Metadata

Namazu Elements Core Features · https://namazustudios.com/docs/namazu-elements-core/schemas-and-metadata-specifications/

Metadata in Elements provides a flexible way to define and store structured information.
It consists of two parts:

1. **Metadata Specs** – define the *schema* (types and structure).
2. **Metadata Objects** – store actual data, validated against a spec.

This lets you enforce consistency across different parts of your game/application, while still allowing for flexible definitions.

Metadata Specs

A **Metadata Spec** is a schema that describes what a metadata object should look like.

Structure

A spec has:

- **name**: A unique identifier for the spec.
- **properties**: A list of fields, each with a type and optional values.

Property Types

Each property has a **name** and a **type**. Four types are supported:

Type	Description	Extras
STRING	A text field	Optional placeholder value
NUMBER	A numeric field (int or float)	Optional placeholder value

Type	Description	Extras
BOOLEAN	A true/false value	–
OBJECT	A nested structure containing its own properties	Properties can be any of the four types

Example Spec

```
{
  "name": "WeaponSpec",
  "properties": [
    { "name": "damage", "type": "NUMBER", "placeholder": 10 },
    { "name": "rarity", "type": "STRING", "placeholder": "common" },
    { "name": "isLegendary", "type": "BOOLEAN" },
    {
      "name": "stats",
      "type": "OBJECT",
      "properties": [
        { "name": "range", "type": "NUMBER", "placeholder": 100 },
        { "name": "weight", "type": "NUMBER" }
      ]
    }
  ]
}
```

Metadata Objects

A **Metadata Object** is an actual piece of data created based on a Metadata Spec.

Structure

A metadata object has:

- **name**: A human-readable identifier.
- **access level**: Controls who can see this object:
  - **Unprivileged** – visible to anyone.
  - **User** – visible only to logged-in users.

    - **Superuser** – visible only to administrators.

- **spec**: The Metadata Spec that defines its structure.
- **data**: A map of values that follows the rules from the spec.

Example Object

Using the `WeaponSpec` defined earlier:

```
{
  "name": "Excalibur",
  "accessLevel": "User",
  "spec": "WeaponSpec",
  "data": {
    "damage": 50,
    "rarity": "legendary",
    "isLegendary": true,
    "stats": {
      "range": 150,
      "weight": 5
    }
  }
}
```

Scoping

Metadata objects can be attached **globally** or **scoped** to specific domains within Elements:

- **Global** – accessible across the entire application.
- **User Profile** – metadata specific to a given user.
- **Application** – metadata tied to the app instance.
- **Items** – metadata attached to inventory/game objects.

This makes metadata a flexible tool for storing structured information at different levels of your system.

Access Control

When querying metadata objects, Elements automatically filters results based on the **user's access level**:

- **Unprivileged** users only see `Unprivileged` metadata.
- **Users** see both `Unprivileged` and `User` metadata.
- **Superusers** see everything.

This ensures sensitive or internal metadata is never leaked to unintended clients.

Common Use Cases

- Defining **item properties** (weapons, gear, consumables).
- Creating **application-level configs**.
- Attaching **custom profile attributes** to users.
- Storing **dynamic game data** (quests, NPC definitions, world states).

Summary

- **Specs** define *what* metadata looks like.
- **Objects** hold the actual *data*, validated against a spec.
- **Access levels** control visibility.
- **Scoping** lets you attach metadata where it's needed — global, user, app, or items.