

Metaschema Authoring Principles

The original OSCAL Technical Team had normalized on several guiding principles for authoring metaschema content that were not captured. Going forward, as topics come up, they will be added here as *suggested* principles and/or to capture historic guiding principles for awareness and discussion.

Nothing in this chapter is authoritative at this time, but could become authoritative in the future if reviewed and approved by an appropriate governing body.

- [Defining Allowed Values](#)

Defining Allowed Values

This page is still under development.

The `<allowed-values>` assembly provides a consistent and unambiguous list of **machine-readable tokens** to be used as data for an identified OSCAL field or flag values.

Human readability is coincidental and **not** their intended purpose.

Taxonomy

In metaschema, allowed values are a type of constraint applied to a metaschema flag or field. They are applied to the target of the `<constraint>`.

The `<allowed-values>` assembly consists of:

- an optional `@id` flag: A unique identifier within the OSCAL metaschema. Used for error reporting and debugging.
- an optional `@allow-other`: The default value is `no`, which means these values are strictly enforced and no other value is allowed. When `yes` other values are allowed.
- one `<enum>` element for each allowed value, containing:
 - a required `@value` set to the machine-readable token. **These are case sensitive**
 - required *text* content inside the element describing the concept represented by the token.

Example: Party Type Allowed Values

NEED DIFFERENT EXAMPLE

This example says, "`{{INSERT}}`".

In this example, the constraint's target is implied by its placement within the `{{INSERT}}`. Some constraint targets are explicitly defined using `<target />`.

Guiding Principles

When defining or modifying an allowed values set, the following principles should be applied to the greatest degree practical:

- **Mutually Exclusive:** Each allowed value should be cleanly distinct from any other allowed value in the set. There should be no ambiguity as to which value should be used.
- **Collectively Exhaustive:** Every possible concept should be represented.

When a set of values has been well researched and consensus agrees the set is both *mutually exclusive* and *collectively exhaustive*, the `allow-other` should be set to `no`.

When it is not practical to achieve *collectively exhaustive*, `allow-other` should be set to `yes`. This can happen when a an allowed value set needs to be able to grow to accomodate an evolving industry and when an exhaustive research effort to identify all possible values is not practical due to resource constraints.

Applying the Principles

The above example `{{INSERT}}`, and does not allow other values.

This is mutually exclusive in that it separates `{{INSERT}}`.

They reflect the concepts of `{{INSERT}}`.

It would ...`{{INSERT}}`

Reference

Read more about the [Mutually Exclusive / Collectively Exhaustive \(MECE\) Principle](#).