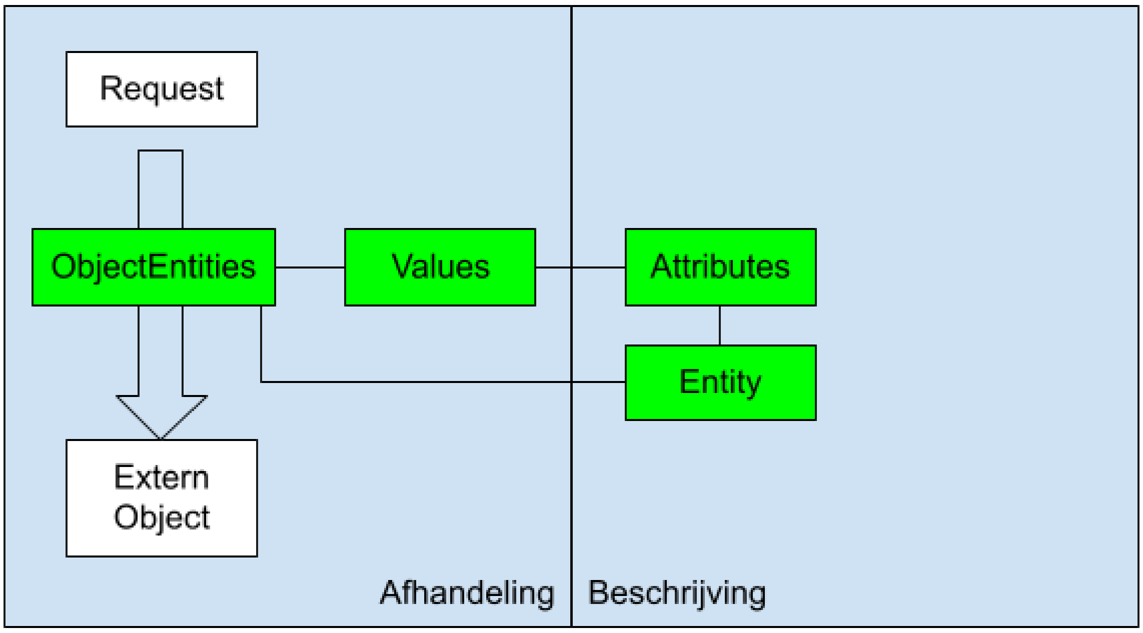
EAV Component



Note that ObjectEntities is named this way because naming an entity 'Object(s)' was not allowed.
Anywhere in this document > you can replace cc / people with the object you want to use as an external object.

Creating an EAV/entity with attributes

In order to save values in the EAV connected to another object in a different component you first need to make sure an Entity in the EAV component exists for it and if it has all the attributes you need to save connected to this object.
(use https://taalhuizen-bisc.commonground.nu/api/v1/eav/#operation/getEntityCollection to check if it already exist, can be done by useing the apifilter type = cc/people)
In this case of an CC/people object it should look like this:

```
"type":"cc/people",
   "name":"person",
   "attributes":[
      {
         "name":"favoriteColor",
         "type":"string",
         "format":"string"
      },
      { etc. }
   ]
```

If it does not exist creating one can be done using this call and a body similar to the one above^:
https://taalhuizen-bisc.commonground.nu/api/v1/eav/#operation/postEntityCollection
Make sure the Entity type is written in plural form, so CC/people not CC/person!

Attribute(s) type and format should (for now) always be the same, one of these:
`"string", "integer", "boolean", "array", "datetime"`

(If you want to store things like an enum, a maximum length of a string or such variables you can set these when creating or updating an Attribute. Not all of the currently available attribute settings work right now, i will keep an up to date list down below of the ones that do work Attribute Settings)
in the case of an enum you could do this:

```
"attributes":[
    {
        "name":"favoriteColor",
        "type":"string",
        "format":"string",
        "enum": [
            "red",
            "blue",
            "green"
        ]
    }
 ]
```

(For an even more complex example of this, see: EAV/entity example)

Attribute Settings

https://taalhuizen-bisc.commonground.nu/api/v1/eav#operation/postAttributeCollection

Here is a list of the attribute settings that are currently checked when an EAV/object with values is created or updated: *(so these are the ones that do something when you would create an EAV/entity with attributes with these settings set)*

- enum (The attribute value must always be one of these values. Works for all attribute types (even booleans, even though that is not very useful 😀).
- defaultValue (The default value for this attribute. Works with string, an integer saved as a string, a boolean saved as a string, a datetime saved as a string (example: `”30-11-2019 20:00:00”`) and an array saved as a string (example:) `"default,value,1;default2,value2;default3:value3,default4:value4"` Makes:

```
[
    [
        "default",
        "value",
        "1"
    ],
    [
        "default2",
        "value2"
    ],
    {
        "default3": "value3",
        "default4": "value4"
    }
]
```

)
- required (The attribute must always have a value. true/false)
- nullable (The attribute can be/have the value: null. true/false)
- minLength (The minimum length for a string type attribute)
- maxLength (The maximum length for a string type attribute)
- minimum (The minimum value for an integer type attribute)
- exclusiveMinimum (Defines if the minimum is exclusive, e.g. a exclusive minimum of 5 would invalidate 5 but validate 6. true/false)

- maximum (The maximum value for an integer type attribute)
- exclusiveMaximum (Defines if the maximum is exclusive, e.g. an exclusive maximum of 5 would invalidate 5 but validate 4. true/false)
- multipleOf (Specifies a number where the value should be a multiple of, e.g. a multiple of 2 would validate 2,4 and 6 but would prevent 5)
- minItems (The minimum length for an array type attribute)
- maxItems (The maximum length for an array type attribute)
- uniqueItems (Define whether or not values in an array should be unique. true/false.
  Works only for non multidimensional arrays with no string keys!!!
  So something **like this**: [“value1”, ”value2”, “etc”]
  And **not**: [“stringkey”:”value1”, ”stringkey2”:”value2”, “etc”:”etc”]
  or: [“value1”,[“value2”,”etc”]])

Example of an EAV/entity with more complex attributes:

! EAV/entity with a type that has **eav**/[name] in it makes it so that EAV/objectEntities created for this EAV/entity will be stored in the EAV <u>not connected to any external object</u> like CC/people !

```
{
    "type":"eav/test",
    "name":"test",
    "attributes":[
        {
            "name":"lievelingsdier",
            "type":"string",
            "format":"string",
            "required":true,
            "minLenght":3,
            "maxLength":10
        },
        {
            "name":"lievelingsnummer",
            "type":"integer",
            "format":"integer",
            "defaultValue":"5",
            "minimum":2,
            "maximum":100,
            "exclusiveMaximum":true,
            "multipleOf":2
        },
        {
            "name":"eenboolean",
            "type":"boolean",
            "format":"boolean",
            "nullable":true
        },
        {
            "name":"eenarray",
            "type":"array",
            "format":"array",
            "enum":[
                [
                    "niet",
                    "vergeten",
                    "dat"
                ],
                [
                    "dit"
                ],
                [
                    {
                        "eigenlijk":"allemaal"
                    },
                    {
                        "in":"het",
                        "engels":"moet"
                    }
                ]
            ],
            "minItems":1
        },
        {
            "name":"eendatetime",
            "type":"datetime",
            "format":"datetime",
            "required":true
        }
    ]
}
```

Using EAV with custom endpoints

Creating an EAV/object connected to an extern object

Then, in order to actually save values connected to an CC/people object you can do the following call:

POST to url: taalhuizen-bisc.commonground.nu/api/v1/eav/object_entities/cc/people

With in the body:

If you have already a CC/people object in the CC component **but not yet** any CC/people object created in the EAV use @self set to the url of the existing CC/people

"@self":
"https://taalhuizen-bisc.commonground.nu/api/v1/cc/people/15b8ccd5-3164-4086-9e1d-75c1049e5e5b",

If you are creating a new CC/people in the CC component **and** want to create a new CC/people object in the EAV for storing values, you do not add the @self to the body.

The last thing to do, is add the values for the body of the normal CC/people object like this:

"givenName":"EAV Test Persoon2",

These will be used to create or update a CC/people object in the CC component.

Then of course add the values for the Entity created in the EAV:

"favoriteColor":"red",

Updating an EAV/object connected to an extern object

(This will/can update the CC/people object **and** the EAV values connected to it!)
Updating a CC/people object in the CC component and the values in the EAV connected to it can be done in almost the same way, ~~there are 2 options for doing this~~:

1. POST to url:
taalhuizen-bisc.commonground.nu/api/v1/eav/object_entities/cc/people/{UUID}
Where {UUID} should be the uuid of the EAV/object_entities object. This object and its id can be found using the following call with apifilter uri = {the CC/people object uri} :
https://taalhuizen-bisc.commonground.nu/api/v1/eav/#operation/getObjectEntityCollection
The body for doing an update is **exactly the same** as the body for creating a new object, (with or without @self).

2. W.I.P
~~2. Or by doing a POST to url:~~
~~taalhuizen-bisc.commonground.nu/api/v1/eav/object_entities/cc/people~~
~~In this case you **need to set** the @self! The body for doing an update is otherwise **exactly the same** as the body for creating a new object.~~

Getting an EAV/object connected to an extern object

(This will return the CC/people object **and** the EAV values connected to it as one object!)
Doing a get on an CC/people object and its EAV values in the EAV component can be done doing one of the following 2 calls:

1. GET to url:
taalhuizen-bisc.commonground.nu/api/v1/eav/object_entities/cc/people/{UUID}
Where {UUID} should be the uuid of the EAV/object_entities object. This object and its id can be found using the following call with apifilter uri = {the CC/people object uri} :
https://taalhuizen-bisc.commonground.nu/api/v1/eav/#operation/getObjectEntityCollection

2. Or by doing a GET to url:
taalhuizen-bisc.commonground.nu/api/v1/eav/object_entities/cc/people
In this case you **need to set** the @self in the body!

Using EAV with object_communication

Creating an EAV/object connected to an extern object

Post to: taalhuizen-bisc.commonground.nu/api/v1/eav/object_communications

```
{
    "componentCode":"cc",
    "entityName":"people",

"@self":"https://taalhuizen-bisc.commonground.nu/api/v1/cc/people/7a73a09f-7dd3-47b4-88
bc-20e4db490014",
    "givenName":"EAV Test Persoon",
    "lievelingskleur":"rood"
}
```

Updating an EAV/object connected to an extern object

(This will/can update the CC/people object **and** the EAV values connected to it!)

Post to: taalhuizen-bisc.commonground.nu/api/v1/eav/object_communications

objectEntityId must be set in order to do an update! @self can also be used:

(For now objectEntityId is required to do an update, even though it can be set to nothing if @self is set.
I will change it so that using the objectEntityId or the @self also works without need of always having objectEntityId set.
Without objectEntityId set, for now, it wil just be a post and create a new objectEntity instead of updating one, i think. Needs testing 😬)

```
{
    "componentCode":"cc",
    "entityName":"people",
    "objectEntityId":"25cfd77e-a29d-481a-9f74-e9a001fdcc61",

"@self":"https://taalhuizen-bisc.commonground.nu/api/v1/cc/people/7a73a09f-7dd3-47b4-88
bc-20e4db490014",
    "givenName":"EAV Test Persoon2",
    "lievelingskleur":"blauw"
}
```

Getting an EAV/object connected to an extern object

(This will return the CC/people object **and** the EAV values connected to it as one object!)

Get to: taalhuizen-bisc.commonground.nu/api/v1/eav/object_communications

objectEntityId or @self is enough, but they can be used together:

```
{
    "componentCode":"cc",
    "entityName":"people",
    "objectEntityId":"25cfd77e-a29d-481a-9f74-e9a001fdcc61",

"@self":"https://taalhuizen-bisc.commonground.nu/api/v1/cc/people/7a73a09f-7dd3-47b4-88
bc-20e4db490014"
}
```

Deleting an EAV/object connected to an extern object

(This will only delete the EAV object and values, **not** the actual CC/people object!)
When deleting an CC/people object that has or might have EAV values in the EAV component, first find and delete the ObjectEntity in the EAV before deleting the CC/people itself. This object can be found using the following call with apifilter uri = {the CC/people object uri} :
https://taalhuizen-bisc.commonground.nu/api/v1/eav/#operation/getObjectEntityCollection
Then delete it with this call:
https://taalhuizen-bisc.commonground.nu/api/v1/eav#operation/deleteObjectEntityItem

Last but very important note!

1. When Creating, updating or getting an CC/people object through the EAV component, you get a response object back that has an id and @id in it that is connected to the EAV/object_entity and not the CC/people object from the CC component. This @id can be used for the update or get calls described above^^^.
The @self in this response however is the @id of the actual CC/people object.

If you want to get the @id of the CC/people object using the id of the EAV objectEntity, you need to do the following call with the id from the response:
https://taalhuizen-bisc.commonground.nu/api/v1/eav#operation/getObjectEntityItem
This ObjectEntity object has a variable 'uri' that has the @id/url of the CC/people object.

2. Note that it is also important to create the EAV/entity with the type written in plural form, So CC/people and not CC/person (or CC/organizations and not CC/organization).
The same goes for every other time i used CC/people in this document

- 3.1 Deleting an EAV/entity will also delete any attribute, objectEntity and value connected.
- 3.2 Deleting an EAV/attribute will also delete any value connected.
- 3.3 Deleting an EAV/objectEntity will also delete any value connected.