# Data Models

General Concepts · https://namazustudios.com/docs/general-concepts/data-models/

Among the data models presented within Elements, several data models will contain the following properties. In all cases the properties represent the same consistent concept.

- **id** is always the system-assigned unique ID of a particular object.
- **name** represents a unique system-readable reference to the particular object.
- **metadata** represents user editable metadata that Elements will store with each object for more information on effective using metadata.
- **tags** are a list or set of words used to categorize and filter content.
- **displayName** represents a name used to refer to the object within the UI and potentially the final page or application.
- **timestamp** represents a time stamp or similar property.

## id Property

**id** always refers to the system-assigned unique ID of a particular object. Due to the server authoritative nature of Elements, the unique ID will always be the unique identifier for a particular object. In all cases, the identifier is generated server-side.

### Tips for Using the id Property

- Always treat IDs as opaque values. For example:
- Do not assume any particular format of the ID. At the time of this writing, all server IDs are hex strings. There is not a strict requirement for future editions of Elements.
- Some IDs may contain encoded data. Unless specifically documented, do not assume that any IDs will always follow the same encoding format.
- IDs are always handled as strings in request and response objects.

- The server will always generate IDs.
- Some APIs will accept either an ID or a name for certain operations. Unless otherwise specified, always pass the ID.

## **name** Property

**name** represents a unique system-readable reference to a particular object. The name assigned to any object is always intended to be an alternative unique id for the object. Understanding this is crucial for integrating backend scripts and client-side application code.

Unless otherwise specified, the name can be used as a programmatically unique id in all parts of the code. Additionally, two unrelated items may share a name. For example an Item and a Mission may share a name, however two Items may not share the same name. Names are case-sensitive and must be an alphanumeric string with no spaces.

### Tips for Using the **name** Property

- Use a consistent convention. For example, all lowercase letters with underscore separation. If, for example, you wish to model a Paper Hat as an item. A good name would simply be `paper_hat`. All code may not refer to this when fetching Paper Hat Items from the database.
- Use the name wherever possible, unless there is a specific need to use the database id. This ensures that the scripting engine code can refer to the specific object even if the object has been deleted and recreated, such as may be the case during migration between two separate instances of Elements.
- Give all objects a clear name that is obvious to the reader. This will avoid problems with ambiguity when designing your system.

## **metadata** Property

**metadata** represents user editable metadata that Elements will store with each item. This data may be used for any purpose and will not be modified or touched by Elements in any way. Depending on the circumstance, this may or may not be editable by regular users. At the time of this writing, it is not possible to index by metadata. At the time of this writing, all metadata must be written in a server-authoritative way. In other words, only superusers and scripts running in the backend may modify metadata. This may be subject to future enhancements.

In all cases, metadata is a multi-layered key-value object. The admin panel allows for simple editing of metadata as key-value strings, as well as editing inline JSON metadata. Some objects work with metadata schema, and full support for metadata schema is actively under development for all remaining system types.

### Tips for Using the metadata Property

- Metadata is not indexed, so you should not make objects searchable via metadata.
- Metadata is meant to be used to to add custom fields to objects that do not have sufficient support for data on their own. For example, when attached to a Profile, metadata may represent the current exp, level, or status of equipment slots.
- In general, the metadata should not include secure or sensitive information.

## tags Property

Tags are indexed for optimal querying from the database and should be used to quickly separate content when making API calls. Effectively using tags enables developers to quickly filter items as the tags are indexed and all APIs supporting tags support the querying of the tags. For example, items tagged with "armor" and "head" may represent helmets equipped to a character in a game.

### Tips for Using the tags Property

- When querying tags, they are always all-inclusive. For example if `paper_hat` has tags, `armor` and `head` will show up in searches for both `armor` and `head` only `armor`, or only `head`. It will never appear if `weapon` is specified in the search terms.
- There is no specific limit on tags, and while too many tags may run up against database limitations, practically speaking, Elements will allow far more tags than what is necessary for a majority of use cases.

## displayName Property

The display name is simply a name to use when displaying the object in the admin console or potentially to end users. The displayName should not be indexed and should never be used to make data decisions or in programmatic code. It's simply there for human readability and visual reference.

## timestamp Property

Elements represents timestamps and dates using the same notation used by the Java Virtual Machine. Timestamps are a 64-bit integer counting the number of milliseconds since the Unix Epoch (January 1, 1970). For more information see System.currentTimeMillis()