Data Organization

Structs XINA groups employ certain organizational requirements to ensure they are interpretted correctly by structs API calls and front end tools.

Data Models

The primary organizational concept of the struct system is the **data model**. Abstractly, a data model (or simply **model**) is defined as having a set of **synchronously relevant data**. For example, a project might have a flight model, ETU model, etc. Models store data in independent databases, and multiple models may import data in parallel.

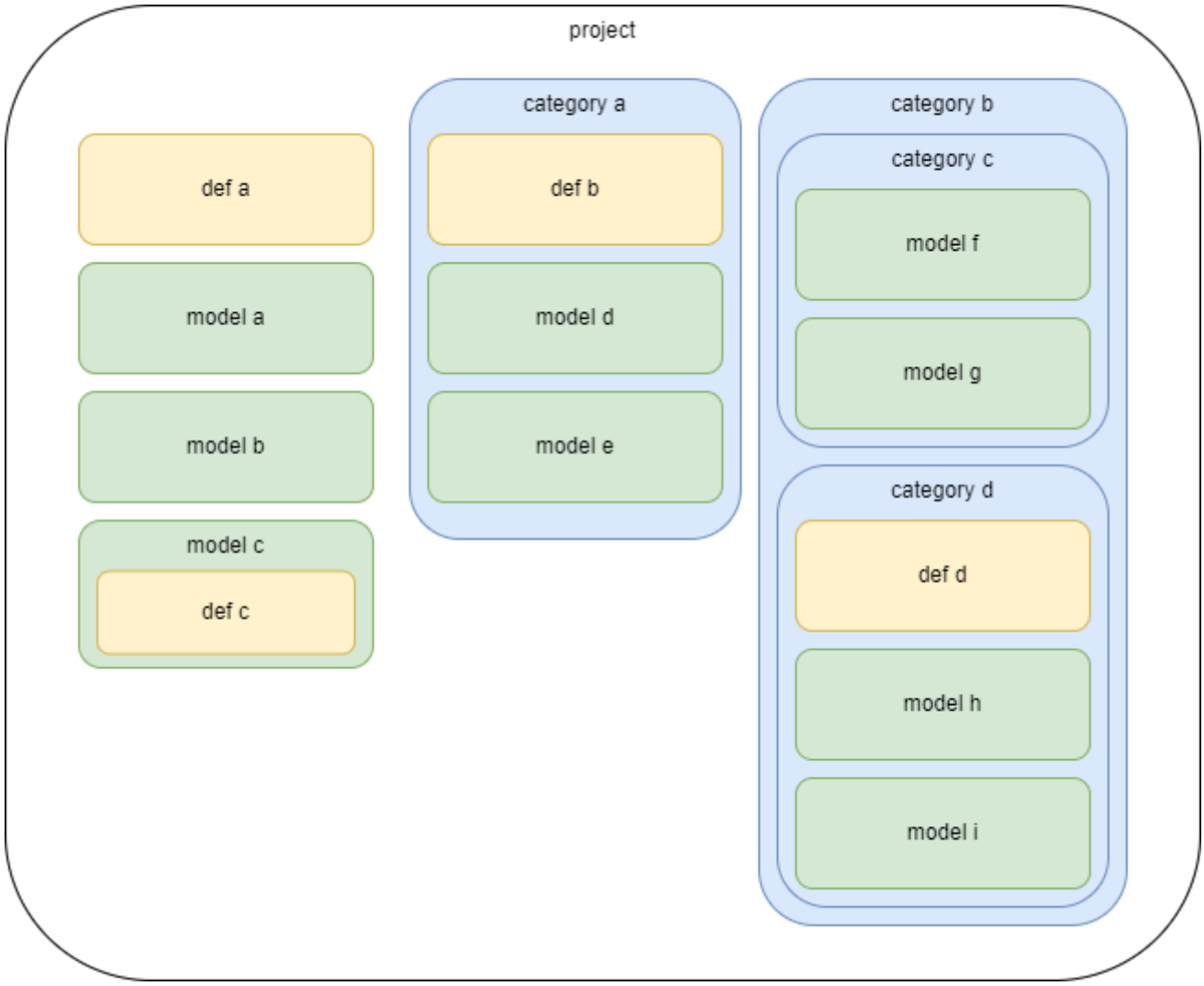
Broadly we use **time** as the primary method to organize and synchronize data within a model. In XINA this is represented as an 8-byte unsigned integer Unix time with microsecond precision. We use Unix time because it is:

- Widely and consistently supported
- Time zone independent
- Efficiently convertible to other formats and time systems

Other time formats may be available for data export depending on project requirements.

Projects / Categories

Projects and categories provide organization for multiple models. A project is a top-level group containing multiple models and/or categories, with each category containing multiple models or further categories. Project and category groups may also include additional groups and databases of data or resources which are not model specific, such as notebooks or definitions databases. In most cases with standard structures, models will default to databases or groups within the model, but search for them up the tree if not found. A complete project group might look like:



Note the definitions groups, which provide context for the data in models. A model is associated with the definitions defined by its closest ancestor. In this case, `"definitions a"` apply to models `a`, `b`, `f`, and `g`, `"definitions b"` to models `d` and `e`, `"definitions c"` to model `c`, and `"definitions d"` to models `h` and `i`.

In practice it is strongly recommended to use a single definitions group as broadly as possible to facilitate comparing data. Tools are designed to efficiently compare data (even across models) with the same set of definitions.

Model Organization

Data within a model falls into four primary classifications:

- **Telemetry**
  - source data file(s) from data collection point
  - typically stored in a raw (sometimes binary) format
  - storage cost is cheap
  - accessing data means downloading files or most likely requires custom XINA tools
  - may be divided into multiple **pipes** (see below)
- **Viewable Data**
  - extracted from telemetry into XINA database(s)
  - telemetry is the single source of truth for this data, not intended to be user editable
    - (except under controlled circumstances with struct API calls)
  - data is either **mnemonic**, **instant**, or **interval** (see below)
  - can be accessed and analyzed with built-in XINA tools

    - storage is expensive
    - optimizations may be needing depending on project requirements, data volumes
- **User Metadata**
    - additional data added by users, often directly through the XINA interface
    - XINA likely the primary repoository for this data
    - for example, a notebook
- **Definitions / References**
    - may be user entered or defined outside XINA
    - may exist at model level or above (category/project level)
    - more formal and restricted than user metadata

Pipes

Abstractly, a **data pipe** (or simply **pipe**) is a single point of data import to a model. In many cases, a model will only have a single pipe; for example, if all data is provided directly from a single instrument, or multiple components are merged into a single data stream through FEDS before import into XINA.

However, in environments with multiple import points running in parallel, databases must be designed with multiple origins.

In this example each source file would need to specify either `origin_a` or `origin_b`. Additionally, each origin has distinct databases for instant, interval, and mnemonic data. This would be required if each data source provided all three data types. As requirements for instants and intervals are less stringent than mnemonics, in some circumstances instants and intervals could be considered a single source and populated independently:

Revision #25
Created 28 July 2022 15:51:05 by Nick Dobson
Updated 27 August 2024 14:03:22 by Nick Dobson