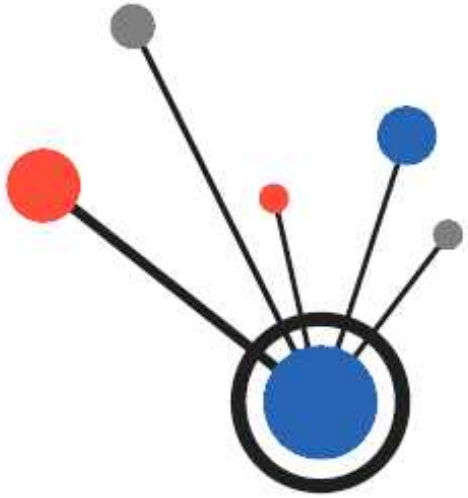


DOCUMENTATION API DOMOSCIO V1.1

Mise à jour le :	15 mai 2015
Nos références :	
Contact commercial :	M. Jean-François Guyomarch
Contact technique :	M. Benoit PRALY

Table des matières

1 First steps

1.1 Use our REST API

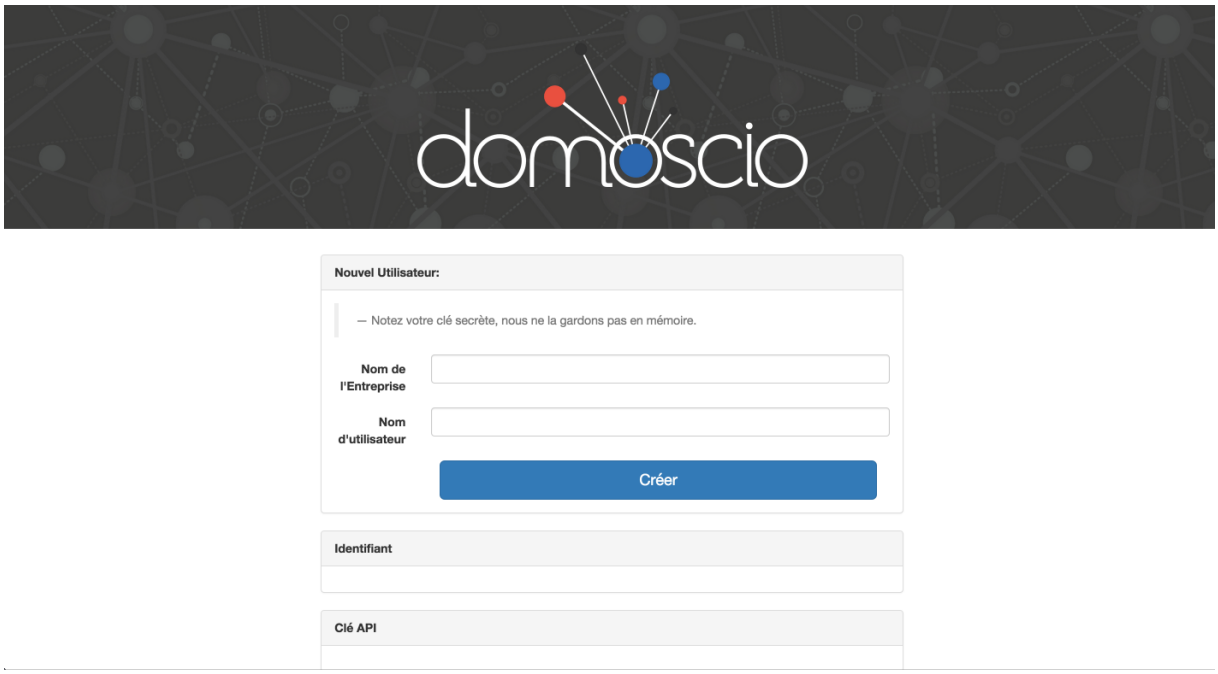
We designed the Domoscio API in a Restful way, so that your user experience is simple and straightforward (Wikipedia). You are able to:

- Submit data requires an **HTTP POST** request
- Retrieve data requires an **HTTP GET** request
- Change data requires an **HTTP PUT** request

Requests must be sent using content-type «application/json». The request and response body encoding is always UTF-8.

You can create a sandbox account to start working with the API. Go and visit us at: http://stats-engine.domoscio.com

1.2 Create Sandbox Account



Once you submitted the form, you are provided with an authentication key and a client ID. Be sure to keep these keys since we are not storing it.

For instance:

00c21bd543b6094c5206f3eaba8b4046

Now that you have your authentication keys, let us see how to properly authenticate your application to our API.

1.3 Authenticate your app

We provide two ways to authenticate and communicate with the service:

- **Basic Access Authentication** is a fast way to implement our API.
- ############ **LTI / OAuth**.

1.3.1 With Authorization header

We use a standard Basic Access Authentication.

The Authorization header is constructed as follows:

```
'Authorization' => "Token token=#{DomoscioRails.configuration.client_passphrase}",
```

1.3.2 With URL parameters

You can also pass your authentication key as a token in the URL to authenticate your application :

URL: `stats-engine.domoscio.com/companies/1/users?token="AZERTY"`

1.4 Create production account

Get in touch with our IT service whenever you feel ready to go live. We will arrange the setup and get back to you with the details.

1.5 Features

Here is the list of all Domoscio API features. We explain in this section how to build URLs to access to our services and we present the main objects and actions available.

1.5.1 Access

/	**OAuth** Basic authentication

1.5.2 Company

The main identifier of your data is your company. You can create a new one as explained previously and get your unique and secured identifier in return.

POST /companies	**Company** Create a new company

From this point, all URLs to access our services are built as such:

[HOST_URL]/companies/[YOUR_COMPANY_ID]/[ACTION]

1.5.3 User

You can add admin users to your company account to help you manage the API.

POST /users	**User** Create a new user
GET /users/{user_Id}	**Fetch a user** Fetch the user corresponding with user_id

1.5.4 Content

The first main object about content of our API is related to graphs and structures. This is the main object that you would have to instantiate. This gives you great liberty to structure your content with the granularity desired.

POST /knowledge_graphs	**Knowledge graph** Create a new knowledge graph
GET /knowledge_graphs/{knowledge_graph_Id}	**Fetch a knowledge graph** Fetch the knowledge graph corresponding

Then you will find the inner graph content objects that are edges and nodes. On the one hand, the edges provide you a way to build your learning path in the knowledge graph.

POST /knowledge_edges	**Knowledge edge** Create a new knowledge edge
GET /knowledge_edgess/{knowledge_graph_Id}	**Fetch a knowledge edge** Fetch the knowledge edge corresponding

On the other hand, the nodes represent the pedagogical unit containers of your knowledge graph.

POST /knowledge_nodes	**Knowledge node** Create a new knowledge node
GET /knowledge_nodes/{knowledge_node_Id}	**Fetch a knowledge node** Fetch the knowledge node corresponding

If you want to go further with the integration of our API, you can declare and link actual content to the structure previously defined with graphs, edges, and nodes.

POST /knowledge_node_contents	**Knowledge node content** Create a new link between knowledge node and content
GET / knowledge_node_contents /{ knowledge_node_content_Id}	**Fetch a knowledge node's content** Fetch the knowledge node's content corresponding
POST /contents	**Content** Create a new content
GET /contents/{content_Id}	**Fetch a content** Fetch the content corresponding

1.5.5 Tags

The main graph is about content. But you can tag this content to give it a meaning. As such you can declare sets of tags that contain tags and tag edges. Once you have declared your graphs of tags, you can easily create a tagging between an object of the knowledge graph and a tag.

POST /tag_sets	**Tag set** Create a new graph of tags

GET /tag_sets/{tag_set_Id}	**Fetch a Tag set** Fetch the set of tags corresponding
POST /tags	**Tag** Create a new tag
GET /tags/{tag_Id}	**Fetch a Tag** Fetch the tag corresponding
POST /taggings	**Tagging** Create a new link between an object of the knowledge graph and a tag
GET /taggings/{tagging_Id}	**Fetch a Tag** Fetch the tagging corresponding

1.5.6 Student

To organize your students, our API provides three layers of aggregation.

First you can gather student into clusters and then groups:

POST /students	**Student** Create a new student
GET /students/{student_Id}	**Fetch a Student** Fetch the student corresponding
POST /student_groups	**Student group** Create a new student
GET /student_groups/{student_group_Id}	**Fetch a Student group** Fetch the student group corresponding
POST /student_clusters	**Student cluster** Create a new student
GET /student_clusters/{student_cluster_Id}	**Fetch a Student cluster** Fetch the student cluster corresponding

You can fetch civil and learning profile for students:

GET /civil_profiles/{civil_profile_Id}	**Fetch a Civil Profile** Fetch the civil profile corresponding
GET /learning_profiles/{learning_profile_Id}	**Fetch a learning profile**

	Fetch the learning profile corresponding

1.5.7 Data

All the data generated by the user during their learning are compiled in this data section.

Once the association between a student and a knowledge node has been defined through our API via:

POST /knowledge_node_students	**Knowledge node student** Create a new link between knowledge node and student
GET /knowledge_node_students /{ knowledge_node_student_Id}	**Fetch a knowledge node's student link** Fetch the knowledge node's student corresponding

You can use this link to store the results generated by this student and gets the API compute his new revision for this knowledge node.

POST /results	**Result** Create a new result
GET /results/{result_Id}	**Fetch a result** Fetch the result corresponding

2 Rules

2.1 Format rules

2.1.1 Dates

All date and time values are displayed as integer numbers and represent the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT), like PHP time() function. The date/time property type in the specification is specified as "Time", the actual JSON type is "Number".

2.1.2 Case Sensitive Parameters

The field and parameter names in requests are case sensitive.

2.1.3 Currencies

We support these currencies

- « EUR »
- « USD »

The ISO_4217 format is expected

2.1.4 Countries

We support these countries

- « FR »
- « EN »
- « ES »

The ISO_3166 format is expected

2.2 Http Response Code

The following HTTP codes are used by the API to respond to requests

- **200**: request successful
- **400**: any kind of logical error (missing parameters, invalid operation, constraint violation etc.)
- **403**: access is forbidden (authentication or authorisation failure)
- **404**: object not found
- **405**: HTTP method not allowed (for example if you try to update a read-only object)
- **411**: Length Required (if the request body is empty for methods that require it)
- **500**: internal server error

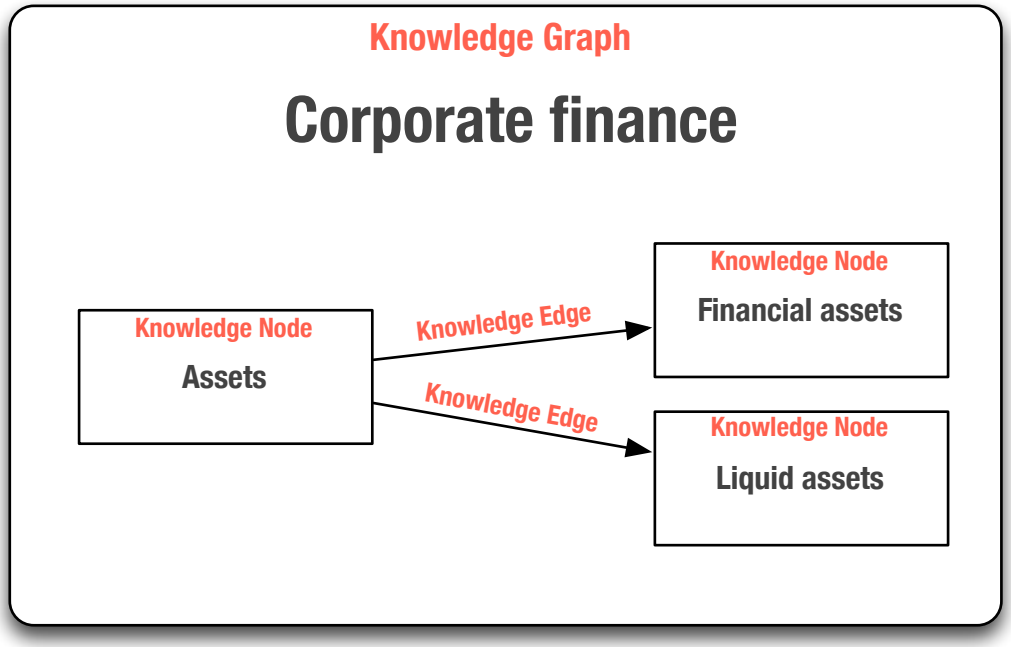
2.3 Error code

TODO

3 Knowledge's Structure

3.1 Knowledge Graph

The content's structure is gathered into knowledge graphs. A knowledge graph is the first object to instantiate while setting up your content on our API.



3.1.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
name	String	Custom data
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

3.1.2 Create (POST)

Method : POST

URL : /knowledge_graphs

3.1.2.1 Parameters (*Required parameters)

Property	Type	Description /expected values
name *	String	Custom data (<255 chars)

3.1.2.2 Send the request (JSON input example)

```
{
        name : « Corporate Finance »
}
```

3.1.2.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « Corporate Finance »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
```

3.1.3 Fetch (GET)

Method : GET

URL : /v1/knowledge_graphs/{knowledge_graph_id}

If no **knowledge_graph_id** is provided it will fetch all the knowledge graphs for your company.

3.1.3.1 Get the response (JSON output example)

```
{
        id : 1,
        name : « Corporate Finance »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
```



```
[{
        id : 1,
        name : « Corporate Finance »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « Mathematics»,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
```


3.1.4 EDIT (PUT)

Method : PUT

URL : /v1/knowledge_graphs/{knowledge_graph_id}

3.1.4.1 Editable Parameters (*Required parameters)

Property	Type	Description /expected values
name	String	Custom data (<255 chars)

3.1.4.2 Send the request (JSON input example)

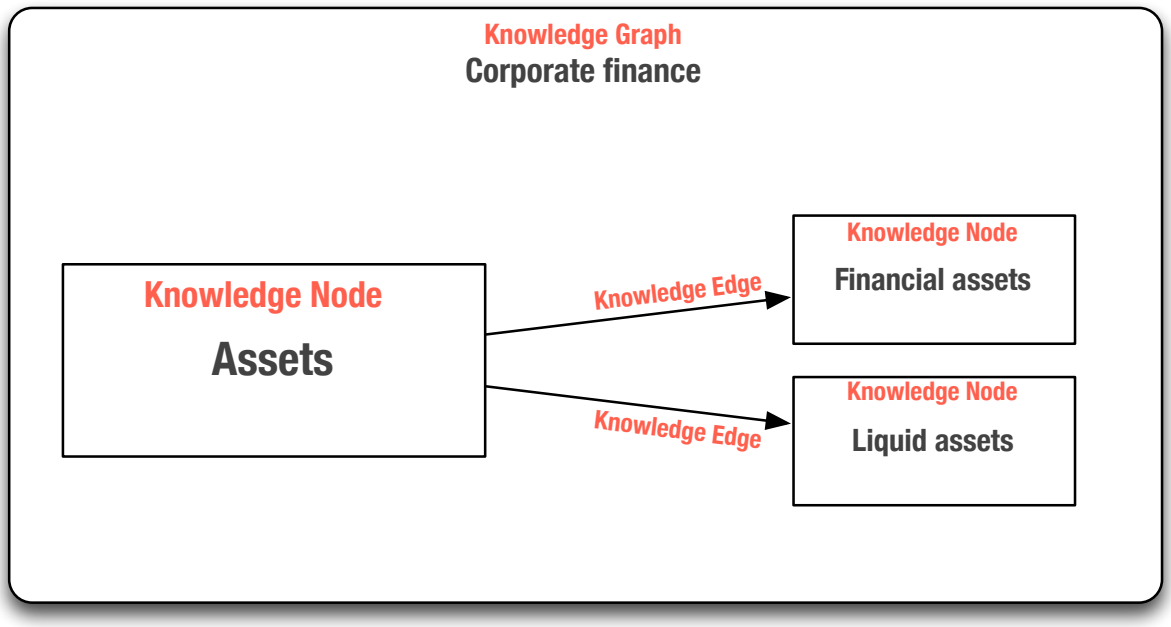
```
{
        name : « No more Corporate Finance »
}
```

3.1.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more Corporate Finance »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 11:21:03
```

3.2 Knowledge Node

The content's structure is scattered into knowledge nodes. Knowledge nodes are the second objects to instantiate while setting up your content on our API.



3.2.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
knowledge_graph_id	Integer	The knowledge graph this knowledge node belongs to
name	String	Custom data
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

3.2.2 Create (POST)

Method : POST

URL : /v1/knowledge_nodes

*3.2.2.1 Parameters (*Required parameters)*

Property	Type	Description /expected values
knowledge_graph_id *	Integer	The knowledge graph this knowledge node belongs to
name *	String	Custom data (<255 chars)

3.2.2.2 Send the request (JSON input example)

```
{
        knowledge_graph_id : 1,
        name : « Financial asset »
}
```

3.2.2.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « Financial asset »,
        knowledge_graph_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

3.2.3 Fetch (GET)

Method : GET

URL : /v1/knowledge_nodes/{knowledge_node_id}

If no **knowledge_node_id** is provided it will fetch all the knowledge nodes for your company.

3.2.3.1 Get the response (JSON output example)

```
{
        id : 1,
        name : « Financial asset »,
        knowledge_graph_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        name : « Financial asset »,
        knowledge_graph_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « Liquid asset »,
        knowledge_graph_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```


3.2.4 EDIT (PUT)

Method : PUT

URL : /v1/knowledge_nodes/{knowledge_node_id}

3.2.4.1 Editable Parameters (*Required parameters)

Property	Type	Description /expected values
name	String	Custom data (<255 chars)

3.2.4.2 Send the request (JSON input example)

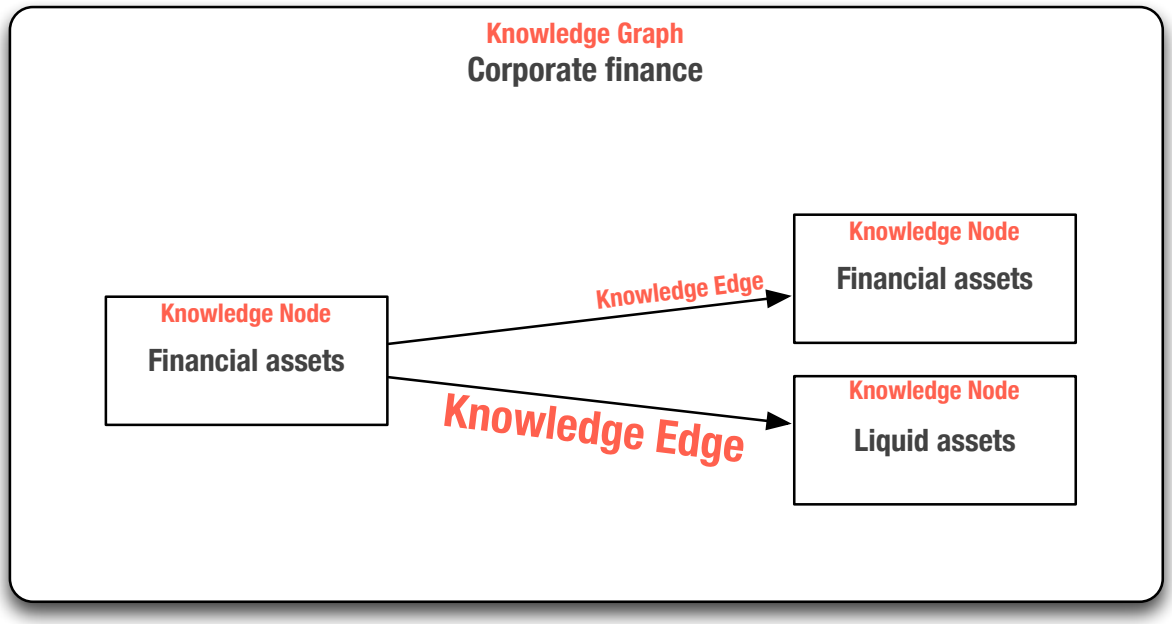
```
{
        name : « No more Financial asset »
}
```

3.2.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more Financial asset »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 11:21:03
}
```

3.3 Knowledge Edge

The content's structure is symbolised by knowledge edges. Knowledge edges are the last objects to instanciate while setting up your content on our API. It gives you a simple way to build your learning path.



3.3.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge edge belongs to
knowledge_graph_id	Integer	The knowledge graph the knowledge edge belongs to
source_node_id	Integer	The source node of the knowledge edge
destination_node_id	Integer	The destination node of the knowledge edge
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

3.3.2 Create (POST)

Method : POST

URL : /v1/knowledge_edges

3.3.2.1 Parameters (*Required parameters)

Property	Type	Description
knowledge_graph_id *	Integer	The knowledge graph the knowledge edge belongs to
source_node_id *	Integer	The source node of the knowledge edge
destination_node_id *	Integer	The destination node of the knowledge edge

3.3.2.2 Send the request (JSON input example)

```
{
        knowledge_graph_id : « Corporate Finance »,
        source_node_id : 1,
        destination_node_id : 2
}
```

3.3.2.3 Get the response (JSON output example)

```
{
        id : 1,
        knowledge_graph_id : 1,
        source_node_id : 1,
        destination_node_id : 2,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```


3.3.3 Fetch (GET)

Method : GET

URL : /v1/knowledge_edges/{knowledge_edge_id}

If no **knowledge_edge_id** is provided it will fetch all the knowledge edges for your company.

3.3.3.1 Get the response (JSON output example)

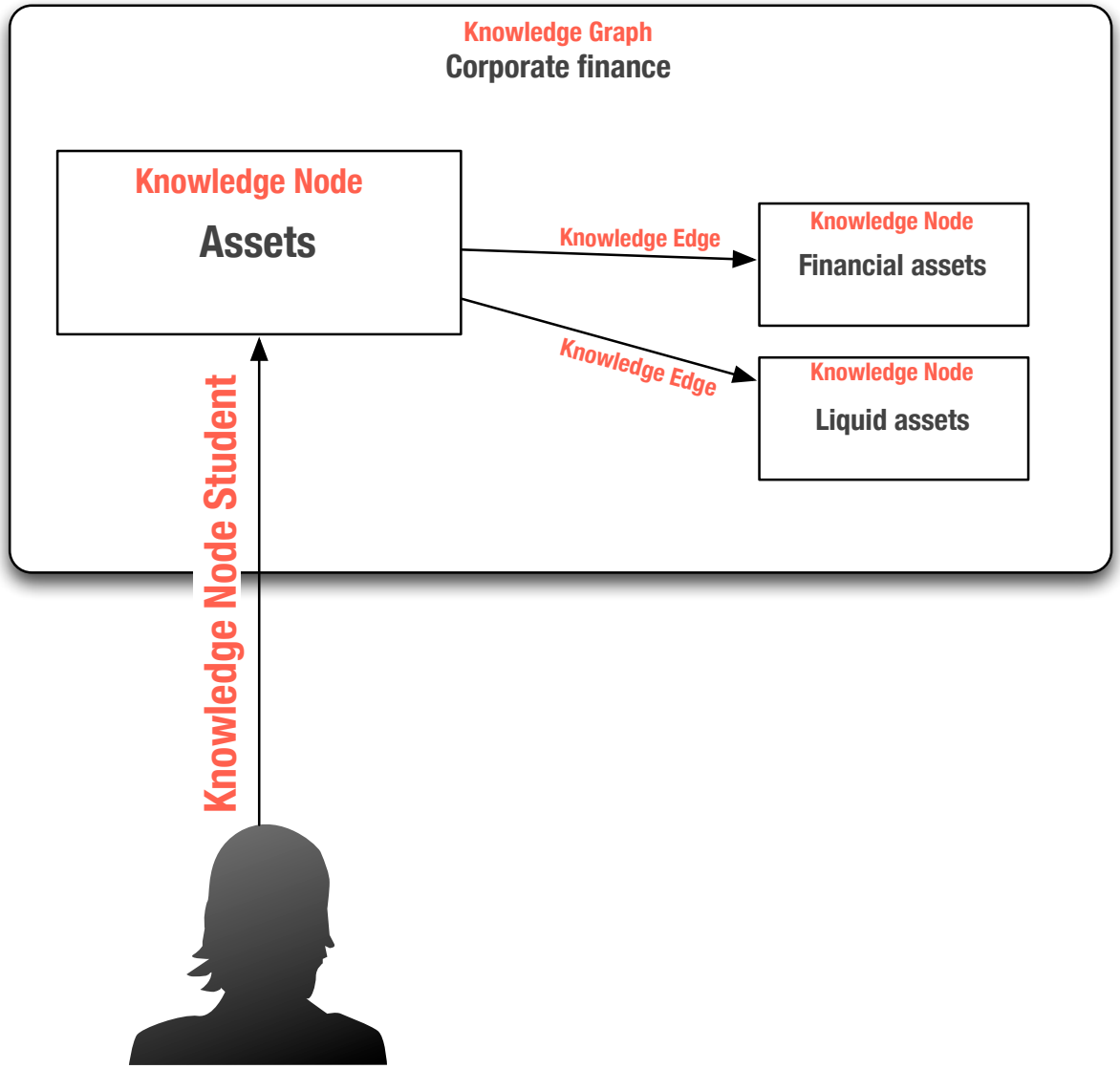

```
{
        id : 1,
        knowledge_graph_id : 1,
        source_node_id : 1,
        destination_node_id : 2,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        knowledge_graph_id : 1,
        source_node_id : 1,
        destination_node_id : 2,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        knowledge_graph_id : 1,
        source_node_id : 1,
        destination_node_id : 2,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03

}]
```


3.4 Knowledge Node Student

Once the content's structure is declared on our API, you can create logical links between students and knowledge nodes. We could call that « subscriptions ». But it goes further than that since this link is also the container of all the results, data computations and analysis for the student.



3.4.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge node student belongs to
knowledge_node_id	Integer	The knowledge node the knowledge node student belongs to
student_id	Integer	The student the knowledge node student belongs_to

state	Integer	The destination node of the knowledge edge
history	String	Sequence of binary results for the student on the knowledge node
next_review_at	Timestamp	The next revision date computed by the engine
current_review_interv	Float	The current revision interval
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

3.4.2 Create (POST)

Method : POST

URL : /v1/knowledge_node_students

*3.4.2.1 Parameters (*Required parameters)*

Property	Type	Description
knowledge_node_id *	Integer	The knowledge node the knowledge node student belongs to
student_id *	Integer	The student the knowledge node student belongs_to

3.4.2.2 Send the request (JSON input example)

```
{
        knowledge_node_id : 1,
        student_id: 1
}
```

3.4.2.3 Get the response (JSON output example)

```
{
        id : 1,
        knowledge_node_id : 1,
        student_id : 1,
        company_id : 1,
        state : 0,
        history : '',
        next_review_at : null,
        current_review_interv : 0,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

3.4.3 Fetch (GET)

Method : GET

URL : /v1/knowledge_node_students/{knowledge_node_student_id}

If no **knowledge_node_student_id** is provided it will fetch all the knowledge node students for your company.

3.4.3.1 Get the response (JSON output example)

```
{
        id : 1,
        knowledge_node_id : 1,
        student_id : 1,
        company_id : 1,
        state : 0,
        history : ‘’,
        next_review_at : null,
        current_review_interv : 0,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        knowledge_node_id : 1,
        student_id : 1,
        company_id : 1,
        state : 0,
        history : '',
        next_review_at : null,
        current_review_interv : 0,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        knowledge_node_id : 1,
        student_id : 1,
        company_id : 1,
        state : 1,
        history : '1',
        next_review_at : 2014-09-21 09:21:03,
        current_review_interv : 3.5,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03

}]
```

4 Knowledge's Content

4.1 Content

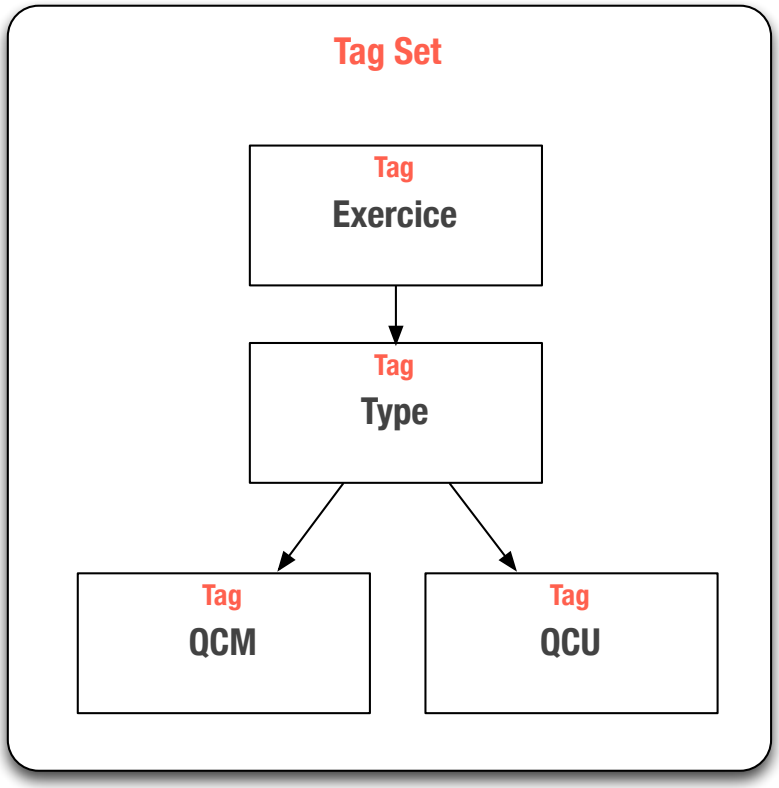
4.2 Knowledge Node Content

5 Tag System

In order to give meaning to your knowledge graph, you can create as many set tags that you need. By default, we have 2 sets of tags to categorize the edges and the nodes. The tag system will come in handy when we will deal with the learning rules.

5.1 Set of tags

In parallel of creating your knowledge's structure, you can create any set of tags that you want.



5.1.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
name	String	Custom data

created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

5.1.2 Create (POST)

Method : POST

URL : /tag_sets

*5.1.2.1 Parameters (*Required parameters)*

Property	Type	Description /expected values
name *	String	Custom data (<255 chars)

5.1.2.2 Send the request (JSON input example)

```
{
        name : « Typology »
}
```

5.1.2.3 Get the response (JSON output example)

{
id : 1,
name : « Typology »,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

5.1.3 Fetch (GET)

Method : GET

URL : /v1/tag_sets/{tag_set_id}

If no **tag_set_id** is provided it will fetch all tag sets for your company.

5.1.3.1 Get the response (JSON output example)

{
id : 1,
name : « Typology »,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

```
[{
        id : 1,
        name : « Typology »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « Skills»,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```


5.1.4 EDIT (PUT)

Method : PUT

URL : /v1/tag_sets/{tag_set_id}

*5.1.4.1 Editable Parameters (*Required parameters)*

Property	Type	Description /expected values
name	String	Custom data (<255 chars)

5.1.4.2 Send the request (JSON input example)

```
{
        name : « No more typology »
}
```

5.1.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more typology »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 11:21:03
}
```

5.2 Tags

Once you have create your first set of tags, you can start create the actual Tags.

5.2.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
name	String	Custom data
tag_set_id	Integer	The tag set this tag belongs to
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

5.2.2 Create (POST)

Method : POST

URL : /tags

5.2.2.1 Parameters (*Required parameters)

Property	Type	Description /expected values
name *	String	Custom data (<255 chars)
tag_set_id *	Integer	The tag set this tag belongs to

5.2.2.2 Send the request (JSON input example)

```
{
        name : « QCM»,
        tag_set_id : 1
}
```

5.2.2.3 Get the response (JSON output example)

{
id : 1,
name : « QCM»,
tag_set_id : 1,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

5.2.3 Fetch (GET)

Method : GET

URL : /v1/tags/{tag_id}

If no **tag_id** is provided it will fetch all tag sets for your company.

5.2.3.1 Get the response (JSON output example)

{
id : 1,
name : « QCM »,
tag_set_id : 1,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

```
[{
        id : 1,
        name : « QCM»,
        company_id : 1,
        tag_set_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « QCU»,
        company_id : 1,
        tag_set_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 3,
        name : « Type»,
        company_id : 1,
        tag_set_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```


5.2.4 EDIT (PUT)

Method : PUT

URL : /v1/tags/{tag_id}

5.2.4.1 Editable Parameters (*Required parameters)

Property	Type	Description /expected values
name	String	Custom data (<255 chars)

Please note that you cannot change the set_tag_id of a tag because it will henceforth break the structure of all the tag set.

5.2.4.2 Send the request (JSON input example)

```
{
        name : « No more QCM»
}
```

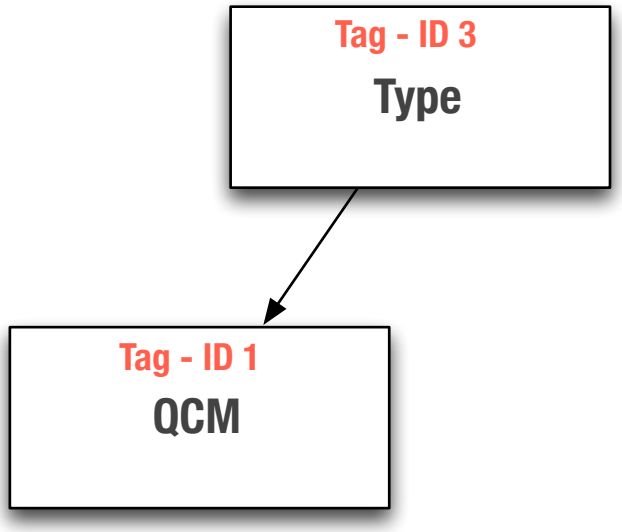
5.2.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more QCM»,
        company_id : 1,
        tag_set_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 11:21:03
}
```

5.3 Link between tags

To create a link between two tags because one includes the other, you can create a tag edge from one to the other. As such you are creating a DAG (directed acyclic graph) of tags. For performance purposes, when you create an edge we will create a bunch of indirect edges to fully describe the graph.

For instance we will work with these two tags:



5.3.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
ancestor_id	Integer	The target tag of the current edge
descendant_id	Integer	The source tag of the current edge
direct	Boolean	If the edge is direct or indirect
count	Integer	The number of different ways between the ancestor and descendant tags
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

5.3.2 Create (POST)

Method : POST

URL : /tag_edges

5.3.2.1 Parameters (*Required parameters)

Property	Type	Description /expected values
ancestor_id *	Integer	The target tag of the current edge
descendant_id *	Integer	The source tag of the current edge

5.3.2.2 Send the request (JSON input example)

```
{
        ancestor_id : 3,
        descendant_id : 1
}
```

5.3.2.3 Get the response (JSON output example)

```
{
        id : 1,
        ancestor_id : 3,
        descendant_id : 1,
        direct : true,
        count : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

5.3.3 Fetch (GET)

Method : GET

URL : /v1/tag_edges/{tag_edge_id}

If no **tag_edge_id** is provided it will fetch all tag sets for your company. Be aware that for each edge you created, we also created a lot of indirect edges to build the graph. Hence, you will have fetch your “direct” edges but also the “indirect” edges.

5.3.3.1 Get the response (JSON output example)

```
{
        id : 1,
        ancestor_id : 3,
        descendant_id : 1,
        direct : true,
        count : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

5.4 Tagging the knowledge’s structure

At this point, your tags are structured and set. You will want to start tagging your content.

5.4.1 Object resources

Property	Type	Description
company_id	Integer	The company the knowledge graph belongs to
tag_id	String	The tag this tagging belongs to

taggable_id	Integer	The taggable object ID this tagging belongs to
taggable_type	String	The taggable object type this tagging belongs to
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

5.4.2 Create (POST)

Method : POST

URL : /taggings

*5.4.2.1 Parameters (*Required parameters)*

Property	Type	Description /expected values
tag_id *	Integer	The tag this tagging belongs to
taggable_id *	Integer	The taggable object ID this tag belongs to
taggable_type *	Integer	The taggable object type this tag belongs to

5.4.2.2 Send the request (JSON input example)

```
{
        tag_id : 1,
        taggable_id : 2,
        taggable_type : « KnowledgeNode »
}
```


5.4.2.3 Get the response (JSON output example)


```
{
        id : 1,
        tag_id : 1,
        taggable_id : 2,
        taggable_type : « KnowledgeNode »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```


5.4.3 Fetch (GET)

Method : GET

URL : /v1/taggings/{tagging_id}

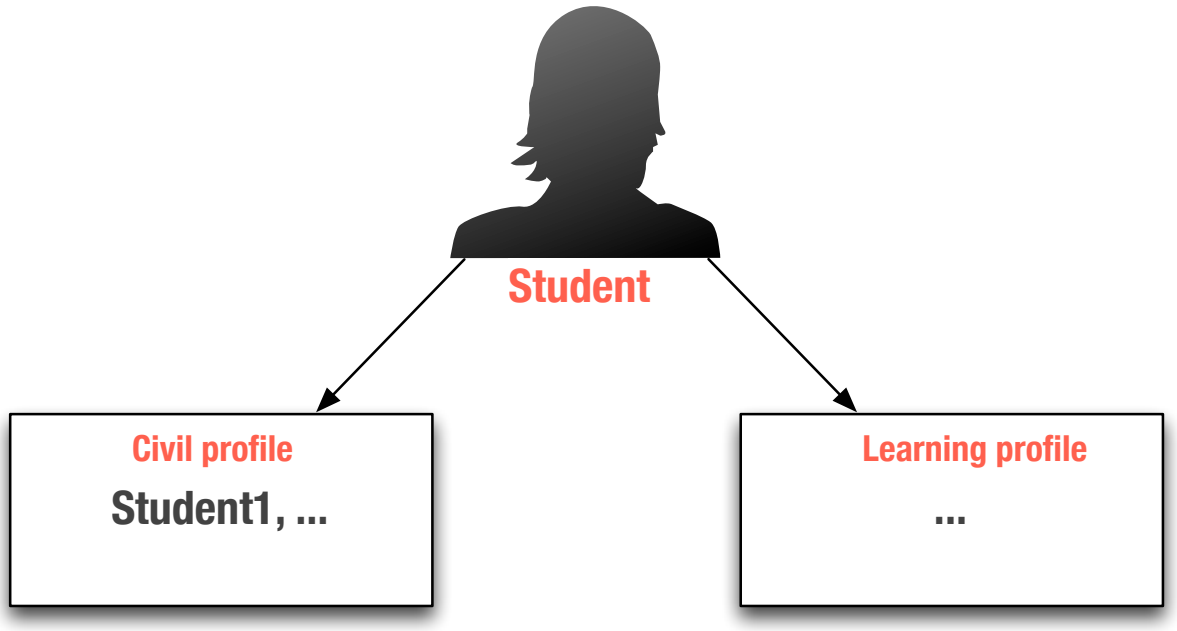
If no **tagging_id** is provided it will fetch all tag sets for your company.

5.4.3.1 Get the response (JSON output example)

```
{
        id : 1,
        tag_id : 1,
        taggable_id : 2,
        taggable_type : « KnowledgeNode »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```


6 Student

As a new student is created on your application, you would have to declare a new student on our API as well.



6.1.1 Object resources

Property	Type	Description

company_id		Integer	The company the knowledge graph belongs to
student_group_id		Integer	The group this student belongs_to
created_at		Timestamp	The creation date of the object
updated_at		Timestamp	The last update date of the object
civil_profile_attributes			
	name	String	Custom data
	sexe	String	Amongst (‘male’, ’female’, ‘undefined’)
	day_of_birth	Timestamp	The date of birth of the student
	place_of_birth	String	The place of birth of the student
	country_of_residence	String	The coutry of residence of the student
	city_of_residence	String	The city of residence of the student
learning_profile_attributes			
	forgetting_parameters	String	

6.1.2 Create (POST)

Method : POST

URL : /v1/students

6.1.2.1 Parameters (*Required parameters)

Property		Type	Description
student_group_id		Integer	The group this student belongs_to
civil_profile_attributes			
	name	String	Custom data
	sexe	String	Amongst ('male', 'female', 'undefined')
	day_of_birth	Timestamp	The date of birth of the student
	place_of_birth	String	The place of birth of the student
	country_of_residence	String	The coutry of residence of the student
	city_of_residence	String	The city of residence of the student
learning_profile_attributes			
	forgetting_parameters	String	

As a convenience we let you declare in the same time a student, his civil profile and his learning profile:

6.1.2.2 Send the request (JSON input example)

```
{
        student_group_id: 1,
        civil_profile_attributes: {
                name: "Student1",
                sexe: "male",
                day_of_birth: Date.today, place_of_birth:
                "FR",
                country_of_residence: "FR",
                city_of_residence: "Paris"
        },
        learning_profile_attributes: {
                forgetting_parmeters: "[1,2,3,4]"
        }
}
```

6.1.2.3 Get the response (JSON output example)

```
{
        id : 2,
        student_group_id: 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03 ,
        civil_profile_attributes: {
                name: "Student1",
                sexe: "male",
                day_of_birth: Date.today, place_of_birth:
                "FR",
                country_of_residence: "FR",
                city_of_residence: "Paris"
        },
        learning_profile_attributes: {
                forgetting_parmeters: "[1,2,3,4]"
        }
}
```


6.1.3 Fetch (GET)

Method : GET

URL : /v1/students/{student_id}

If no **student_id** is provided it will fetch all the knowledge graphs for your company.

6.1.3.1 Get the response (JSON output example)

```
{
        id : 2,
        student_group_id: 1,
        name: "Student1",
        sexe: "male",
        day_of_birth: Date.today,
        place_of_birth: "FR",
        country_of_residence: "FR",
        city_of_residence: "Paris",
        forgetting_parmeters: "[1,2,3,4]",
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        student_group_id: 1,
        name: null,
        sexe: null,
        day_of_birth: null,
        place_of_birth: null,
        country_of_residence: null,
        city_of_residence: null,
        forgetting_parmeters: null,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        student_group_id: 1,
        name: "Student1",
        sexe: "male",
        day_of_birth: Date.today,
        place_of_birth: "FR",
        country_of_residence: "FR",
        city_of_residence: "Paris",
        forgetting_parmeters: "[1,2,3,4]",
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```


6.1.4 EDIT (PUT)

Method : PUT

URL : /v1/students/{student_id}

6.1.4.1 Editable Parameters (*Required parameters)

Property		Type	Description
student_group_id		Integer	The group this student belongs_to
civil_profile_attributes			
	name	String	Custom data
	sexe	String	Amongst ('male', 'female', 'undefined')
	day_of_birth	Timestamp	The date of birth of the student
	place_of_birth	String	The place of birth of the student
	country_of_residence	String	The country of residence of the student
	city_of_residence	String	The city of residence of the student
learning_profile_attributes			
	forgetting_parameters	String	

6.1.4.2 Send the request (JSON input example)

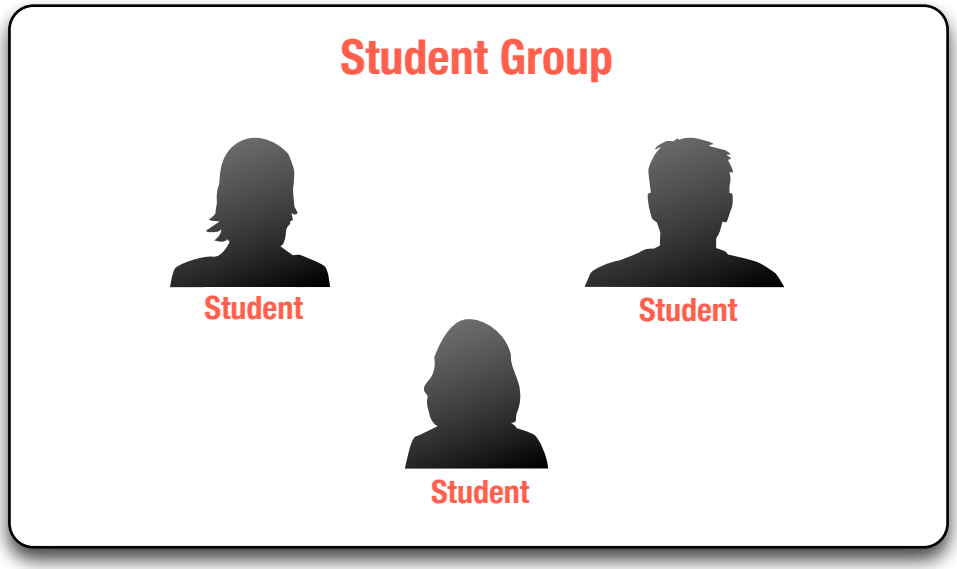
```
{
        student_group_id: 1,
        civil_profile_attributes: {
                name: "Student with new name",
                sexe: "female",
        },
        learning_profile_attributes: {
                forgetting_parmeters: "[1,2,3,4]"
        }
}
```

6.1.4.3 Get the response (JSON output example)

```
{
        id : 2,
        student_group_id: 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03 ,
        civil_profile_attributes: {
                name: "Student with new name",
                sexe: "female",
                day_of_birth: Date.today, place_of_birth:
                "FR",
                country_of_residence: "FR",
                city_of_residence: "Paris"
        },
        learning_profile_attributes: {
                forgetting_parmeters: "[1,2,3,4]"
        }
}
```

6.2 Student Group

A good way to manage your students is to gather them into groups on our API. Il will provide you some convenient method to get aggregated results on a group.



6.2.1 Object resources

Property	Type	Description
company_id	Integer	The company this student group belongs to
name	String	Custom data
student_cluster_id	Integer	The student cluster this student group belongs to
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

6.2.2 Create (POST)

Method : POST

URL : /v1/student_groups

*6.2.2.1 Parameters (*Required parameters)*

Property	Type	Description /expected values
name *	String	Custom data (<255 chars)
student_cluster_id	Integer	The student cluster this student group belongs_to

6.2.2.2 Send the request (JSON input example)

```
{
        name : « 2nd year associates financial services »,
        student_cluster_id : 1
`
```

6.2.2.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « 2nd year associates financial services »,
        student_cluster_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

6.2.3 Fetch (GET)

Method : GET

URL : /v1/student_groups/{student_group_id}

If no **student_group_id** is provided it will fetch all the student groups for your company.

6.2.3.1 Get the response (JSON output example)

```
{
        id : 1,
        name : « 2nd year associates financial services »,
        student_cluster_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        name : « 2nd year associates financial services »,
        student_cluster_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « senior associates financial services »,
        student_cluster_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```

6.2.4 EDIT (PUT)

Method : PUT

URL : /v1/student_groups/{student_group_id}

6.2.4.1 Editable Parameters (*Required parameters)

Property	Type	Description /expected values
name	String	Custom data (<255 chars)
student_cluster_id	Integer	The student cluster this student group belongs to

6.2.4.2 Send the request (JSON input example)

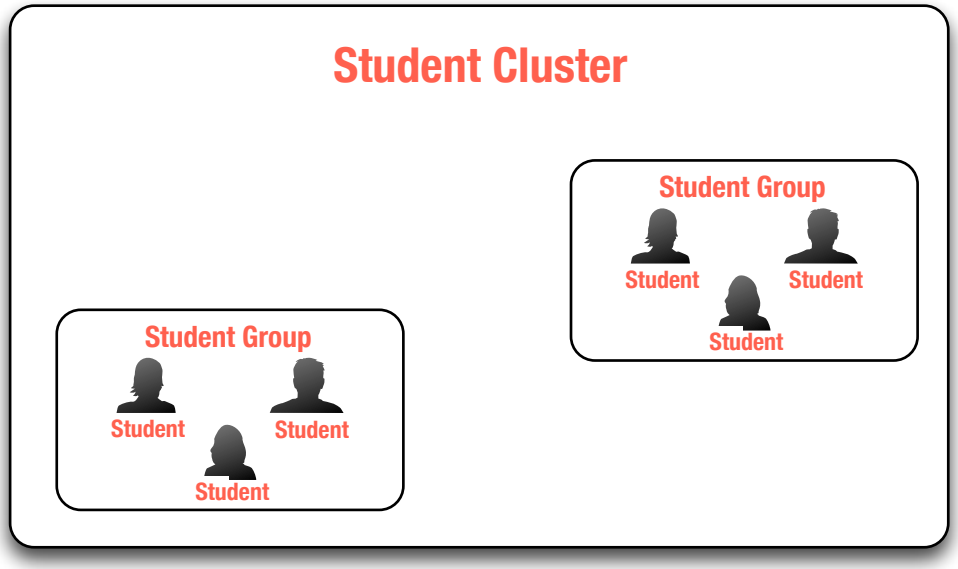
```
{
        name : « No more 2nd year associates financial services »
}
```

6.2.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more 2nd year associates financial services »,
        student_cluster_id : 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

6.3 Student Cluster

If you need some higher level of student's management you can use clusters. Clusters will let you organize at high level your organization.



6.3.1 Object resources

Property	Type	Description
company_id	Integer	The company the student cluster graph belongs to
name	String	Custom data
created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

6.3.2 Create (POST)

Method : POST

URL : /v1/student_clusters

6.3.2.1 Parameters (*Required parameters)

Property	Type	Description /expected values
name *	String	Custom data (<255 chars)

6.3.2.2 Send the request (JSON input example)

```
{
        name : « Financial services »
}
```

6.3.2.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « Financial services »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

6.3.3 Fetch (GET)

Method : GET

URL : /v1/student_clusters/{student_cluster_id}

If no **student_cluster_id** is provided it will fetch all the student clusters for your company.

6.3.3.1 Get the response (JSON output example)


```
{
        id : 1,
        name : « Financial services »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

```
[{
        id : 1,
        name : « Financial services »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        name : « HR services »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
```


6.3.4 EDIT (PUT)

Method : PUT

URL : /v1/student_clusters/{student_cluster_id}

6.3.4.1 Editable Parameters (*Required parameters)

Property	Type	Description /expected values

name	String	Custom data (<255 chars)

6.3.4.2 Send the request (JSON input example)

```
{
        name : « No more Financial services »
}
```

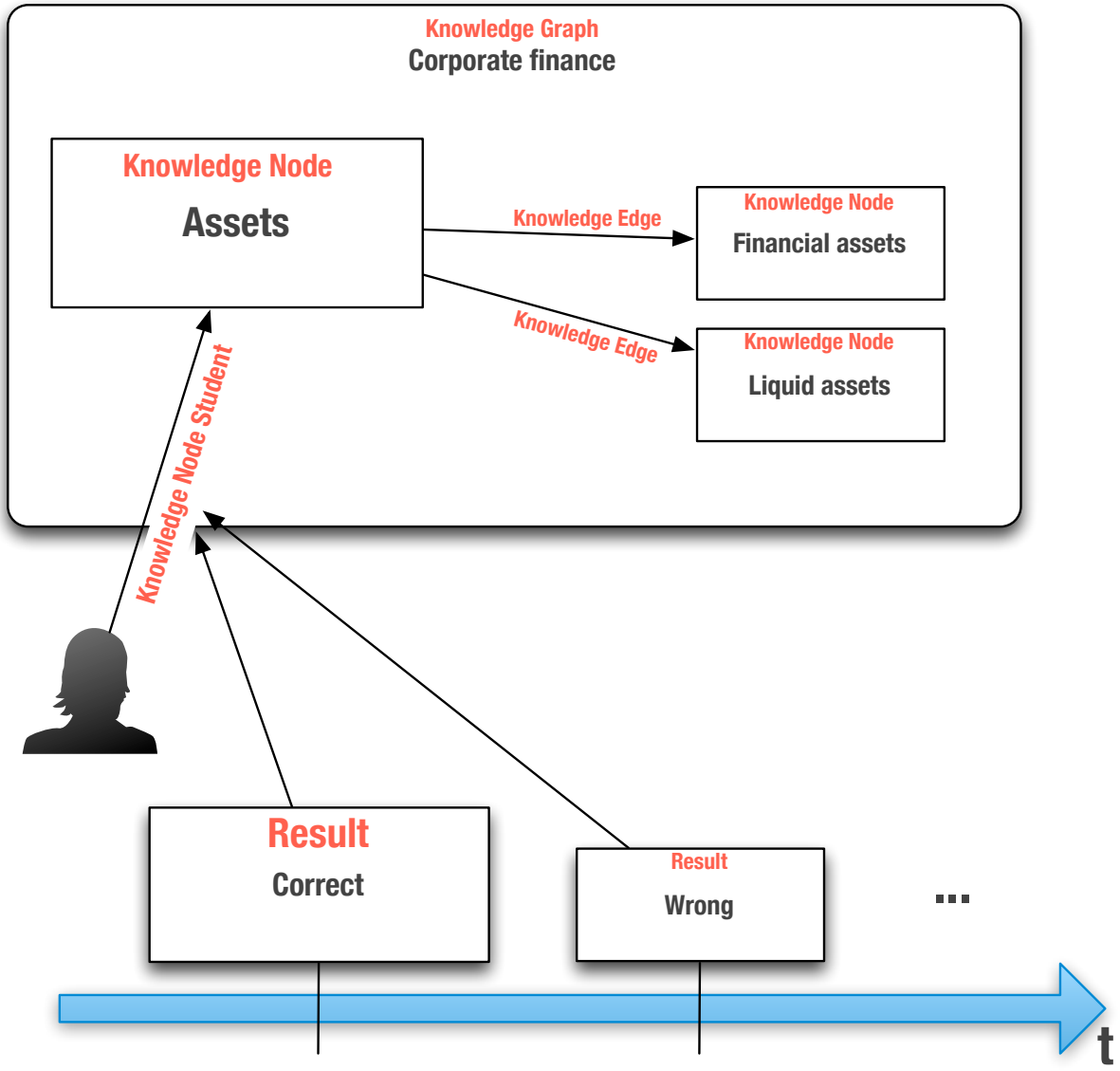
6.3.4.3 Get the response (JSON output example)

```
{
        id : 1,
        name : « No more Financial services »,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}
```

7 Data

7.1 Result

Once all the management layers and knowledge's structure are declared, you can start registering all your student's data and retrieve their statistics on our API.



7.1.1 Object resources

Property	Type	Description
company_id	Integer	The company this result belongs to
knowledge_node_student_id	Integer	The knowledge cell/student pair this result belongs to
value	Integer	The result of the student. Either 0 or 1.

created_at	Timestamp	The creation date of the object
updated_at	Timestamp	The last update date of the object

7.1.2 Create (POST)

Method : POST

URL : /v1/results

*7.1.2.1 Parameters (*Required parameters)*

Property	Type	Description /expected values
knowledge_node_student_id*	Integer	The knowledge cell/student pair this result belongs to
value *	Integer	Either a 0 or a 1.

7.1.2.2 Send the request (JSON input example)

```
{
        knowledge_node_student_id : 1,
        value: 1
}
```

7.1.2.3 Get the response (JSON output example)

{
id : 1,
knowledge_node_student_id : 1,
value: 1,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

7.1.3 Fetch (GET)

Method : GET

URL : /v1/results/{result_id}

If no **result_id** is provided it will fetch all the student clusters for your company.

7.1.3.1 Get the response (JSON output example)

{
id : 1,
knowledge_node_student_id : 1,
value: 1,
company_id : 1,
created_at : 2014-09-17 09:21:03,
updated_at : 2014-09-17 09:21:03
}

```
[{
        id : 1,
        knowledge_node_student_id : 1,
        value: 1,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
},
{
        id : 2,
        knowledge_node_student_id : 1,
        value: 0,
        company_id : 1,
        created_at : 2014-09-17 09:21:03,
        updated_at : 2014-09-17 09:21:03
}]
```


8 Statistics

9 Flow diagram

10 Appendix