#### Not permitted

Nobody can view logs, enter data or access any data management as the project hasn't been activated yet.

Non-administrators cannot edit anything.

### 1.6.3 Active

An active project has had all metadata records generated and exported and the data ingesters are currently running.

Once a project has come to the end of its life either the project creator, administrator or user with adequate permission shared should press the disable button on the submit page. Pressing the disable button will deactivate data ingestion, all data and metadata records will be maintained.

#### Permitted

The project creator, administrators and users with shared permissions can access the data maintenance sections which allow adding/editing of data and limited updating of ingesters.

#### Not permitted

Project configurations can no longer be edited by anyone.

### 1.6.4 Disabled

The disabled state is where the project is finished but all data and metadata records are persisted.

If the project needs to be re-activated (there is more data to ingest) the re-enable button on the submit page should be pressed by the creator, administrator or other user with adequate shared permissions.

If this is a project that was set up wrong to begin with and has no valid data, the super administrator can delete it. Once the project is deleted it cannot be recovered, this functionality is only intended for administration and cleanup purposes and would never be used in an ideal world.

### Permitted

The project creator, administrators and users with shared permissions can access the data maintenance sections which allow adding/editing of data and limited updating of ingesters.

### Not permitted

Project configurations can no longer be edited by anyone.

## 1.7 Permissions & Sharing

When on a project page there is a sharing option in the left hand contextual options menu which allows the project creator (or administrators) to give other users permissions for on the currently viewed project:

- **View** allows the user to view project configurations.
- **Edit** allows the user to view and edit the project configurations.
- **Submit** allows the user to press the submit button requesting administrator approval.
- *Disable\** allows the user to press the disable button to deactivate data ingestion.
- **Re-enable** allows the user to press the re-enable button to reactivate data ingestion.
- **View Data** allows users to view ingested data.
- **Manage Data** allows the user to view and edit ingested data.
- **Manage Ingesters** allows the user to update ingester configurations (eg. how ingested data is processed).

**Note:** Users must be within the EnMaSSe system before permissions can be shared with them. If you cannot find the user you are looking for, ask them to login and then they should be available.

## 1.8 Dataset Event Logs

When on a project page there is a view logs option in the left hand contextual options menu which allows the user to see any error messages or state changes of the data ingestion.

By default all logs for all datasets will be displayed as they are loaded, you can provide filtering options or click on the link to see only logs for the selected dataset.

Log filtering options include:

- **Log level** which is the category or type of message such as error or information.
- **Start date** is the date of the earliest logs to show.
- **End date** is the date of the last logs to show.



---

## EnMaSSe Administrator Guide

---

### 2.1 Overview

*This guide explains how to setup and administer the Environmental Monitoring and sensor storage (EnMaSSe) application.*

*This guide doesn't cover how to use the system and it is recommended that administrators first read the user guides.*

There are a number of integrated systems to setup and configure:

- **Provisioning interface** - This is the interface that end users will interact with.
- **Ingestor platform** - Handles data streaming and data storage.
- **ReDBox** - Provides a host of metadata functionality including publishing to ANDS. ReDBox is an open source 3rd party software and only EnMaSSe specific configuration is included in this document.
- **Shibboleth** - Is a federated authentication system that wraps the application. Shibboleth is 3rd party software and only an overview of our experiences and recommendations are provided in this document.
- **CC-DAM** - Is the data storage implementation chosen for our implementation and deployment of EnMaSSe. CC-DAM is proprietary 3rd party software and out of the scope of this documentation, please contact Coastal-Coms for further information.

Log files can be found under:

- Provisioning interface -> <install location>/src/jcu.dc24.provisioning/provisioning.log
- Ingestor platform -> <install location>/src/jcu.dc24.ingesterplatform/platform.log

### 2.2 Installation & Configuration

The EnMaSSe system has source code in a number of repositories including:

- <https://github.com/jcu-ereseach/EnMaSSe-Deployment.git>
- <https://github.com/jcu-ereseach/TDH-rich-data-capture.git>
- <https://github.com/jcu-ereseach/TDH-dc24-ingester-platform.git>
- <https://github.com/jcu-ereseach/jcu.dc24.ingesterapi.git>
- <https://github.com/jcu-ereseach/python-simplesos.git>
- <https://github.com/davidjb/ColanderAlchemy.git> branch=davidjb

The good part is that deployment is provided using buildout which will pull the sources from all the associated repositories and set them up correctly.

Full deployment of EnMaSSe is documented and provided by the [jcu-ereseach/EnMaSSe](#) repository, a summary of the full instructions is:

1. Install Python 2.7 with the required packages and development libraries (see repo for full commands).
2. Checkout the EnMaSSe-Deployment repository.
3. Create a virtualenv and pip install `zc.buildout` distribute `uwsgi`.
4. Run buildout.
5. Set the provisioning interface and ingester platform up as services.

### 2.2.1 Interface/Application

Configuration of the provisioning interface is provided through the `production.ini` file located in the base provisioning interface directory (`<install location>/src/jcu.dc24.provisioning/production.ini`).

Application specific configuration values (**Compulsory items are bold**):

<b>sqlalchemy.url</b>	MySQL username, password, address and database.
<b>ingesterapi.url</b>	Address of the ingester platform, in most situations this shouldn't change unless you require custom ingester platform configuration.
<b>ingester-api.username</b>	Ingester platform username as configured.
<b>ingester-api.password</b>	Ingester platform password as configured.
<b>mint.location</b>	Location of the Mint name authority for looking up users.
<b>redbox.url</b>	Location of the ReDBox installation.
<b>red-box.search_url</b>	Location of ReDBox search url (prepended to redbox.url)
<b>red-box.alert_url</b>	Location of the ReDBox alert url (prepended to redbox.url). ReDBox will read and ingest records when this url is hit.
<b>red-box.ssh_host</b>	SSH Address for the ReDBox server.
<b>red-box.ssh_port</b>	SSH port to use for connecting to the ReDBox server.
<b>red-box.rsa_private_key</b>	Location of the SSH Private key for the ReDBox server.
<b>red-box.ssh_username</b>	SSH username to use for connecting to the ReDBox server.
<b>red-box.ssh_password</b>	Either the SSH password or the private key password (may be blank if using an SSH key that isn't password protected).
<b>red-box.ssh_harvest_dir</b>	Location to upload generated record XML files to.
<b>red-box.local_config_file</b>	Where the ReDBox field mappings file will be saved to. The mapping file is generated when the provisioning interface is started.
<b>red-box.identifier_pattern</b>	Prefix for all generated ReDBox identifiers.
<b>provisioning.for_codes</b>	CSV File that contains FOR codes.
<b>provisioning.seo_codes</b>	CSV File that contains SEO codes.
<b>workflows.files</b>	Location that file uploads are saved to, this isn't contained in temp as some file uploads need to be persistent (such as files attached to project or method templates).
<b>tmpdir</b>	Temporary directory
<b>pyramid_deform.tmpdir</b>	Temporary directory for pyramid/application specific files.
<b>mint.tmpdir</b>	Temporary directory for mint specific files.
<b>redbox.tmpdir</b>	Temporary directory for ReDBox specific files. This is used to write record XML to be transferred to ReDBox using SFTP.
<b>session.secret</b>	Secret/password that your session uses (random generated number).
<b>host</b>	(non-UWSGI server) IP Addresses that your server will accept.
<b>port</b>	(non-UWSGI server) Port that the provisioning interface can be accessed on.
<b>base_dir</b>	base installation directory of the provisioning interface
<b>socket</b>	(UWSGI server) IP addresses to accept and the port that the UWSGI container will be accessible on.

Other settings can be changed as per their documentation:

- General pyramid (the framework) configurations: <http://docs.pylonsproject.org/projects/pyramid/en/latest/narr/project.html#myproject.ini>
- SQLAlchemy (database): [http://docs.sqlalchemy.org/en/rel\\_0\\_8/core/engines.html](http://docs.sqlalchemy.org/en/rel_0_8/core/engines.html)

- pyramid\_beaker add-on settings: [http://docs.pylonsproject.org/projects/pyramid\\_beaker/en/latest/](http://docs.pylonsproject.org/projects/pyramid_beaker/en/latest/)
- Logging configuration: <http://docs.pylonsproject.org/projects/pyramid/en/latest/narr/logging.html>

## 2.2.2 Web Server & Shibboleth

**Shibboleth** is a federated authentication service that sits in front of the application and injects user authentication headers.

**Shibboleth** requires a web server that supports CGI, JCU eResearch has chosen the **NGINX** webserver using **Fast CGI** to support **Shibboleth**.

The application login address is located at /login/shibboleth, shibboleth should be configured to handle authentication at this address and redirect back to the application when complete.

The EnMaSSe provisioning interface is based on **Pyramid** using a **UWSGI** container for production deployment.

## 2.3 ReDBox Integration

**ReDBox** is a research metadata repository that among other things, publishes research data nationally to Research Data Australia (RDA).

EnMaSSe integrates with **ReDBox** by exporting project metadata and per dataset metadata when a project is submitted and approved. The export occurs by:

1. Converting the provisioning interfaces metadata tables into XML files.
2. Uploading the files to the ReDBox server over SFTP (File transfer over SSH).
3. Polling the ReDBox alert URL to start the ReDBox harvest.

Before EnMaSSe can integrate with **ReDBox** it requires a new alerts harvest configuration to be added within **ReDBox** and the SSH permissions and configurations to be set-up.

### 2.3.1 SSH Access

A new user should be created on the **ReDBox** server with a public key and write access to the enmasse-alerts folder (see alerts configuration). The actual steps to do this are operating system dependent but for a linux machine it would be something like:

1. `useradd <username>`
2. `passwd <username>`
3. Copy/paste your public key into `~/.ssh/authorized_keys`
4. `cd <enmasse-alerts location>`
5. `sudo chown <username>:<username> ./`

Once the new user has been created and has write permission the provisioning interface configuration file needs to be updated with the connection details:

1. Open `<install location>/src/jcu.dc24.provisioning/production.ini`.
2. Edit `redbox.ssh_host` to the server IP address or domain name (eg. `example.com.au`).
3. Set `redbox.ssh_port` as configured on the server (default is 22).



4. Set `redbox.rsa_private_key` to the fully qualified location of the private key that corresponds to the public key added above.
5. Set `redbox.ssh_username` as the user created above.
6. Set `redbox.ssh_password` as the SSH private key password (or leave blank).
7. Set `redbox.ssh_harvest_dir` to the location of the `enmasse-alerts` folder (eg. `/opt/deployment/redbox/home/harvest/enmasse-alerts`)
8. Update `redbox.identifier_pattern` to an appropriate identifier prefix for your organisation (eg. JCU uses `jcu.edu.au/collection/enmasse`)

## 2.3.2 Alerts Configuration

There are 2 steps to configuring the EnMaSSe alerts harvester:

1. Copy the `enmasse-alerts` folder from the root folder of the github repository and add it to `<redbox installation dir>/home/harvest/`
2. Update the `<redbox installation dir>/home/system-config.json` file as illustrated below:

```
{
  ...
  "houseKeeping": {
    "config": {
      "quartzConfig": "${fascinator.home}/quartz.properties",
      "desktop": true,
      "frequency": "3600",
      "jobs": [
        ...
        {
          "name": "newalerts-enmasse",
          "type": "external",
          "url": "http://localhost:${jetty.port}/redbox/default/hkjobs/newalerts",
          "timing": "0 0 1 * * ?"
        }
      ]
    }
  }
  ...
  "new-alerts": {
    "alertSet": [
      {
        "name": "EnMaSSe Provisioning",
        "path": "${fascinator.home}/harvest/enmasse-alerts",
        "harvestConfig": "${fascinator.home}/harvest/enmasse-alerts/config/enmasse-dataset.js",
        "handlers": {"xml": "XMLAlertHandler"},
        "baseline": {
          "workflow_source": "EMAS Alert"
          "viewId": "default",
          "packageType": "dataset",
          "redbox:formVersion": "1.5.2",
          "redbox:newForm": "true",
          "redbox:submissionProcess.redbox:submitted": "true",
          "xmlns:dc": "http://dublincore.org/documents/2008/01/14/dcmi-terms/",
          "xmlns:foaf": "http://xmlns.com/foaf/spec/",
          "xmlns:anzsrc": "http://purl.org/anzsrc/",
          "dc:type.rdf:PlainLiteral": "dataset",
          "dc:type.skos:prefLabel": "Dataset",

```

```

        "dc:language.dc:identifier": "http://id.loc.gov/vocabulary/iso639-2/eng",
        "dc:language.skos:prefLabel": "English"
    },
    "timestampFields": ["redbox:submissionProcess.dc:date"],
    "XMLAlertHandlerParams": {
        "configMap": {
            "xml": {"xmlMap": "${fascinator.home}/harvest/enmasse-alerts/config/"}
        }
    },
    ],
    "baseline": {
        "viewId": "default",
        "packageType": "dataset",
        "redbox:formVersion": "1.5.2",
        "redbox:newForm": "true",
        "redbox:submissionProcess.redbox:submitted": "true"
    }
    ...
}

```

## 2.4 Project Management

A project workflow has been setup to encourage ease of use while allowing administrators to maintain a high quality of data and metadata.

EnMaSSe has the following possible states:

State	Transition	Group/Permissions	State Description
Open	Submitted	CREATOR/SUBMIT	Project is open for all users with write permission to edit.
Submitted	Open	Approve	ADMIN/REOPEN, ADMIN/APPROVE   Users have finished setting up the project and submitted it for administrator approval.
Approved	Disabled	ADMIN/DISABLE	Administrators have approved the project, ReDBox records have been created and data ingesters have been set-up.
Disabled	Approved	Delete	ADMIN/ENABLE, SUPER_ADMIN/DELETE   Project has been disabled, stopping all ingesters from collecting more data.
Deleted			Project has been deleted, this action is restricted to super administrators.

The intended workflow is illustrated below:

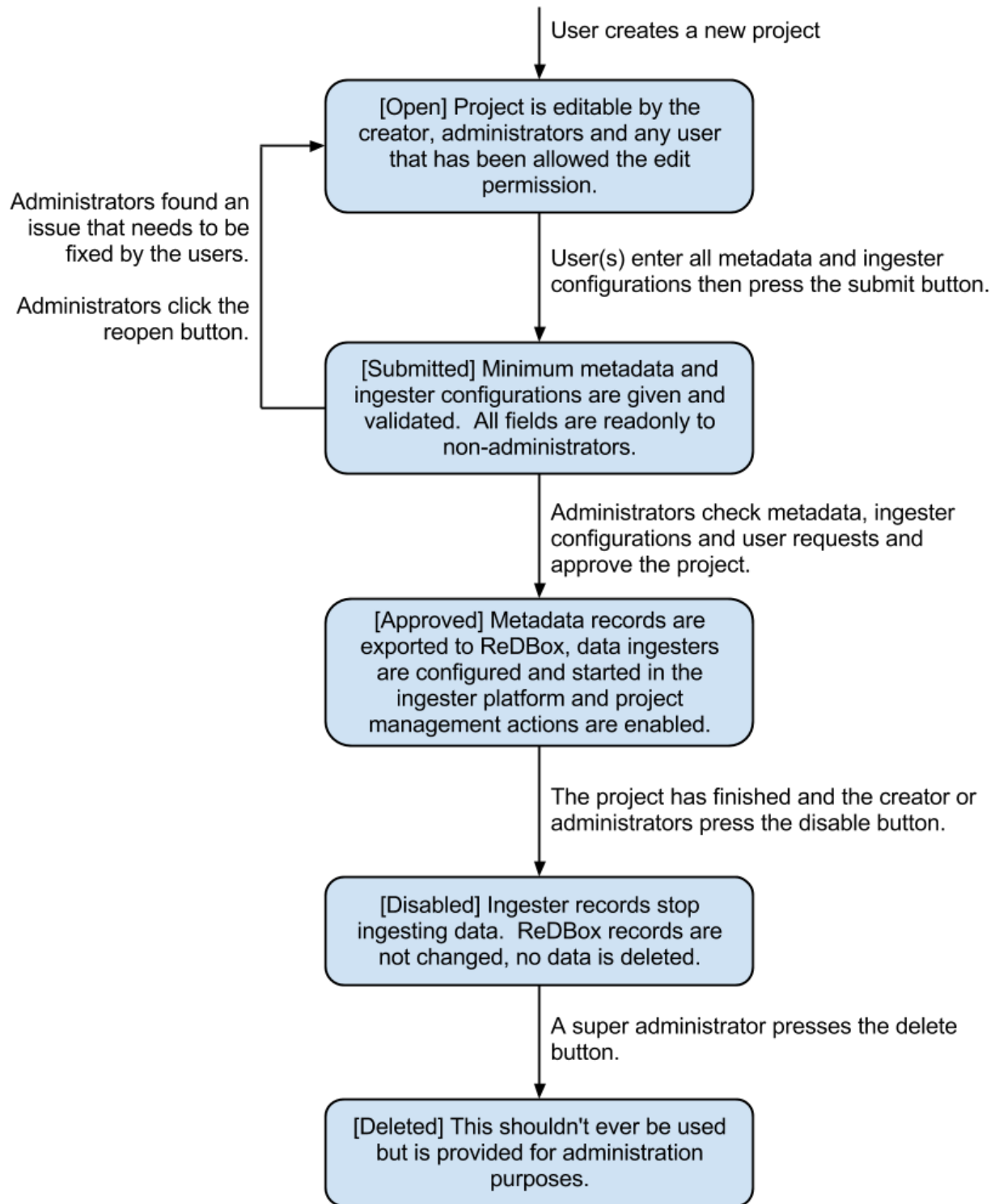


Figure 1: Project states and transitions.

The manage data and logs contextual options are only available in the approved state (please refer to the EnMaSSe user guide for more information).

Administrators (and any user with the `ADVANCED_FIELDS` permission) can view some additional fields that are usually hidden, at the time of writing these included usage rights, other licenses and access rights url (information

page).

## 2.5 Editing Page Templates & Text

All pages are displayed from a view using form models and templates:

- **Views** describe what is displayed where and when, they are the logic behind the website.
- **Models** are abstractions of both forms/form elements and database tables (the data).
- **Templates** are what is actually displayed on the screen, they are the human readable representation of data.
- **Stylesheet** is the CSS stylesheet that makes the templates look good (eg. colours).

For changing **text in a form** you should look at the **models** section. To change **static text** or how things are displayed, look in the **templates** section. For **menu text or page help**, look in the **views** section. For **styling** such as colours, positioning and backgrounds look at **stylesheets**.

Please note that these files are written in python, HTML and CSS programming languages, you should be able to edit plain text for minor changes but may need developer support for anything more advanced.

### 2.5.1 Templates

Everything on the provisioning interface is displayed through templates in the <installation directory/src/jcu.dc24.provisioning/jcudc24provisioning/templates>:

- **custom\_widgets** are templates developed for specific situations or pages and aren't related to the frameworks used (deform).
- **widgets** are default framework (deform) templates, most have been modified in some way (mostly for placeholder text, help text and descriptions).
- templates in the base directory are individual pages or part of the framework for rendering pages.

custom\_widgets and widgets control how form elements are displayed more-so than what is displayed such as a text input is displayed using the help, description and value as provided by the model (eg. text isn't hard-coded).

Templates in the base directory are mostly hard-coded such that you can directly edit the text seen on the website.

### 2.5.2 Models (Database & Form)

Models are broken into 2 basic classifications:

1. Project models are held in **src/jcu.dc24.provisioning/jcudc24provisioning/models/project.py** and describe how all forms are displayed as well as all project related database tables.
2. **src/jcu.dc24.provisioning/jcudc24provisioning/models/website.py** contains general website forms and database tables (eg. authentication models).

For the most part you should be looking at:

- **ca\_help/help** sets the text or HTML code under the ? symbol for that form element.
- **ca\_description/description** set the text/HTML code that is displayed permanently below form element titles.
- **ca\_title/title** sets the displayed title.
- **ca\_placeholder/placeholder** sets the greyed out text that disappears when clicked.
- **ca\_force\_required** makes a form element required.

- If you see **ca\_child**... it means that this element is a sequence (can add elements by clicking the add button) and the added value will be used on the item added rather than the sequence itself.
- If you see **ca\_group**... it is grouping this and following elements together (for display purposes) and the set value will be used on the grouping rather than the element itself.

Specifics of all models and the underlying frameworks is relatively complicated, if you would like more information please refer to the developer guides.

**Note: All changes to models won't take effect until the application is restarted.**

### 2.5.3 Views (Display & Logic)

Views are also split between project pages and general website pages:

- **src/jcu.dc24.provisioning/jcudc24provisioning/views/view.py** contains general website pages such as the dashboard, login and help.
- **src/jcu.dc24.provisioning/jcudc24provisioning/views/workflows.py** contains project specific views such as creating new projects, configuring the project and managing project data.

Page help is set by the `page_help` value passed into `self._create_response(page_help=<help>)`, this value is usually set in a `page_help` variable at the top of the view.

Menu text for project pages is found at the top of the `workflows.py` page, contextual options are set in `WORKFLOW_ACTIONS` and project configuration menus are set in `WORKFLOW_STEPS`:

- **title** sets the menu text itself.
- **page\_title** sets the page heading as well as the browser tab text.
- **tooltip** sets the pop-up text when you hover the mouse over the menu.

For more information please refer to the developer guide.

**Note: All changes to views won't take effect until the application is restarted.**

### 2.5.4 Stylesheet

The stylesheet is located at:

`src/jcu.dc24.provisioning/jcudc24provisioning/static/template.css.`

Stylesheets are used in almost all websites and there are many CSS guides available on the web for reference.

A quick hint is to right click on the element you want to change and click inspect element or similar (supported in most modern browsers), look for a CSS button and you will see what/how the website elements look is being set.

## 2.6 User Roles & Permissions

By default there are 2 roles available:

- ADMINISTRATOR's have all permissions except `EDIT_PERMISSIONS` and `DELETE`.
- SUPER\_ADMIN's have all permissions.

There are also 2 dynamic roles that aren't assigned to users:

- AUTHENTICATED is any user that has logged in.
- CREATOR is the user that created the project.

Default roles, permissions and authentication is configured in `src/jcu.dc24.provisioning/jcudc24provisioning/controllers/authentication`.

Actual permission, rolem user and other authentication databases are located at `src/jcu.dc24.provisioning/jcudc24provisioning/models/website.py`.

At the stage of writing there is no user interface for managing user permissions and roles (except the sharing page which are only a subset and per-project). Configuration will need to be done on the database directly.

## 2.7 Managing Templates

There are two types of templates within EnMaSSe as well as standardised fields (or parent data schemas):

- Project templates allow the pre-filling of all data in a project.
- Method templates pre-fill both methods and any datasets associated with them.
- Standardised fields are pre-made data configurations that allow users to select common types of measurements/fields easily. This not only encourages users to standardise their data but also increases the ease of searching the data portal.

Templates provide administrators a generic way of providing users with partially complete projects and methods/datasets.

The way templates work is by using a preconfigured database entry to pre-fill newly created projects or methods and datasets.

### 2.7.1 Project Templates

At the time of writing there is no administration interface for project templates and the database will need to be edited directly.

To make a project into a template a new entry needs to be added to the `project_template` table linked to the relevant project.

Project templates should have an appropriate name and description and they can be grouped by using the same category.

All fields in the associated project will be pre-filled on any project that uses this template.

### 2.7.2 Method Templates

At the time of writing there is no administration interface for method templates and the database will need to be edited directly.

To make a method into a template a new entry needs to be added to the `method_template` table linked to the relevant method.

Method templates should have an appropriate name and description and they can be grouped by using the same category.

Method templates may also be linked to a dataset.

All fields in the associated methods and datasets will be pre-filled on any method that uses this template.

### 2.7.3 Standard Data Configuration Fields

At the time of writing there is no administration interface for method templates and the database will need to be edited directly.

To make a data configuration into a standardised field its `method_schema` row needs to be updated with a `template_schema` value of 1.

Standardised fields are hierarchical such that each data configuration may have many parents each of which add their associated `method_schema_field`'s and parent `method_schema`'s.

This means that similar data configurations can be reused for similar data streams and their data would be cross searchable. Or common data configurations could be standardised so that no custom configuration is needed at all for compatible methods.





---

## EnMaSSe Developer Guide

---

*This guide explains the project structure and implementation details of the Environmental Monitoring and sensor storage (EnMaSSe) application.*

*This guide doesn't cover how to use the system or how to do general administration and it is recommended that administrators first read the user and administrator guides.*

The EnMaSSe application is designed to provide a user friendly interface for:

- Flexible and scalable research data ingestion (both streamed or manually input).
- High quality, fine grained, project based, metadata creation and export (eg. Enter 1 record, export many).
- Administerable and maintainable project lifecycle and workflows.

The project has been developed in three components:

- **Provisioning interface** provides a user friendly experience for creating metadata records, getting data type and ingestion configurations and managing the project.
- **Ingestor platform** handles data ingestion, indexing and storage.
- **Ingestor API** provides separation of the ingestor platform from the provisioning interface.

EnMaSSe integrates with the following 3rd party components:

- **CoastalCOMS Digital Asset Manager (CC-DAM)** as scalable, persistent data storage (used by the ingestor platform).
- **ReDBox** is an open source metadata repository that among other things, publishes metadata records nationally to Australian National Data Services (used by the provisioning interface).

Other core frameworks used by EnMaSSe include:

- **Pyramid** is the web framework used by the provisioning interface.
- **Deform** provides python object based form creation.
- **SQLAlchemy** is used for object oriented database access.
- **ColanderAlchemy** wraps both Deform and SQLAlchemy models together so the same models are used for forms and database access.
- **MySQL** is the database used in both the provisioning interface and ingestor platform (not the ingestor platform only uses it for local configurations not actual data storage).
- **Shibboleth** is a federated single sign-on framework that provides secure and controlled authentication, and release of user attributes.
- **UWSGI** is a more efficient web container for Pyramid than alternative HTTP containers.

- **XMLRPC** is used by the `ingesterapi` and `ingester` platform to provide platform independent programming interfaces.

*It is recommended that you read the `Ingestor API` first.*

## 3.1 Ingestor API

The project has been developed with the Ingestor API to increase its flexibility, the Ingestor API integrates the provisioning interface and ingester platform or in other words it separates the user interface and project functionality from the data and data ingestion/streaming.

The ingester API is object oriented where most functionality is implemented through generic model methods (eg. insert, update post, ...) and there are a handful of methods for specific functionality.

Most of the API functions can be used within a unit of work allowing for many calls to be used atomically (eg. all work or all fail rather than failing half way through and leaving the system in an unknown state).

### 3.1.1 Project Structure

The ingester API source files are structured with general API code in the base folder, all models in the models folder and schema definitions within the schemas folder.

The main API file that contains all methods is `ingester_platform_api.py`.

### 3.1.2 API Methods

API Methods are all implemented in the `ingester_platform_api.py`, this file also contains other useful classes such as `Marshaller` which converts ingester API models to and from dicts.

IngestorPlatformAPI methods:

<code>__init__</code>	Requires the data layer's address as well as an authentication object.
<code>ping</code>	A simple diagnostic method which should return "PONG"
<code>post</code>	Create a new entry or update an existing entry using the passed in object, the entry type will be based on the objects type (Included in <code>UnitOfWork</code> ).
<code>insert**</code>	Create a new entry using the passed in object, the entry type will be based on the objects type (Included in <code>UnitOfWork</code> ).
<code>update</code>	Update an entry using the passed in object, the entry type will be based on the objects type (Included in <code>UnitOfWork</code> ).
<code>delete</code>	Delete an entry using the passed in object, the entry type will be based on the objects type (Included in <code>UnitOfWork</code> ).
<code>search</code>	Not implemented yet.
<code>commit</code>	Commit a unit of work (Included in <code>UnitOfWork</code> ).
<code>enable-Dataset</code>	Enable data ingestion for this dataset (Included in <code>UnitOfWork</code> ) .
<code>disable-Dataset</code>	Disable data ingestion for this dataset (Included in <code>UnitOfWork</code> ).
<code>getIngester-Logs</code>	Get all ingester logs for a single dataset.
<code>getRegion</code>	Get the region object specified by the given id.
<code>getLocation</code>	Get the location object specified by the given id.
<code>getSchema</code>	Get the schema object specified by the given id.
<code>getDataset</code>	Get the dataset object specified by the given id.
<code>getDataEntry</code>	A data entry is uniquely identified by dataset id + data entry id.
<code>reset</code>	Resets the service
<code>findDatasets</code>	Search for datasets
<code>createUnitOfWork</code>	Creates a unit of work object that can be used to create transactional consistent set of operations.
<code>close</code>	Close should always be called when finished with the <code>IngesterPlatformAPI</code> to allow it to clean up any related data.

### 3.1.3 Models

The ingester API contains a number of models for representing the different types of data and configurations required for ingestion and storage.

Data Entry	Data Entries model the actual data that is ingested, they require a valid timestamp, location and dataset and their data attribute will contain the ingested data with indexes as configured by the datasets' <code>DataEntrySchema</code> .
Data Sources	Data sources configure how data can be ingested. Implemented data sources include <code>DatasetDataSource</code> , <code>PullDataSource</code> , <code>PushDataSource</code> , <code>SOSScraperDataSource</code> and <code>FormDataSource</code> . Each of the data sources mentioned require specific configurations.
Dataset	A Dataset represents a collection of data associated with a sensor or data method and hold the full configuration for an ingester. All <code>DataEntry</code> 's are associated with a Dataset.
Locations	There are currently two types of location which are <code>Location</code> and <code>OffsetLocation</code> . Locations represent a point on earth and there must be a <code>Location</code> object associated with each Dataset and there may be an <code>OffsetLocation</code> associated with a Dataset. <code>OffsetLocations</code> provide an offset from the set <code>Location</code> .
Region	Represents a 2D area on earth (eg. Queensland)
Metadata Entry	Metadata is extra information (eg. QA information) that can be attached to other models, this is useful as a flexible way of adding real world information when the requirements are unknown at the time of developing.
Sampling	Some data sources poll external systems and <code>_Sampling</code> objects indicate when this should occur. There is currently only one implementation ( <code>PeriodicSampling</code> ) that samples at a set interval.
Search Criteria	Search criteria is intended to provide advanced search features but hasn't been fully implemented at the time of writing.
Ingestor Log	Represents a single dataset and contains the information required to ingest the data as well as location metadata.

### 3.1.4 Schemas

Schemas allow flexible data storage configuration and are all based from the `Schema` class in `schemas/__init__.py`.

Schemas are minimal objects that can be extended with new (typed) attributes as well as inheriting attributes from parent schemas.

Schemas are generated from the data configurations section on the methods page of EnMaSSe.

There are three subclasses of `Schema`:

1. **`DataEntrySchema`** is the primary schema that is used for all data configurations.
2. **`DataEntryMetadataSchema`** can be used to attach metadata (eg. QA information) to data.
3. **`DatasetMetadataSchema`** can be used to attach metadata to datasets.

All schemas begin with 3 attributes:

- `name`
- `id`
- `version`

Additional attributes can be added using the `addAttr()` method, new attributes must be instances of `DataType` and will typically include:

- `description`
- `name`
- `units`

Parent schemas can be added using the `extends()` setter, the provided values must be a list of valid schema id's.

## 3.2 Developing the Provisioning Interface

The provisioning interface is based on a number of frameworks and integration with external components, so the first step to development is gaining a high level understanding of:

Frame-work	Why you need to know?	Description
Pyramid	How the application works as a whole. How views are set-up.	Web framework used by the provisioning interface.
Deform	How models are turned into forms. Editing of models.	Provides python object based form creation.
SQLAlchemy	How models represent the underlying database. Editing of models.	Used for object oriented database access.
ColanderAlchemy	Model attributes and how models relate to Deform as well as SQLAlchemy.	Wraps both Deform and SQLAlchemy models together so the same models are used for forms and database access.
MySQL	In-depth debugging of database issues.	Database used in both the provisioning interface and ingester platform (not the ingester platform only uses it for local configurations not actual data storage).
ReD-Box	High level understanding of the metadata records. Debugging metadata record export.	The metadata repository that publishes records to Australian National Data Services (ANDS).
In-gester API	In-depth understanding of data ingestion and storage. Debugging ingester issues.	Used to integrate with the ingester platform (ingests and stores data).
Tal/Metal	Customisation of template logic and dynamic data.	Allows python data and minor logic to be used in the HTML templates.
jQuery	Customisation of template scripting.	Javascript library used in most templates.
Open Layers	Understanding and customising of embedded maps.	Javascript library used to display maps within the information and datasets pages.

### 3.2.1 Framework Modifications

#### SQLAlchemy + Deform + ColanderAlchemy

In short `ColanderAlchemy` allows a single model for both `Deform` and `SQLAlchemy` by creating `SQLAlchemy` models using drop in replacements for Column and relationship with custom deform attributes.

The Deform schema is generated using `SQLAlchemyMapping(<model class>)`, this schema is then used to display forms and data read from the database could be displayed by calling the `dictify()` method and passing it to the `form.render()` method.

#### ColanderAlchemy Modifications & Usage

The EnMaSSe system required a number of extra features that have been implemented in the `models/ca_model.py`, `views/ca_scripts.py` and template files:

- **CAModel provides additional functionality to `ColanderAlchemy` models when added as a parent (eg. `MyModel(CAModel`**
  - `dictify()` with support for other EnMaSSe specific changes.

- `update()`
- `to_xml()`
- `ca_scripts.py->convert_schema()` alters the ColanderAlchemy generated schema to add additional display options including
  - Removing the top level title (ColanderAlchemy outputs a schema with a name at the top of every form).
  - Removing items not on the specified page (based on attributes on schema nodes).
  - Allowing form elements to be required.
  - Grouping of nodes under a MappingSchema for display purposes.
  - Prevent duplicate field names causing problems by adding parent schema names separated by ‘.’.
  - Removal of advanced/restricted fields based on a passed in parameter (this could be upgraded to integrate with the Pyramid permissions system).
  - There is also a `fix_schema_field_name()` method that reverts names to their original value.
- Most template files have been modified to support or modify help, description and placeholder text. Other features have been implemented on a per template basis as well.

Usage of the modified ColanderAlchemy models has been made as straightforward as possible:

- Extend all database/form models from both CAModel and SQLAlchemy Base (in that order).
- Wrap the ColanderAlchemy schema generation with `convert_schema` giving `convert_schema(SQLAlchemyMapping(<model class>), **kw)`.
- Use the additional custom attributes supported by the modifications.

Custom attributes include:

Attribute	Description
<code>ca_help</code>	Shows the value as HTML code hidden under a ? symbol next to the elements title.
<code>ca_placeholder</code>	Displays greyed out text that disappears when clicked (only text elements).
<code>ca_child_</code>	Uses the following attribute on children rather than the element it is added to (only mapping or sequences), for example <code>child_title</code> would set the title of all child schema items.
<code>ca_force_required</code>	Set the form element as required.
<code>ca_restrict_admin</code>	Remove this field if <code>convert_schema()</code> is called with <code>restrict_admin=True</code> .
<code>ca_page</code>	Remove this field if <code>convert_schema()</code> is called with a page value that is different.
<code>ca_group_</code>	Uses the following attribute on the parent group rather than the element it is added to, for example <code>group_title</code> would set the title of the surrounding mapping (This is mainly used in conjunction with <code>group_start</code> ).
<code>ca_collapsed</code>	A +/- symbol is displayed next to the mapping’s title, when this symbol is clicked it shows/hides respectively (only mapping’s). If the value is <code>True</code> the mapping will be hidden on page load otherwise it will be displayed.
<code>ca_group_start</code>	Surround this element and following elements in a mapping (for displaying grouped elements together). Other <code>group_</code> attributes would typically be used to improve the mappings display such as <code>group_title</code> , <code>group_help</code> , <code>group_collapsed</code> . The value identifies which mapping for the closing <code>group_end</code> .
<code>ca_group_end</code>	Ends surrounding mappings created with a <code>group_start</code> attribute. If there is no open group than this attribute will do nothing. If there is a second group mapping nested within this group it will close both.

**Note:** All attributes on ColanderAlchemy\_ models are appended with `ca_`, if Deform models are used directly the `ca_` should be removed from attributes.

## Customised Deform Widgets & Templates

There were a limited number of custom **Deform** widgets created which are located in:

- **models->file\_upload.py** has minor changes to `FileUploadWidget` for Provisioning Interface specific storage.
- **views->deform\_widgets.py** implements:
  - **ConditionalCheckboxMapping** which hides/shows the following **Deform** element based on if the checkbox is selected.
  - **MethodSchemaWidget** which provides customised functionality for custom fields and standardised fields.

Most of the customisations were implemented in the templates and without going through each of the 50 or so files customised the general pattern followed was:

- If it is functionality specific to the Provisioning Interface the template goes in `templates->custom_widgets`, otherwise they go in `templates->widgets`.
- Copy the closest **Deform** template and modify to add required functionality.
- If the template requires information that isn't passed in from the widget, directly add it to the schema in the view and access it from the template through `field.schema.<name>`.
- **Javascript** for widgets that occur many times is implemented as a function in `static->scripts->widgets.js`, Javascript that can only occur once may stay in the template.

### 3.2.2 Project Structure

The Provisioning Interface structure is organised as:

- **data** - session data and isn't used by Provisioning Interface otherwise.
- **jcudc24provisioning** - contains all Provisioning Interface source code
  - **Views** - all served websites, these are closer to view-controllers though as they contain much of the applications functional code.
  - **Models** - all database and form models.
  - **Controllers** - additional functional code that doesn't fit in the views, this is mostly code for integrating with other systems.
  - **Scripts** - initialisation command line scripts that are output to the bin folder when the project is deployed.
  - **Static** - static resources such as images, css and javascript/javascript libraries.
    - \* **css**
    - \* **images**
    - \* **libraries** - Javascript libraries
    - \* **scripts** - custom Javascript
  - **Templates** - **ZPT** template files used by **Deform** widgets and the main website pages.
    - \* **custom\_widgets** - custom, Provisioning Interface specific widget templates
    - \* **widgets** - customised **Deform** widget templates.
- **project\_uploads** - Attachments that are uploaded through the provisioning interface pages.

- **tmp** - temporary files such as for exporting to external systems.

The most relevant files are:

- **\_\_init\_\_.py** initialises the database, configures all views and sets up authentication. Look at this view to find out what URL's associated with views and to add new views.
- **resources.py** packages static resource files for [Fanstatic](#).
- **views->views.py** contains all non-project views.
- **views->workflows.py** contains all project views.
- **scripts->initializedb.py** populates the database the first time the application starts.
- **scripts->create\_redbox\_config.py** creates the XML mapping file for [ReDBox](#) integration.
- **models->ca\_model.py** wraps CAModel to provide transparent model-dict-xml conversions.
- **models->project.py** contains all project related database and/or form models.
- **models->website.py** contains all non-project database and/or form models.
- **controllers->redbox\_mint.py** is the code that handles the metadata exports to ReDBox.
- **controllers->ingesterapi\_wrapper.py** wraps the [Ingester API](#) for transparent use with Provisioning Interface models.

### 3.2.3 Metadata Records (ReDBox)

EnMaSSe integrates with [ReDBox](#) using the new-alerts harvest system by:

1. Creating an XPATH mapping file between the XML export file and [ReDBox](#) fields.
2. Writing the Provisioning interface Metadata table to an XML file.
3. Transferring the XML metadata to the [ReDBox](#) server using SFTP.
4. Hitting the new-alerts URL which tells [ReDBox](#) to run the harvest.

#### Code that handles ReDBox integration in EnMaSSe

[https://github.com/jcu-ereseach/TDH-rich-data-capture/blob/master/jcudc24provisioning/scripts/create\\_redbox\\_config.py](https://github.com/jcu-ereseach/TDH-rich-data-capture/blob/master/jcudc24provisioning/scripts/create_redbox_config.py)

This creates the XPATH mappings from the XML to ReDBox fields and is basically a hard-coded field-to-field mapping but it uses the model attribute names (eg. field names can change, but new/delete fields need to be remapped).

[https://github.com/jcu-ereseach/TDH-rich-data-capture/blob/master/jcudc24provisioning/models/ca\\_model.py#LC487](https://github.com/jcu-ereseach/TDH-rich-data-capture/blob/master/jcudc24provisioning/models/ca_model.py#LC487)

`to_xml()` is what converts the Metadata object to XML.

<https://github.com/jcu-ereseach/TDH-rich-data-capture/blob/master/jcudc24provisioning/models/project.py#LC462>

The `Metadata()` model represents a metadata record, so the provisioning interface sets up all relevant data then just dumps it to an XML file.

#### How ReDBox is updated to integrate with EnMaSSe:

<https://github.com/jcu-ereseach/TDH-Research-Data-Catalogue/tree/master/src/main/config/home/harvest/enmasse-alerts>

This folder contains the harvester structure:



- New records are placed directly in this folder
- **The config folder contains harvester configuration and modifications:**
  - `enmasse-dataset-rules.py` is a customised copy of the `new-alerts dataset-rules.py` file (ID fixes, harvesting directly to published, etc.) which uses the data found with the `XMLAlertHandler` and adds it to `ReDBox` correctly.
  - `enmasse-dataset.json` is the configuration file for the harvester.
  - `enmasseXmlMap.json` is the EnMaSSe mapping file generated from `scripts->create_redbox_mapping.py`.
- Processed harvests go into the `ReDBox` created `.processed` folder (not in repo).

<https://github.com/redbox-mint/redbox/tree/master/config/src/main/config/home/lib/jython/alertlib>

This is the code base for the new-alerts system.

Also, the `system-config.json` file needed to be updated with the following 2 sections:

- <https://github.com/jcu-ereseach/TDH-Research-Data-Catalogue/blob/master/src/main/config/home/system-config.json#LC372>
- <https://github.com/jcu-ereseach/TDH-Research-Data-Catalogue/blob/master/src/main/config/home/system-config.json#LC432>

### 3.2.4 Ingestor Platform Integration

The Provisioning Interface was developed in conjunction with the Ingestor API and Ingestor Platform so the database models and functional concepts are very similar.

This has made the integration particularly easy as all communication with the Ingestor API maps closely to Provisioning Interface database models and the main steps required are:

- Converting Provisioning Interface models to their associated `Ingestor API` models (the Ingestor API models weren't used directly as the Provisioning Interface requires a lot of additional display information).
- Communicating many associated models at the same time (such as all datasets associated with a project).

Full integration with the `Ingestor API` has been implemented almost transparently with the `controllers->ingesterapi_wrapper.py` by extending `IngestorPlatformAPI` to process Provisioning Interface models passed to it's method by:

- Using a unit of work to convert the model itself as well as all child models to Ingestor API models.
- Providing any mappings of slightly different functionality between the models (such as parsing data source `script+parameters` into a string).
- Updating `Ingestor Platform` ID's on the models once the unit of work has successfully been committed.

### 3.2.5 Shibboleth Authentication

`Shibboleth` is a federated single sign-on framework that provides secure and controlled authentication, and release of user attributes. Users are redirected to their home organisation identity provider (IdP), where they supply their passwords, and then organisation policies are consulted during the release of the user's attributes.

A `Shibboleth` enabled website is referred to as a service provider (SP). The front end webserver such as IIS or Apache HTTPD, provides a number of Shibboleth end points that are used to communicate from the IdP to the SP. Once the `Shibboleth` session is established authentication and attributes can be passed to back end application servers by securing a path with `Shibboleth`. The attributes can be passed as environment variables or HTTP headers, however

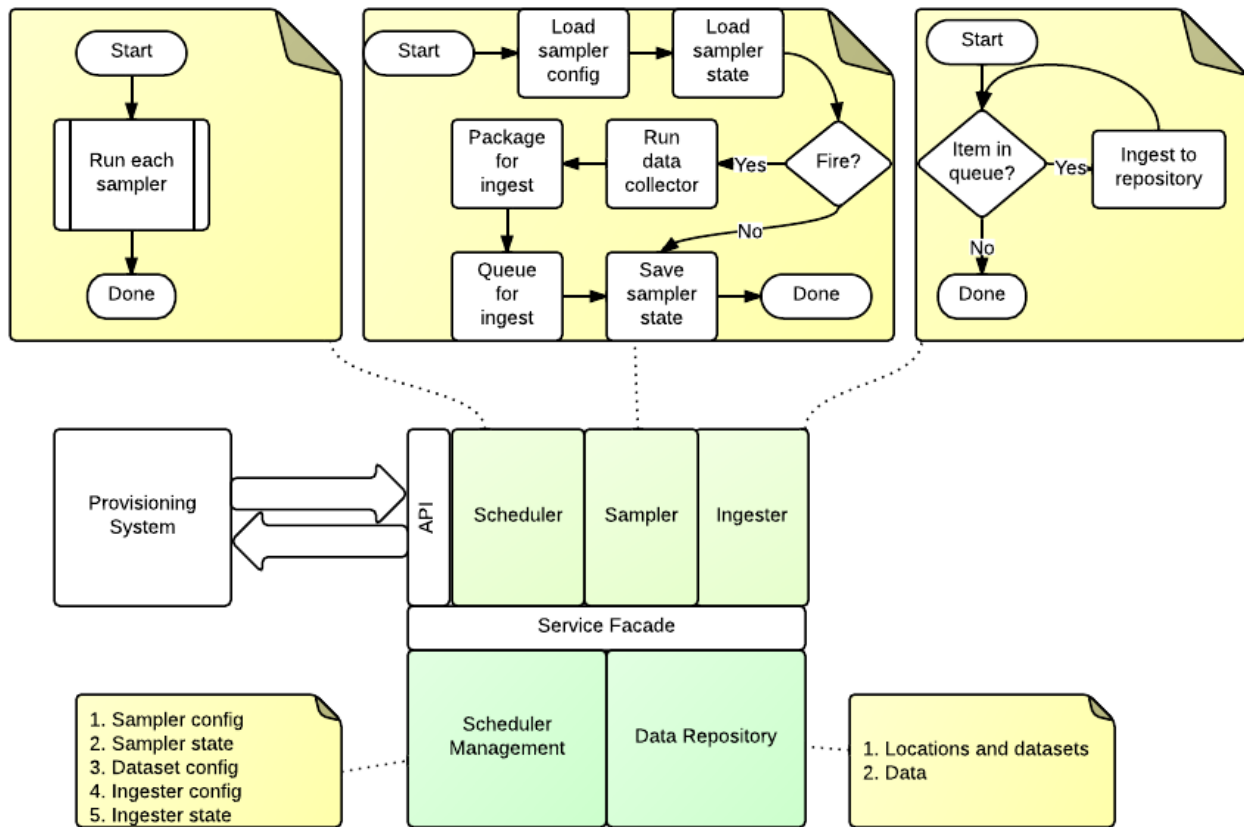
the EnMaSSe provisioning interface only supports the use of HTTP headers. Note, this assumes that the connection between the front end web server and the back end application server is secure. In particular, direct access to the application server must be blocked to prevent fake HTTP headers being injected.

This document assumes that [Shibboleth](#) is already setup on the front end web server, as configurations and requirements will vary between [Shibboleth](#) federations. The protected path within the Provisioning Interface is /login/shibboleth. This SP will require the following attributes:

- firstName
- surname
- commonName
- email
- auEduPersonSharedToken

The final attribute, auEduPersonSharedToken, is a globally unique identifier for the user, and is what is used to link the [Shibboleth](#) account to the local account, as well as to accounts in other repositories.

### 3.3 Developing the Ingestor Platform



#### 3.3.1 Domain Objects

The domain of the ingestor platform is that of the projects, schemas, datasets, and ingestor configurations that are to be collected.

### **Project Metadata: Region**

Not implemented

This represents a 2D area of the Earth's surface.

### **Project Metadata: Location**

This represents a point on the Earth's surface. The point has an elevation above MSL, and optionally be part of a region.

### **Project Metadata: Schema**

Describes the fields that can be part of a metadata entry. The currently supported datatypes are: \* File \* String \* Integer \* Double \* Timestamp

Schemas support multiple inheritance, but all fields must be unique within the schema hierarchy. It is expected that you can search any schema, including base schemas, so any repository implementation will need to take this into account.

### **Project Metadata: Dataset**

A dataset represents a collection of data entries at a particular location, that conform to a particular schema. Optionally they will include a data source and sampler to populate them. The location is considered to be the frame of reference for the dataset. Additional metadata may be applied to the dataset to convey specific location information.

### **Ingestor Metadata: Data Sources**

Data sources retrieve data from a remote source, and processes it to a data entry for ingestion into the repository.

#### **URL Pull Data Source**

This data source pulls a single resource from a specified URL (File, FTP, HTTP) and attaches it to the specified field of a new data entry. In the case of HTTP the Last-Modified header is used as the data entry timestamp. This will map the retrieved file to one specific field and produce one data entry.

#### **Dataset Data Source**

Gets data from another dataset in the repository. Will download the data from the data entry. By default no data entry is produced.

### **Ingestor Metadata: Samplers**

Samplers control when an ingestion event occurs in the case of a pull ingestion.

#### **Periodic Sampler**

The periodic sampler fires at a predetermined rate, specified in seconds.

#### **New Data Sampler**

Triggers whenever there is new data in another dataset. Passes the data entry ID and dataset as metadata to the data source. Only used with Dataset Data Source.

### 3.3.2 Ingestor Service Components

The ingestor platform is build using the <http://twistedmatrix.com/> framework, which is a reactor framework. This means that requests and services started using periodic schedulers are serviced by a single thread. This has scaling implications as blocking processes such as ingesting large amounts of data will affect API clients. To overcome this the Ingestor Platform has two services threads: the data source service; and the ingestor service. Both are part of the IngestorEngine class.

The data source service is fed through a queue by the processSampler method. The the running state of the data source is recorded to prevent multiple concurrent copies of a data source running. Once the data is collected and processed it is placed in an ingest queue.

The ingestor service thread is fed from the ingest queue.

#### processSamplers - Main thread

**This method is called periodically from the Twisted callback. Its purpose is to datasets that are ready to ingest.**

1. Load active datasets
2. For each sampler
  1. Skip if there is no sampler or if the ingestor is running
  2. Load sampler state (processSampler)
3. Run sampler
4. Persist sampler state
5. Queue dataset if sampler returned True

#### processQueue - Data source thread

**This reads the dataset queue for new ingesters to run. It relies on the data source always returning or timing itself out.**

1. Load data source state
2. Create temporary directories
3. Run data source
4. If there is a post processing script run it
5. Queue output data for ingest
6. Persist data source state

#### processIngestQueue - Ingest thread

**This thread ingests data into the repository. It relies on the repository access being thread safe.**

1. For each data entry
  1. Ingest into repository

### 3.3.3 Ingesters

An ingester is an object that is invoked by the ingester platform to assemble zero or more Data Entries. Each ingester will need a domain object in the `jcudc24ingesterapi.models.data_sources` module, and an implementation in the `dc24_ingester_platform.ingester.data_sources` module. The domain object will need to have each of the valid configuration parameters as Python properties, or our `typed` properties.

#### Testing from the command line

To test an ingester you can run it from the command line. To do this you will need to create a config file, a working directory, and then invoke it using the `run_ingester <config> <cwd> [script]` script. The script takes 2 mandatory arguments, and one optional argument. These are

- An ingester config file
- A working directory
- And optionally, a post processing script.

A sample config is

```
{
  "class": "pull_data_source",
  "state": {},
  "parameters": {},
  "config": {"url": "http://www.abc.net.au", "field": "page"}
}
```

Then, to run you could call: `run_ingester pull.json /tmp` then you should see an output such as

```
Initial results
-----
Time: 2012-12-19T22:52:42.000Z Dataset: None
    file = FileObject(f_path: outputfile0, mime_type: )

Time: 2012-12-19T22:56:32.000Z Dataset: None
    file = FileObject(f_path: outputfile1, mime_type: )

...
Processed results
-----
Time: 2013-02-06T10:32:04.493Z Dataset: None
    temp = 29.9375

Time: 2013-02-06T10:32:04.493Z Dataset: None
    temp = 29.75

...
```

### 3.3.4 Ingestor Post Processing Scripts

A post processing script is called with a workspace directory (`cwd`), and the data entry object that is being processed. The returned data entries are those which will be actually ingested into the dataset:

```
from jcudc24ingesterapi.models.data_entry import DataEntry, FileObject
```

```
def process(cwd, data_entry):
    data_entry = data_entry[0]
    ret = []
    with open(os.path.join(cwd, data_entry["file1"].f_path)) as f:
        for l in f.readlines():
            l = l.strip().split(",")
            if len(l) != 2: continue
            new_data_entry = DataEntry(timestamp=datetime.datetime.now())
            new_data_entry["a"] = FileObject(f_path=l[1].strip())
            ret.append( new_data_entry )
    return ret
```

### 3.3.5 Search Interface

A search will return a specific object type, based on a set of criteria that may reference other related objects. Complex searches could be constructed using using a search tree marshalled to a prefix notation, and then used to generate the search queries. Complex situations may arise when crossing between ingester and repository objects.

### 3.3.6 Repository Integration

The ingester platform requires a repository in which to store the data that it processes. The repository gets notified when any action occurs on project metadata with an opportunity to store references back in the ingester platform database. The repository is also exclusively used for the storage of data. It is expected to support at least data storage and retrieval methods.

#### DAM Integration

The DAM is the preferred repository for the ingester platform. May of the project metadata object map directly to the DAM metadata objects. If only value criteria are allowed, then this could be resolved by first querying one system then the other. Initially only a subset of the attributes will be searchable.

Project Metadata	DAM Metadata
Region	•
Location	Location
Schema	Schema
Dataset	Dataset
DataEntry	Observation

### 3.3.7 Management API

The EnMaSSe Ingestor Platform has no specific user interface of its own, rather, all interactions occur using the web services API. The API provides methods for creating and managing all domain objects, and processes.

#### Object Manipulation

#### Management Processes

The main purpose of the EnMaSSe Ingestor Platform is to manage data ingestion. The following methods enable this.

### Enable Ingestion

```
enableDataset(dataset_id)
```

### Disable Ingestion

```
disableDataset(dataset_id)
```

### Reprocess Derived Data

```
runIngestor(dataset_id)
```

It is possible to manually trigger an ingestion of *derived data* in the situation where there is a *raw data dataset*, containing data files, and a *derived dataset* that uses the *raw data dataset* as a dataset data source. This has the caveat that if there is already data in the *derived dataset* then invoking this process may create duplicate data entries.

### Retrieve Ingestor Log

```
getIngestorEvents(dataset_id)
```

This method is used for retrieving all the available ingestor logs. These can give insight into if any why an ingestor process is failing.

## 3.4 Project Reusability

Full user, administrator and developer documentation is provided allowing full reuse of the developed project.

EnMaSSe has been designed to be as reusable as possible but it is a specialised tool with a small group of potential users:

- The full EnMaSSe system can be reused for other universities or research organisations.
- The ingestor platform and provisioning interface are integrated through the ingestor API, so either system could be reused separately.
- Extended form functionality could be pushed to the Deform and ColanderAlchemy projects.

### 3.4.1 EnMaSSe System

The project has been developed with a number API's to increase the flexibility of reuse:

- **ingesterapi** integrates the provisioning interface and ingestor platform or in other words it separates the user interface and project functionality from the data and data ingestion/streaming.
- The **data layer JSON-RPC API** separates the data storage from the data ingestion/streaming functionality.
- ReDBox is integrated with two scripts that exports rows from the provisioning interface metadata table data as XML records and generate ReDBox configurations (There are also ReDBox specific files external to the EnMaSSe system).

The only component that isn't open source is the CoastalCOMS Digital Asset Manager (CC-DAM) which is proprietary 3rd party software. CoastalCOMS is open to working with organisations to use their CC-DAM but the ingester platform is separated from the data layer through a JSON-RPC API and a custom data layer could be implemented relatively easily.

ReDBox is an open source project under active development for research organisations and it is likely that potential users will already have a ReDBox instance. Integration with ReDBox is implemented in 2 files:

- `src/jcu.dc24.provisioning/jcudc24provisioning/controllers/redbox_mint.py` which prepares the metadata records, writes them as XML files, uploads them to the ReDBox server and alerts ReDBox that there is new data available.
- `src/jcu.dc24.provisioning/jcudc24provisioning/scripts/create_redbox_config.py` generates XPATH mappings from the exported EnMaSSe XML files to ReDBox fields.

This means that not only is the ReDBox implementation separated from the provisioning interface but also that other metadata systems could be integrated with ease.

The real power and reusability of the EnMaSSe is directly streaming sensor data into a scalable data storage system in a highly flexible way:

- Reusable and extendable for almost any type of streaming sensor.
- Highly customisable data configurations and custom processing scripts mean the data storage can index the ingested data any way the user requires while the bulk of data is stored as flat files (more scalable).
- Custom processing scripts can be chained together allowing the data to be split and processed in many ways.

### 3.4.2 Ingestor Platform

The ingester platform is designed to facilitate the retrieval, aggregation, processing, and ingestion of data from disparate sources into a central repository. The platform supports a number of types of data source, providing a number of implementations for retrieving data from remote websites or webservices. As well, the data processing step is completely pluggable. Finally, repository access is abstracted so that new repositories can be easily supported. The implementations provided includes a reference implementation using an SQL database, and the CC-DAM repository.

#### Provisioning Interface

The problem with reuse of the EnMaSSe provisioning interface is that it is a wrapper of other components to provide a single, user friendly interface rather than a tool on its own.

To break reuse possibilities up the core things the provisioning interface does are:

- Creates project based metadata records (eg. enter one generate many).
- Provides an interface for generating data configurations (ie. database tables).
- Exports metadata to ReDBox.
- Exports ingestion configurations to the ingester platform.

#### Project Based Metadata

So if someone was looking to create a project based metadata tool with a different focus than ingesting streamed data they could reuse a large portion of the EnMaSSe project by:

- Editing the methods and datasets pages to implement their required functionality.
- Alter the submit page functionality to fit their required workflow.
- Alter the `workflows.py->generate_dataset_record` to generate child records as required,



- No alterations or fixes would be required to keep integration with ReDBox.
- Update the general website pages.

### **Database Schema/Form Creation Interface**

The data configurations is a component that could be reused separately either to create database schemas, forms or both.

With a bit of interface work this could be developed into a handy web development tool.

### **ReDBox Integration**

The ReDBox integration is a smaller, application specific component however it does provide a good example of integrating with the new ReDBox alerts harvester.

### **3.4.3 Deform Templates & Widgets**

To get the required functionality a decent amount of additional functionality has been implement as Deform widgets and templates.

The most notable improvements are the addition of help, description and placeholder to all elements where appropriate.

### **3.4.4 ColanderAlchemy Functionality**

ColanderAlchemy is a relatively young framework that joins SQLAlchemy with Colander (Deform uses Colander for it's models) which means that ColanderAlchemy models generate both user interface forms and database tables.

As part of the core functionality of the EnMaSSe provisioning interface, models and scripts were written for:

- Converting deform appstructs into ColanderAlchemy models.
- Converting ColanderAlchemy models into deform appstructs.
- Implements additional display options such as displaying specified fields as grouped.
- Converting ColanderAlchemy models to XML.

## **3.5 Information on Implementing Specific Functionality**

### **3.5.1 Adding Data Source's**

The most useful and generic data source currently implemented is the PullDataSource, so throughout this explanation we will use it as the example.

#### **Provisioning Interface**

1. Add a new entry to models->project.py->Method->data\_sources.
2. Update the description of models->project.py->Method->data\_source.
3. Copy/Paste models->project.py->PullDataSource and update as needed, this provides the data source configuration options on the datasets page.

4. Copy/Paste `models->project.py->Dataset->pull_data_source` and update for your newly created data source type.
5. Update `controllers->ingesterapi_wrapper.py->_create_data_source` to convert your new data source configuration to the corresponding *Ingestor API* data source model you have/will create.

## Ingestor API

1. Copy/Paste `models->data_sources.py->PullDataSource` and update with the fields and initialisers that your new data source requires.

## Ingestor Platform

**TODO: This section needs updating.**

### 3.5.2 Custom Import Scripts

See *Ingestor Post Processing Scripts* in the Ingestor Platform developers guide.

### 3.5.3 Adding New Custom Field Types for Data Configuration

1. Add the new field type value and name to `models/project.py->field_types`
2. Update `controllers/method_schema_scripts.py->get_schema_fields()` to create the correct:
  - field type
  - python type
  - widget
3. Update `controllers/ingesterapi_wrapper.py->process_schema()` so that the new field type is mapped to the ingester platform correctly.

Note: Please refer to the ingester platform documentation and source code for adding new data types to the ingester platform.

## 3.6 Limitations & Future Work

Here is a list of features and bugs that could be further developed:

- **Currently each method has a unique schema and each dataset has a unique location, even if they are identical with others.**
  - There are some difficulties with this such as what if a user edits the schema for one method.
  - What may work best is to use the same schema/location once submitted and approved (as they can't be changed after that).
- User widget re-write to allow for NLA/Trove lookup + better searching + affiliation select. Currently only JCU users in the Mint name authority are supported.
- Dashboard page with alerts/notifications
- Administrator page (roles/permissions, create/edit templates & standardised fields).

- Reading of licenses directly from ReDBox (they are currently hard-coded).
- Refactor code to import all text from resource files - this may improve the ease of administration/installation for new users.
- Drag-n-drop data configuration widget, it would be nice to mke the data configuration more intuitive, perhaps by minimising all fields except the one currently being edited + real time display of the generated form.
- Synchronise/validate provisioning and ingester schemas (make sure they are the same).
- Provisioning-ingester implementation of regions
- Breif description appears after full description in ReDBox records
- History for all fields in provisioning.
- Progress indicator that the submit page is still working when it is approved.
- Use Diazo for theming the website.
- Limitation of parent schemas only allowing a parent to be extended once (only 1 level of parents)
- unit types (eg temp, humidity, etc.) could be implemented for better searching support and replace the current purpose of parent schemas.
- Help and description printable for all workflow pages.
- (Needs testing, a fix was implemented) Trying to approve 2 projects at the same time may cause timeout errors (I don't anticipate this occuring too often but it is a bug).
- Attachments don't export to ReDBox
- Refactor searching/browsing/manage data into a restful framework - The view functions should be easy to convert to JSON based views if this is required.
- I beleive it would be useful to provide searching based on parties involved and locations within the data management sections
- Use templates for notification emails.
- Cannot add new methods to an active project.
- User must close browser to logout from shibboleth
- (Be aware) Using UTF-8 special characters can cause unexpected issues throughout the system because strings are read from the DB as str, not unicode.
- If the only data configuration is a single parent, use that parent schema directly.
- Only 1 default DataEntry and Dataset calibration is supported.



---

## Nomenclature

---

Within this document the following definitions are assumed.

Term	Definition
user	An end user, such as a researcher or student.
admin	A system administrator who will be responsible for installing, configuring, and administering the entire EnMaSSe platform.
developer	Someone with technical skills and understanding who will modify the source code of the EnMaSSe system.
EnMaSSe	The application this documentation is written for (it is a data and metadata collection tool).
Provisioning Interface	Internal component of the EnMaSSe application the provides the user interface (website).
Ingestor API	Internal component of the EnMaSSe application the provides the Application Programming Interface (API) between the Provisioning Interface and Ingestor Platform (how they communicate).
Ingestor Platform	Internal component of the EnMaSSe application the provides the data ingestion and storage.
Dataset	A collection of data from a method at a set location.
Method	A way that data is collected and stored.