MORa Documentation

Release 0.2.0

Magenta ApS

Nov 06, 2018

Contents

1 Om MORa **3**
- 1.1 Introduktion . . . 3
- 1.2 Opbygning . . . 4
- 1.3 Opsætning af udviklingsmiljø . . . 5
- 1.4 Testsuiten . . . 7
- 1.5 Installering . . . 8
- 1.6 Dokumentation . . . 8

2 Manual testing of MO **11**
- 2.1 Testing the basic features . . . 11
- 2.2 Testing temporality . . . 12

3 API dokumentation **13**
- 3.1 Address . . . 13
- 3.2 Legacy service layer . . . 14
- 3.3 Authentication . . . 17
- 3.4 CPR . . . 18
- 3.5 Details . . . 18
- 3.6 Employees . . . 22
- 3.7 Facets . . . 35
- 3.8 IT Systems . . . 37
- 3.9 Organisation . . . 37
- 3.10 Organisational units . . . 39
- 3.11 Server-side codebase . . . 44
- 3.12 Proposal for new REST API . . . 80

4 Indices and tables **89**

Python Module Index **91**

HTTP Routing Table **93**

Dette er den fuldstændige dokumentation til MORa — MedarbejderOrganisation + LoRa. Indholdet er opdelt i følgende sektioner:

CHAPTER 1

Om MORa

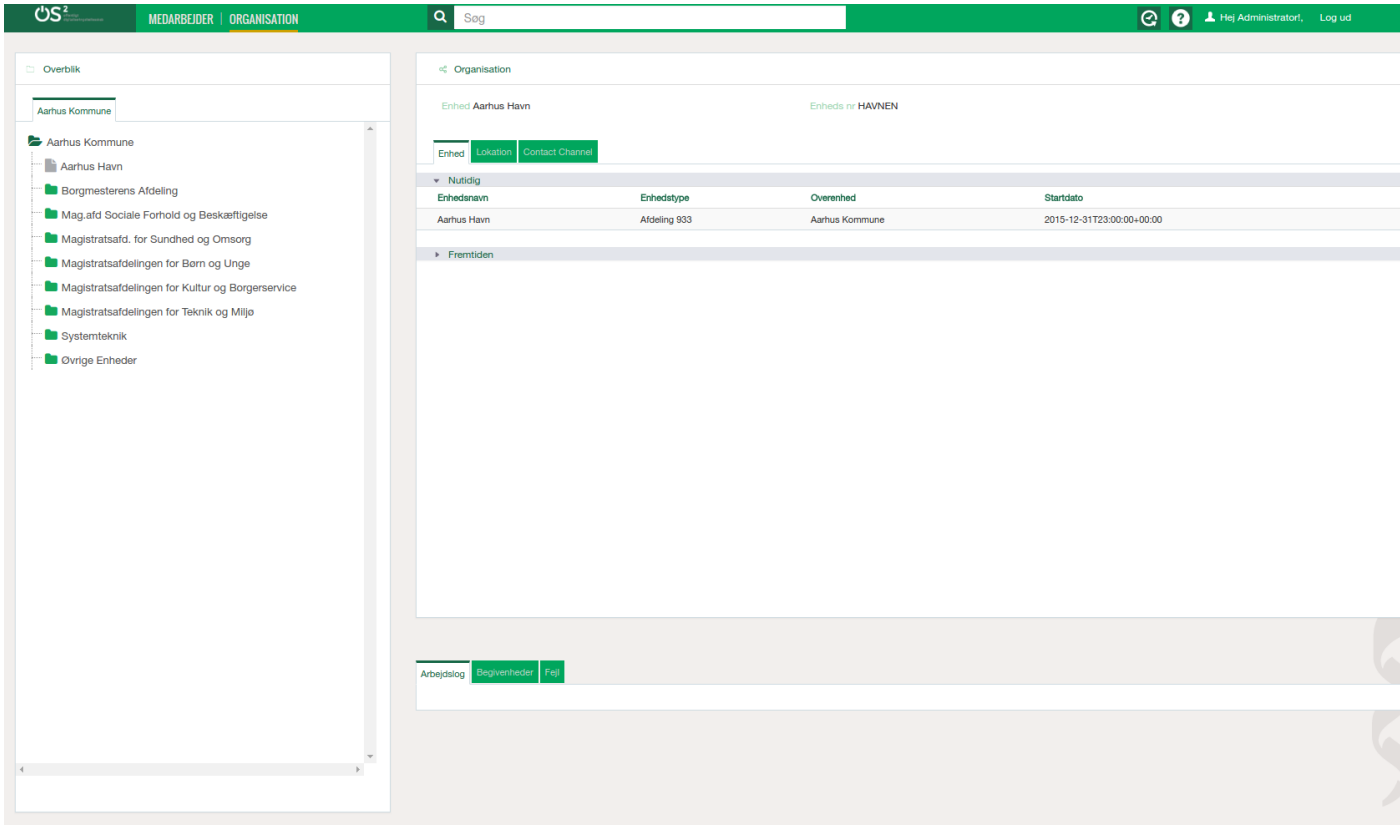
Indhold

- *Om MORa*
  - *Introduktion*
  - *Opbygning*
  - *Opsætning af udviklingsmiljø*
  - *Testsuiten*
  - *Installering*
  - *Dokumentation*

1.1 Introduktion

MORa er en webapplikation til håndtering af et medarbejder- og organisationshierarki. Systemet sætter brugerne i stand til at navigere rundt i eksempelvis organisationshierarkiet, indhente relevante informationer om de forskellige organisationsenheder samt at redigere de data, der er tilknyttet de forskellige enheder.

Navnet MORa er en sammentrækning af MO og LoRa og refererer til hhv. OS2MO og den Lokale Rammearkitektur. Nedenstående figur viser et typisk eksempel på en side i systemet brugerflade:



1.2 Opbygning

Den modulære opbygning af MORa ses på nedenstående figur.

MORa består af frontend og en middleend og sidstnævnte kommunikerer med en LoRa backend. De enkelte moduler kan opfattes som elementer i MVC-modellen:

1.2.1 MO (Frontend / View)

MOs frontend er skrevet i Javascript frameworket AngularJS. Frontenden kan opfattes som *View* i MVC-modellen, og brugerne interagerer med applikationen via denne. Frontenden kommunikerer indirekte med Lora via MOs middleend.

1.2.2 LoRa (Backend / Model)

En LoRa backend, som gemmer alle data i en PostgreSQL-database. Disse data udstilles og manipuleres via en RESTful service skrevet i Python. LoRa kan opfattes som *Model* i MVC-modellen.

1.2.3 MO (Middleend / Control)

MOs middleend fungerer som en bro mellem frontenden og backenden, og den har til opgave at oversætte de data, der sendes mellem frontenden og backenden til passende JSON formater, når der udføres læse- og skriveoperationer fra og til LoRa (se flere detaljer nedenfor).

Når der læses fra LoRa, leverer denne data i et JSON-format, som frontenden ikke umiddelbart kan tolke, hvorfor middleenden oversætter disse til det JSON-format, som frontenden forventer. Tilsvarende sender frontenden ved

skriveoperationer JSON i et format, som skal oversættes af middleenden til det JSON-format, som kræves af LoRa's REST API. Middlend kan opfattes som *Control* i MVC-modellen.

Underopdeling

MOs middleend er underopdelt i en række moduler - se evt. illustrationen i ovenstående afsnit. Formålet med denne modulære opbygning er at gøre koden struktureret (opdelt i en række klare ansvarsområder) og analysérbar samt at facilitere bedre muligheder for at teste kodebasen. MORa-koden består af følgende moduler, som er skrevet i Python:

- **RESTful interface** udviklet i frameworket Flask1 som består af flg.:
  - **LoRa-modul**: håndterer HTTP kommunikationen med LoRas REST API.
  - **Authentication-modul**: Håndterer autentificering.
  - **Routing-modul**: Modtager HTTP kald fra frontenden og kalder logik i de øvrige moduler for at håndtere de indkomne forespørgsler.
  - **Converter-moduler**
    - **Reading-modul**: Konverterer de data, der hentes fra LoRa, til det format, som frontenden forventer.
    - **Writing-modul**: Konverterer data fra frontenden til det format, som LoRa forventer, når der gemmes nye data eller ændres data i LoRa.
    - **Utils-modul**: En samling af nyttig funktioner, som afdækker diverse mindre ansvarsområder (parse datoer, håndtering af URN'er mv.).
  - **Testsuite-modul**

Bemærk, at ovenstående liste ikke nødvendigvis udtømmende, idet der løbende kan blive tilføjet flere moduler i takt med, at kodebasen vokser. Det vil således under videreudviklingsprocessen af og til være nødvendigt at

1. Tilføje nye moduler
2. Splitte eksisterende moduler op i mindre dele for at undgå "responsibility erosion" (dette kunne fx blive relevant for utils-modulet og routing-modulet).

1.3 Opsætning af udviklingsmiljø

I princippet er det muligt at fortage videreudvikling af MORa uden at have en kørende instans af LoRa (idet man blot skriver tests til den udviklede kode), men i praksis vil det være mest praktisk med en kørende LoRa, som man kan udvikle op imod. Det anbefales derfor at installere LoRa i eksempelvis en Linux container som LXC eller lignende, som kører på udviklingsmaskinen. Nærmere instruktioner vedr. selve installationen af LoRa kan findes på LoRas GitHub-side, som er linket til ovenfor.

For at installere de nødvendige afhængigheder på en Ubuntu-maskine, køres følgende kommandoer:

```
$ sudo apt install python3 python3-venv nodejs-legacy npm
```

Efterfølgende klones MORa-projektet fra GitHub:

```
$ mkdir /path/to/folder
$ cd /path/to/folder
$ git clone https://github.com/magenta-aps/mora
```

Man kan nu på sædvanligvis manuelt installere det virtuelle miljø, som Python skal køre i og de nødvendige Python-moduler (med "pip install -r requirements.txt"), men det nemmeste er blot at anvende scriptet **manage.py**:

```
$ cd /path/to/folder/mora
$ ./manage.py run
```

Dette vil automatisk oprette et vituelt Python-miljø, installere de nødvendige Python-afhængigheder og starte applikationen (lyttende på port 5000). Applikationen kan således tilgås på *http://localhost:5000* med et brugernavn og password, som er hhv. *admin* og *secret*. Bemærk dog, at der først skal uploades data til LoRa - til dette formål kan man med fordel anvende **manage.py**.

1.3.1 Generel brug af manage.py

Scriptet manage.py kan bruges til en række forskellige operationer. De mulige funktioner ses ved blot at køre scriptet fra kommandolinjen uden argumenter:

```
$ ./manage.py
```

hvilket vil resultere i flg. output:

```
Usage: manage.py [OPTIONS] COMMAND [ARGS]...

This shell command acts as general utility script for Flask applications.

It loads the application configured (through the FLASK_APP environment
variable) and then provides commands either provided by the application or
Flask itself.

The most useful commands are the "run" and "shell" command.

Example usage:

  $ export FLASK_APP=hello.py
  $ export FLASK_DEBUG=1
  $ flask run

Options:
--version  Show the flask version
--help     Show this message and exit.

Commands:
  auth
  build          Build the frontend application.
  get
  import         Import an Excel spreadsheet into LoRa
  load-fixtures  Import the sample fixtures into LoRA.
  python
  run            Runs a development server.
  shell          Runs a shell in the app context.
  sphinx         Build documentation
  test
  update
```

En liste af mulige funktioner ses under *Commands*. Hvis man fx vil importere et regneark med data til en kørende LoRa-instans, kan dette gøre således (for passende værdier af sti til regneark og URL til LoRa):

```
$ ./manage.py import /sti/til/regneark.xlsx http://lora-url
```

Ønsker man dokumentation for syntaksen af en given kommando, skriver man fx:

```
$ ./manage.py import
```

Som vil angive, hvad den korrekte syntaks er:

```
Usage: manage.py import [OPTIONS] SPREADSHEET [URL]

Error: Missing argument "spreadsheet".
```

For yderligere detaljer om brugen af manage.py henvises til kildekoden.

1.4 Testsuiten

Der arbejdes i proktet med tre typer af tests:

1. Unit tests
2. Integration tests
3. End-to-end tests (Selenium tests)

Der kræves ikke nogen yderligere opsætning for at køre unit testene (samt nogle af integrationstestene), idet disse blot kan køres med kommandoen fra rodmappen af projektet:

```
$ ./manage.py test
```

En del af integrationstestene er sat op til at køre på en sådan måde, at der startes en LoRa-instans før de enkelte test cases kører. Hver test case køres derefter op imod LoRa-instansen, idet der ryddes op i LoRa mellem hver test case, så testene effektivt set køres isoleret. For at anvende denne test feature kræves det, at man installerer *minimox*:

```
$ mkdir /path/to/folder/minimox
$ git clone https://github.com/magenta-aps/mox /path/to/folder/minimox
$ cd /path/to/folder/mox
$ git checkout -b minimox origin/minimox
```

Bemærk at minimox kræver nogle ekstra afhængigeder:

```
$ sudo apt install git python-virtualenv libxmlsec1-openssl postgresql-contrib
```

Det er nu muligt at køre alle integrationstestene vha. den netop installerede minimox:

```
$ ./manage.py test --minimox=/path/to/folder/minimox
```

Ønsker man at se test coverage køres kommandoen:

```
$ ./coverage.py test --minimox=/path/to/folder/minimox
```

som giver et output à la:

```
Name                            Stmts   Miss Branch BrPart  Cover
-----------------------------------------------------------------
mora/__init__.py                    0      0      0      0   100%
mora/app.py                       143     22     30      7    81%
mora/converters/__init__.py         0      0      0      0   100%
mora/converters/addr.py            27      1     10      2    92%
mora/converters/reading.py         58      0     15      0   100%
mora/converters/writing.py        114      0     45      0   100%
mora/exceptions.py                  2      0      0      0   100%
mora/lora.py                      103      8     27      2    89%
mora/util.py                       61      7     41      4    87%
-----------------------------------------------------------------
TOTAL                             508     38    168     15    91%
```

Ønsker man at køre en enkelt testklasse eller blot en enkelt test case, kan det gøres på følgende måde:

```
$ ./manage.py test --minimox=/path/to/folder/minimox tests.test_integration.
↪IntegrationTests
$ ./manage.py test --minimox=/path/to/folder/minimox tests.test_integration.
↪IntegrationTests.test_should_add_one_new_contact_channel_correctly
```

1.5 Installering

Gør følgende for at installere MORa på Ubuntu 16.04:

```
# først, klon MORa
sudo install -d -o $UID -g $GID /srv/mora
git clone https://github.com/magenta-aps/mora /srv/mora

# installér afhængigheder
sudo apt install python3-venv nodejs-legacy npm

# byg applikationen; dette opretter det virtuelle miljø
/srv/mora/manage.py build
# installér gunicorn
/srv/mora/venv-linux-cpython-3.5/bin/pip install gunicorn gevent

# opret en bruger og installer den krævede infrastruktur
sudo adduser --system \
  --home /srv/mora \
  --shell /usr/sbin/nologin \
  --disabled-password --disabled-login \
  --ingroup www-data mora
sudo install -d -o mora -g www-data /var/log/mora /run/mora
sudo install -m 644 /srv/mora/config/mora.service /etc/systemd/system
sudo install -m 644 /srv/mora/config/mora.socket /etc/systemd/system
sudo install -m 644 /srv/mora/config/mora.conf /etc/tmpfiles.d

sudo systemctl daemon-reload
sudo systemctl enable mora.socket mora.service
sudo systemctl start mora.service
```

Du har nu en funktionel installation af MORa som lytter på et lokalt socket. For at eksponere den udadtil skal Apache eller nginx konfigureres til at videresende forespørgsler. For eksempel anvendes følgende til Apache:

```
SSLProxyEngine on

<Location /mo/>
    ProxyPass unix:/run/mora/socket|http://localhost/
    ProxyPassReverse http://localhost/
</Location>
```

Aktivér modulet `proxy_http`, og genstart Apache:

```
sudo a2enmod proxy_http
sudo apache2ctl graceful
```

Til sidst kopieres `config/mora-example.json` til `config/mora.json` og `LORA_URL` justeres til at pege der hvor du har LoRa kørende:

```
{
  "LORA_URL": "https://lora.example.com/"
}
```

Bemærk venligst at anvendelse af HTTPS kræver et betroet certifikat på serveren, og at autentificering med SAML kræver yderligere konfiguration.

1.6 Dokumentation

Det er muligt at autogenerere dokumentation ud fra doc-strings i kildekoden. Til dette anvendes Sphinx. Kør nedenstående kommando for at autogenerere dokumentationen:

```
$ ./manage.py sphinx
```

Dokumentation kan nu findes ved at åbne filen `/sti/til/mora/docs/out/index.html`.

CHAPTER 2

Manual testing of MO

Before merging a branch into master, check that the following is ok. This is not a complete test of all features, but just some quick tests to check that all the major features are not unexpectedly broken.

2.1 Testing the basic features

Remember to check that everything looks ok (maybe also directly in LoRa) after each of the following steps:

1. Create an org unit without specifying an enddate
2. Create an org unit with a final enddate
3. End the org unit created in 1)
4. End the org unit again with a date later than the enddate from 3)
5. Move the org unit created in 1) to beneath the org unit from 2) (with immediate effect)
6. Rename the org unit created in 1) (with immediate effect)
7. Add a contact channel to the org unit from 1) (with immediate effect)
8. Add two more contact channels to the org unit from 1) (with immediate effect)
9. Add another location (not primary address) to the org unit from 1) (with immediate effect)
10. Add another location (primary address) to the org unit from 1) (with immediate effect)
11. Change the address of the just added location (with immediate effect)
12. TODO: more editing of locations and contact
13. Edit the org unit from 1) by changing the unit type - leave the date unchanged
14. Edit the org unit from 1) by changing the org unit startdate (i.e. ‘gyldighed’)
15. Edit both the unit type and the org unit startdate in one go (for the org unit from 1))
16. Test that search is working (with and without wildcards)
17. Check that it is not possible to create an org unit with an invalid date range
18. Check that it is not possible to move an org unit to its own subtree
19. Check that it is not possible to update an address to an empty value

20. Check that the "History" button is working

2.2 Testing temporality

Wipe the DB and use the spreadsheet found in `sandbox/AAK/AARHUS_3orgUnits.xlsx` to test the following (or just make a similar scenario on the LoRa instance, you have running). Check that you can construct the following temporal org unit configuration. Begin by setting the org unit name and 'overordnet' as:

```
-----------[--------n1--------)[-----n2-----)[---------n3-------------
-----------[-------------------------o1-------------------------------
```

where `n` refers to the name and `o` refers to 'overordnet'. Then set the org unit 'overordnet' as:

```
-----------[----o1---)[---------------o2------------------)[---o1------
```

and check that the resulting configuration:

```
-----------[--n1,o1--)[-n1,o2-)[---n2,o2----)[-n3,o2-----)[---n3,o1---
```

is shown correctly in the frontend.

Also test that it is possible to add a new location in the past and a new location in the future for the same org unit.

CHAPTER 3

API dokumentation

Bemærk venligst at selve kodedokumentationen er på engelsk, eftersom det sprog anvendes i al koden.

3.1 Address

This section describes how to interact with addresses.

Resource	Operation	Description
Address	*GET /service/o/(uuid:orgid)/address_autocomplete*	Autocomplete

GET /service/o/(**uuid:** *orgid*)**/address_autocomplete/**
Perform address autocomplete :param orgid: The UUID of the organisation

Query Parameters

- **q** (*str*) – A query string to be used for lookup
- **global** (*boolean*) – Whether or not the lookup should be in the entire country, or contained to the municipality of the organisation

Example Response:

Request JSON Array of Objects

- **uuid** (*uuid*) – A UUID of a DAR address
- **name** (*str*) – A human readable name for the address

```
[
  {
    "location": {
      "uuid": "f0396d0f-ef2d-41e5-a420-b4507b26b6fa",
      "name": "Rybergsvej 1, Sønderby, 5631 Ebberup"
    }
  },
  {
    "location": {
      "uuid": "0a3f50cb-05eb-32b8-e044-0003ba298018",
      "name": "Wild Westvej 1, 9310 Vodskov"
    }
```

```
    }
]
```

3.2 Legacy service layer

This is the legacy service API retained for compatibility with the original UI.

Resource	Operation	Description
	POST /mo/e/(uuid:employeeid)/role-types/(any:role)	
	POST /mo/mo/e/(uuid:employeeid)/actions/role	
	POST /mo/e/(uuid:employeeid)/actions/role	
	POST /mo/o/(uuid:orgid)/org-unit	
	POST /mo/org-unit	
	DELETE /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)	
	DELETE /mo/org-unit/(uuid:unitid)	
	POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)	
	POST /mo/org-unit/(uuid:unitid)	
	POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/actions/move	
	POST /mo/org-unit/(uuid:unitid)/actions/move	
	POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/role-types/location/(uuid:locid)	
	POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/role-types/location	
	POST /mo/org-unit/(uuid:unitid)/role-types/location	
IT systems	*GET /mo/o/(uuid:orgid)/it/*	List
	GET /mo/it/	
	GET /mo/it-system/	
Roles	*GET /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/role-types/(any:role)/*	Get
	GET /mo/org-unit/(uuid:unitid)/role-types/(any:role)/	
	GET /mo/e/(uuid:userid)/role-types/(any:role)/	
	GET /mo/role-types/	Available roles

GET /mo/addressws/geographical-location

GET /mo/e/

GET /mo/e/(**int:** *cpr_number*)**/**

POST /mo/e/(**uuid:** *employeeid*)**/role-types/**
any: *role*

POST /mo/mo/e/(**uuid:** *employeeid*)**/actions/role**

POST /mo/e/(**uuid:** *employeeid*)**/actions/role**
Catch-all function for creating Employees roles

Parameters

- **employeeid** – Employee ID from MO. Not used.

Return The uuid of the employee and a HTTP status code.

GET /mo/e/(**uuid:** *id*)**/**

GET /mo/o/

GET /mo/o/(**uuid:** *id*)**/**

GET /mo/o/(**uuid:** *orgid*)**/full-hierarchy**

GET /mo/o/(**uuid:** *orgid*)**/it/**

GET /mo/it/

GET /mo/it-system/
List the available IT systems.

Parameters

- **orgid** (*string*) – optional organisation UUID restricting the search

Response JSON Array of Objects

- **name** (*string*) – user-friendly name
- **userKey** (*string*) – unique, human-readable identifyer
- **uuid** (*string*) – unique, machine-friendly identifyer

Status Codes

- 200 OK – Always, even if nothing found.
- 404 Not Found – Not used.

POST /mo/o/(**uuid:** *orgid*)**/org-unit**

POST /mo/org-unit
Create a new org unit.

Parameters

- **orgid** – The UUID of the organisation (not used, but given by the frontend).

Return JSON containing the new org unit UUID and the response status code.

GET /mo/o/(**uuid:** *orgid*)**/org-unit/**

DELETE /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid*

DELETE /mo/org-unit/(**uuid:** *unitid*)
Inactivate an org unit.

Parameters

- **orgid** – The UUID of the organisation (not used, but given by the frontend).
- **unitid** – The UUID of the org unit.

Return JSON containing the org unit UUID and the response status code.

POST /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid*

POST /mo/org-unit/(**uuid:** *unitid*)
Change the name or the type of an org unit.

Parameters

- **orgid** – The UUID of the organisation.
- **unitid** – The UUID of the org unit.

Return JSON containing the org unit UUID and the response status code.

GET /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/**

GET /mo/org-unit/(**uuid:** *unitid*)**/**

POST /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/actions/move**

POST /mo/org-unit/(**uuid:** *unitid*)**/actions/move**
Move an org unit.

Parameters

- **orgid** – The UUID of the organisation (not used, but given by the frontend).
- **unitid** – The UUID of the org unit.

Return JSON containing the org unit UUID and the response status code.

GET /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/history/**

GET /mo/org-unit/(**uuid:** *unitid*)**/history/**

GET /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/role-types/any:** *role***/**

GET /mo/org-unit/(**uuid:** *unitid*)**/role-types/**
any: *role***/**

GET /mo/e/(**uuid:** *userid*)**/role-types/**
any: *role***/** Get the roles of an employee or organisational unit.

Query Parameters

- **validity** (*string*) – Only yield entries relevant to the past, present or future.
- **effective-date** (*date*) – Set the "current" date for the time machine.
- **unique** (*boolean*) – Retained for compatibility with original UI.
- **t** (*int*) – Retained for compatibility with original UI.

Parameters

- **role** (*string*) – The relevant role type, see *GET /mo/role-types/*
- **userid** (*uuid*) – Optional employee UUID.
- **unitid** (*uuid*) – Optional unit UUID.
- **orgid** (*uuid*) – Optional organisation UUID, retained for compatibility with original UI.

POST /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/role-types/location/uuid:** *locid*

POST /mo/o/(**uuid:** *orgid*)**/org-unit/**
uuid: *unitid***/role-types/location**

POST /mo/org-unit/(**uuid:** *unitid*)**/role-types/location**
Add a location or contact channel or update existing ones.

Parameters

- **orgid** – The UUID of the organisation.
- **unitid** – The UUID of the org unit.
- **locid** – The UUID of the location (i.e. the address UUID).

Return JSON containing the org unit UUID and the response status code.

GET /mo/org-unit/type

GET /mo/role-types/
List the supported role types.

GET /mo/role-types/contact/facets/properties/classes/

GET /mo/role-types/contact/facets/type/classes/

GET /mo/role-types/engagement/facets/(**any:** *facet*)**/classes/**

3.3 Authentication

This section describes how to authenticate with MO. The API is work in progress.

Resource	Operation	Description
Authentication	*POST /mo/acl/*	Deprecated.
	GET /mo/acl/	
	POST /mo/service/user/(user)/logout	Log out
	POST /mo/service/user/(username)/login	Log in

POST /mo/acl/
Obtain the access control lists of the user — of which we have none.

Deprecated Retained for compatibility with original UI.

GET /mo/acl/
Obtain the access control lists of the user — of which we have none.

Deprecated Retained for compatibility with original UI.

POST /mo/service/user/(*user*)**/logout**
Attempt to log out as the given user name.

Parameters

- **username** – The user ID to logout as.

Return Nothing.

Status Codes

- 200 OK – The logout succeeded — which it almost always does.

POST /mo/service/user/(*username*)**/login**
Attempt a login as the given user name. The internals of this login will be kept from the JavaScript by using httpOnly cookies.

Status Codes

- 200 OK – The login succeeded.
- 401 Unauthorized – The login failed.

Parameters

- **username** – The user ID to login as.

Request JSON Object

- **password** (*string*) – The password of the user.
- **rememberme** (*boolean*) – Whether to persist the login — currently ignored.

Response JSON Object

- **user** (*string*) – The name of the user.
- **token** (*string*) – Retained for compatibility with original UI.

- **role** (*string*) – Retained for compatibility with original UI.

3.4 CPR

This section describes functionality for retrieving information about people based on their CPR number.

Resource	Operation	Description
	GET /service/e/cpr_lookup/	

GET /service/e/cpr_lookup/
Search for a CPR number in Serviceplatformen and retrieve the associated information

Query Parameters

- **q** – The CPR no. of a person to be searched

Request JSON Array of Objects

- **name** (*string*) – The name of the person
- **cpr_no** (*string*) – The person's CPR number.

Example Response:

```
{
  "name": "John Doe",
  "cpr_no": "1234567890"
}
```

3.5 Details

This section describes how to interact with employee and organisational unit metadata, referred to as *details* within this API.

Resource	Operation	Description
Detail	*GET /service/(any:type)/(uuid:id)/details/*	List
	GET /service/(any:type)/(uuid:id)/details/(function)	Get

GET /service/(**any:** *type*)**/**
uuid: *id***/details/** List the available 'detail' types under this entry.

Example response:

```
{
  "address": false,
  "association": false,
  "engagement": true,
  "it": false,
  "leave": true,
  "manager": false,
  "role": false
}
```

The value above informs you that at least one entry exists for each of 'engagement' and 'leave' either in the past, present or future.

GET /service/(**any:** *type*)**/**
uuid: *id***/details/***function* Obtain the list of engagements, associations, roles, etc. corresponding

to a user or organisational unit. See *GET /service/(any:type)/(uuid:id)/details/* for the available list of endpoints.

Most of these endpoints are broadly similar to engagements, with the notable exception being IT systems.

All requests contain validity objects on the following form:

Request JSON Array of Objects

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.
- **validity** (*string*) – Only show *past*, *present* or *future* values – which the default being to show *present* values.
- **start** (*int*) – Index of first item for paging.
- **limit** (*int*) – Maximum items.

Parameters

- **type** – 'ou' for querying a unit; 'e' for querying an employee.
- **id** (*uuid*) – The UUID to query, i.e. the ID of the employee or unit.
- **function** – See *GET /service/(any:type)/(uuid:id)/details/* for the available values for this field.

Status Codes

- 200 OK – Always.

Example engagement response:

Request JSON Array of Objects

- **job_function** (*object*) – See *GET /service/o/(uuid:orgid)/f/(facet)/*.
- **type** (*object*) – See *GET /service/o/(uuid:orgid)/f/(facet)/*.
- **org_unit** (*object*) – See *GET /service/o/(uuid:orgid)/f/(facet)/* .
- **uuid** (*string*) – Machine-friendly UUID.
- **validity** (*string*) – The validity times of the object.

```
[
      {
          "job_function": {
              "example": null,
              "name": "Fakultet",
              "scope": null,
              "user_key": "fak",
              "uuid": "4311e351-6a3c-4e7e-ae60-8a3b2938fbd6"
          },
          "org_unit": {
              "name": "Humanistisk fakultet",
              "user_key": "hum",
```

```
            "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e"
        },
        "person": {
            "name": "Anders And",
            "uuid": "53181ed2-f1de-4c4a-a8fd-ab358c2c454a"
        },
        "engagement_type": {
            "example": null,
            "name": "Afdeling",
            "scope": null,
            "user_key": "afd",
            "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
        },
        "uuid": "d000591f-8705-4324-897a-075e3623f37b",
        "validity": {
            "from": "2017-01-01T00:00:00+01:00",
            "to": null
        },
    }
]
```


Example association response:


```
[
  {
    "address": {
      "href": "https://www.openstreetmap.org/"
              "?mlon=12.57924839&mlat=55.68113676&zoom=16",
      "name": "Pilestræde 43, 3., 1112 København K",
      "value": "0a3f50a0-23c9-32b8-e044-0003ba298018"
    },
    "address_type": {
      "example": "<UUID>",
      "name": "Adresse",
      "scope": "DAR",
      "user_key": "Adresse",
      "uuid": "4e337d8e-1fd2-4449-8110-e0c8a22958ed"
    },
    "association_type": {
      "example": null,
      "name": "Medlem",
      "scope": null,
      "user_key": "medl",
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "job_function": {
      "example": null,
      "name": "Hund",
      "scope": null,
      "user_key": "hund",
      "uuid": "c2b23c43-87c6-48bb-a99c-53396bfa99fb"
    },
    "org_unit": {
      "name": "Humanistisk fakultet",
      "user_key": "hum",
      "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e"
    },
    "person": {
      "name": "Fedtmule",
      "uuid": "6ee24785-ee9a-4502-81c2-7697009c9053"
    },
    "uuid": "30cd25e1-b21d-46fe-b299-1c1265e9be66",
    "validity": {
```

```
      "from": "2017-01-01T00:00:00+01:00",
      "to": "2018-01-01T00:00:00+01:00"
    }
  }
]
```


Example IT response:


```
[
  {
    "name": "Active Directory",
    "user_name": "Fedtmule",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb",
    "validity": {
      "from": "2002-02-14T00:00:00+01:00",
      "to": null
    }
  }
]
```


Example address response:


```
[
   {
     "name": "Christiansborg Slotsplads 1, 1218 København K",
     "value": "bae093df-3b06-4f23-90a8-92eabedb3622"
     "href": "https://www.openstreetmap.org/"
         "?mlon=12.58176945&mlat=55.67563739&zoom=16",
     "address_type": {
       "scope": "DAR"
     },
     "validity": {
       "from": "2002-02-14T00:00:00+01:00",
       "to": null
     },
   },
   {
     "name": "goofy@example.com",
     "href": "mailto:goofy@example.com",
     "value": "urn:mailto:goofy@example.com"
     "address_type": {
       "example": "test@example.com",
       "name": "Emailadresse",
       "scope": "EMAIL",
       "user_key": "Email",
       "uuid": "c78eb6f7-8a9e-40b3-ac80-36b9f371c3e0"
     },
     "validity": {
       "from": "2002-02-14T00:00:00+01:00",
       "to": null
     },
   },
   {
     "name": "goofy@example.com",
     "href": "mailto:goofy@example.com",
     "value": "urn:mailto:goofy@example.com"
     "address_type": {
       "example": "test@example.com",
       "name": "Emailadresse",
       "scope": "EMAIL",
       "user_key": "Email",
       "uuid": "c78eb6f7-8a9e-40b3-ac80-36b9f371c3e0"
     },
```

```
      "validity": {
        "from": "2002-02-14T00:00:00+01:00",
        "to": null
      },
    }
]
```

Example org_unit response:

An array of objects as returned by *GET /service/ou/(uuid:unitid)/*.

```
[
  {
    "name": "Afdeling for Fortidshistorik",
    "user_key": "frem",
    "uuid": "04c78fc2-72d2-4d02-b55f-807af19eac48"
    "org": {
      "name": "Aarhus Universitet",
      "user_key": "AU",
      "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62"
    },
    "org_unit_type": {
      "example": null,
      "name": "Afdeling",
      "scope": null,
      "user_key": "afd",
      "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
    },
    "parent": {
      "name": "Historisk Institut",
      "user_key": "hist",
      "uuid": "da77153e-30f3-4dc2-a611-ee912a28d8aa"
    },
    "validity": {
      "from": "2018-01-01T00:00:00+01:00",
      "to": "2019-01-01T00:00:00+01:00"
    }
  }
]
```

Example manager response:

3.6 Employees

This section describes how to interact with employees.

Resource	Operation	Description
	GET /service/e/(uuid:employee_uuid)/history/	
Employee	*POST /service/e/(uuid:employee_uuid)/create*	Create relation
	POST /service/e/(uuid:employee_uuid)/edit	Edit employee
	POST /service/e/(uuid:employee_uuid)/terminate	Terminate
	GET /service/e/(uuid:id)/	Get
	POST /service/e/create	Create
	GET /service/o/(uuid:orgid)/e/	List & search

POST /service/e/(**uuid:** *employee_uuid*)**/create**

Creates new employee relations

Status Codes

- 200 OK – Creation succeeded.

Parameters

- **employee_uuid** – The UUID of the employee.

All requests contain validity objects on the following form:

Request JSON Array of Objects

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

Request payload contains a list of creation objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Engagement:

Request JSON Array of Objects

- **type** (*string*) – **"engagement"**
- **org_unit** (*string*) – The associated org unit
- **job_function** (*string*) – The job function of the association
- **engagement_type** (*string*) – The engagement type
- **validity** (*object*) – The validities of the created object.

```
[
  {
    "type": "engagement",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "job_function": {
      "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
    },
    "engagement_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Association:

Request JSON Array of Objects

- **type** (*string*) – **"association"**
- **org_unit** (*string*) – The associated org unit
- **job_function** (*string*) – The job function of the association
- **association_type** (*string*) – The association type
- **location** (*string*) – The associated location.
- **validity** (*object*) – The validities of the created object.

```
[
  {
    "type": "association",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "job_function": {
      "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
    },
    "association_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "location": {
      "uuid": "89faa44c-f37a-4e4a-9cd8-b25f67cfd7bc"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    },
  }
]
```

IT system:

Request JSON Object

- **type** (*string*) – `"it"`
- **itsystem** (*object*) – The IT system to create a relation to, as returned by *GET /service/o/(uuid:orgid)/it/*. The only mandatory field is `uuid`.

```
[
    {
        "type": "it",
        "itsystem": {
            "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb"
        },
        "validity": {
            "from": "2017-12-01T00:00:00+01",
            "to": null
        }
    }
]
```

Role:

Request JSON Array of Objects

- **type** (*string*) – **"role"**
- **org_unit** (*string*) – The associated org unit
- **role_type** (*string*) – The role type
- **validity** (*object*) – The validities of the created object.

```
[
  {
    "type": "role",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "role_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
```

```
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Manager:

Request JSON Array of Objects

- **type** (*string*) – **"manager"**
- **org_unit** (*string*) – The associated org unit
- **manager_type** (*string*) – The manager type
- **responsibility** (*string*) – The manager responsibility
- **manager_level** (*string*) – The manager level
- **validity** (*object*) – The validities of the created object.

```
[
  {
    "type": "manager",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "manager_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "responsibility": {
      "uuid": "e6b24f90-b056-433b-ad65-e6ab95d25826"
    },
    "manager_level": {
      "uuid": "f17f2d60-9750-4577-a367-8a5f065b63fa"
    },
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Leave:

Request JSON Array of Objects

- **type** (*string*) – **"leave"**
- **leave_type** (*string*) – The leave type
- **validity** (*object*) – The validities of the created object.

```
[
  {
    "type": "leave",
    "leave_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    },
```

```
  }
]
```

Address:

Request JSON Array of Objects

- **type** (*string*) – "address"
- **address_type** (*object*) – The type of the address, exactly as returned by returned by *GET /service/o/(uuid:orgid)/f/(facet)/*.
- **address** (*string*) – The value of the address field. Please note that as a special case, this should be a UUID for *DAR* addresses.

```
[
  {
    "value": "1234567890",
    "address_type": {
      "example": "5712345000014",
      "name": "EAN",
      "scope": "EAN",
      "user_key": "EAN",
      "uuid": "e34d4426-9845-4c72-b31e-709be85d6fa2"
    },
    "type": "address",
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

POST /service/e/(**uuid:** *employee_uuid*)**/edit**

Edits an employee

Status Codes

- 200 OK – The edit succeeded.

All requests contain validity objects on the following form:

Request JSON Array of Objects

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00"
}
```

Request payload contains a list of edit objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Engagement:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – **"engagement"**
- **uuid** (*string*) – The UUID of the engagement,

- **original** (*object*) – An **optional** object containing the original state of the engagement to be overwritten. If supplied, the change will modify the existing registration on the engagement object. Detailed below.
- **data** (*object*) – An object containing the changes to be made to the engagement. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **org_unit** (*string*) – The associated org unit
- **job_function** (*string*) – The job function of the association
- **engagement_type** (*string*) – The engagement type
- **validity** (*object*) – The validities of the changes.

```
[
  {
    "type": "engagement",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      },
      "engagement_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      }
    },
    "data": {
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2019-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      }
    }
  }
]
```

Association:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – **"association"**
- **uuid** (*string*) – The UUID of the association,
- **original** (*object*) – An **optional** object containing the original state of the association to be overwritten. If supplied, the change will modify the existing registration on the association object. Detailed below.

- **data** (*object*) – An object containing the changes to be made to the association. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **org_unit** (*string*) – The associated org unit
- **job_function** (*string*) – The job function of the association
- **association_type** (*string*) – The association type
- **location** (*string*) – The associated location.
- **validity** (*object*) – The validities of the changes.

```
[
  {
    "type": "association",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      },
      "association_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      },
      "location": {
        "uuid": "89faa44c-f37a-4e4a-9cd8-b25f67cfd7bc"
      }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      }
    }
  }
]
```

IT system:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – `"it"`
- **uuid** (*string*) – The UUID of the IT system,
- **original** (*object*) – An **optional** object containing the original state of the role to be overwritten. If supplied, the change will modify the existing registration on the role object. Detailed below.

- **data** (*object*) – An object containing the changes to be made to the role. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **uuid** (*string*) – Change the IT system to another.

```
[
  {
    "type": "it",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb",
    "original": {
      "name": "Active Directory",
      "user_name": "Fedtmule",
      "uuid": "00000000-0000-0000-0000-000000000000",
      "validity": {
        "from": "2002-02-14T00:00:00+01:00",
        "to": null
      }
    },
    "data": {
      "uuid": "11111111-1111-1111-1111-111111111111",
      "validity": {
        "to": "2020-01-01T00:00:00+01:00"
      }
    }
  }
]
```

Role:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – **"role"**
- **uuid** (*string*) – The UUID of the role,
- **original** (*object*) – An **optional** object containing the original state of the role to be overwritten. If supplied, the change will modify the existing registration on the role object. Detailed below.
- **data** (*object*) – An object containing the changes to be made to the role. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **org_unit** (*string*) – The associated org unit
- **role_type** (*string*) – The role type
- **location** (*string*) – The associated location.
- **validity** (*object*) – The validities of the changes.

```
[
  {
    "type": "role",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
```

```
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "role_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "role_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```

Leave:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – **"leave"**
- **uuid** (*string*) – The UUID of the leave,
- **original** (*object*) – An **optional** object containing the original state of the leave to be overwritten. If supplied, the change will modify the existing registration on the leave object. Detailed below.
- **data** (*object*) – An object containing the changes to be made to the leave. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **leave_type** (*string*) – The leave type
- **validity** (*object*) – The validities of the changes.

```
[
  {
    "type": "leave",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "leave_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      }
    },
    "data": {
```

```
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "leave_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```

Manager:

Parameters

- **employee_uuid** – The UUID of the employee.

Request JSON Object

- **type** (*string*) – **"manager"**
- **uuid** (*string*) – The UUID of the manager,
- **original** (*object*) – An **optional** object containing the original state of the leave to be overwritten. If supplied, the change will modify the existing registration on the leave object. Detailed below.
- **data** (*object*) – An object containing the changes to be made to the leave. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Array of Objects

- **manager_type** (*string*) – The manager type
- **org_unit** (*string*) – The associated org unit
- **manager_type** – The manager type
- **responsibility** (*string*) – The manager responsibility
- **manager_level** (*string*) – The manager level
- **validity** (*object*) – The validities of the changes.

```
[
  {
    "type": "manager",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
        "org_unit": {
          "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
        },
        "manager_type": {
          "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
        },
        "responsibility": {
          "uuid": "e6b24f90-b056-433b-ad65-e6ab95d25826"
        },
        "manager_level": {
          "uuid": "f17f2d60-9750-4577-a367-8a5f065b63fa"
        },
        "validity": {
            "from": "2016-01-01T00:00:00+00:00",
            "to": "2018-01-01T00:00:00+00:00"
        }
```

```
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "manager_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```

GET /service/e/(**uuid:** *employee_uuid*)**/history/**

Get the history of an employee :param employee_uuid: The UUID of the employee

Example response:

Request JSON Array of Objects

- **from** (*string*) – When the change is active from
- **to** (*string*) – When the change is active to
- **action** (*string*) – The action performed
- **life_cycle_code** (*string*) – The type of action performed
- **user_ref** (*string*) – A reference to the user who made the change

```
[
  {
    "from": "2018-02-21T11:27:20.909206+01:00",
    "to": "infinity",
    "action": "Opret orlov",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T11:27:20.803682+01:00",
    "to": "2018-02-21T11:27:20.909206+01:00",
    "action": "Rediger engagement",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T11:27:20.619990+01:00",
    "to": "2018-02-21T11:27:20.803682+01:00",
    "action": null,
    "life_cycle_code": "Importeret",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  }
]
```

POST /service/e/(**uuid:** *employee_uuid*)**/terminate**

Terminates an employee and all of his roles from a specified date.

Status Codes

- 200 OK – The termination succeeded.

Parameters

- **employee_uuid** – The UUID of the employee to be terminated.

Request JSON Object

- **valid_from** (*string*) – The date on which the termination should happen, in ISO 8601.

Example Request:

```
{
  "validity": {
    "from": "2016-01-01T00:00:00+00:00"
  }
}
```

GET /service/e/(uuid: *id*)/
Retrieve an employee.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.

Request JSON Object

- **name** (*string*) – Human-readable name.
- **uuid** (*string*) – Machine-friendly UUID.
- **org** (*object*) – The organisation that this employee belongs to, as yielded by *GET /service/o/*.
- **cpr_no** (*string*) – CPR number of for the corresponding person. Please note that this is the only means for obtaining the CPR number; due to confidentiality requirements, all other end points omit it.

Status Codes

- 200 OK – Whenever the user ID is valid and corresponds to an existing user.
- 404 Not Found – Otherwise.

Example Response:

```
{
  "cpr_no": "1011101010",
  "name": "Hans Bruger",
  "uuid": "9917e91c-e3ee-41bf-9a60-b024c23b5fe3",
  "org": {
    "name": "Magenta ApS",
    "user_key": "Magenta ApS",
    "uuid": "8efbd074-ad2a-4e6a-afec-1d0b1891f566"
  }
}
```

POST /service/e/create
Create a new employee

Status Codes

- 200 OK – Creation succeeded.

Example Request:

Request JSON Object

- **name** (*string*) – The name of the employee
- **cpr_no** (*string*) – The CPR no of the employee
- **org** (*object*) – The organisation with which the employee is associated

```
{
  "name": "Name Name",
  "cpr_no": "1234567890",
  "org": {
    "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
  }
}
```

Returns UUID of created employee

GET /service/o/(**uuid:** *orgid*)**/e/**

Query employees in an organisation.

Parameters

- **orgid** (*uuid*) – UUID of the organisation to search.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.
- **start** (*int*) – Index of first unit for paging.
- **limit** (*int*) – Maximum items
- **query** (*string*) – Filter by employees matching this string. Please note that this only applies to attributes of the user, not the relations or engagements they have.

Request JSON Object

- **items** (*string*) – The returned items.
- **offset** (*string*) – Pagination offset.
- **total** (*string*) – Total number of items available on this query.

Request JSON Array of Objects

- **name** (*string*) – Human-readable name.
- **uuid** (*string*) – Machine-friendly UUID.

Status Codes

- 200 OK – Always.

Example Response:

```
{
  "items": [
    {
      "name": "Hans Bruger",
      "uuid": "9917e91c-e3ee-41bf-9a60-b024c23b5fe3"
    },
    {
      "name": "Joe User",
      "uuid": "cd2dcfad-6d34-4553-9fee-a7023139a9e8"
    }
  ],
  "offset": 0,
  "total": 1
}
```

3.7 Facets

This sections describes how to interact with facets, i.e. the types of objects.

Resource	Operation	Description
Facet	*GET /service/o/(uuid:orgid)/f/*	List types
	GET /service/o/(uuid:orgid)/f/(facet)/	Get

GET /service/o/(**uuid:** *orgid*)**/f/**
List the facet types available in a given organisation.

Parameters

- **orgid** (*uuid*) – Restrict search to this organisation.

Response JSON Array of Objects

- **name** (*string*) – The facet name.
- **path** (*string*) – The location on the web server.
- **desc** (*string*) – Human readable description.

Status Codes

- 200 OK – Always.

Example Response:

```
[
  {
    "name": "address",
    "path": "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/address/",
    "user_key": "Adressetype",
    "uuid": "e337bab4-635f-49ce-aa31-b44047a43aa1"
  },
  {
    "name": "ou",
    "path": "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/ou/",
    "user_key": "Enhedstype",
    "uuid": "fc917e7c-fc3b-47c2-8aa5-a0383342a280"
  }
]
```

GET /service/o/(**uuid:** *orgid*)**/f/**
*facet***/** List classes available in the given facet.

Parameters

- **orgid** (*uuid*) – Restrict search to this organisation.
- **facet** (*string*) – One of the facet names listed by *GET /service/o/(uuid:orgid)/f/*

Query Parameters

- **start** (*int*) – Index of first item for paging.
- **limit** (*int*) – Maximum items.

Response JSON Array of Objects

- **name** (*string*) – Human-readable name.
- **uuid** (*string*) – Machine-friendly UUID.

- **user_key** (*string*) – Short, unique key.
- **example** (*string*) – An example value. In most cases the value is meant to be presented to the user as an aid.
- **scope** (*string*) – A representation of the type of value, see the table below for details.

Status Codes

- 200 OK – On success.
- 404 Not Found – Whenever the facet isn't found.

Scopes:

Key	Description
"DAR"	UUID of a DAR address, as found through the API. Please note that this requires performing separate calls to convert this value to and from human-readable strings.
"EMAIL"	An email address, as specified by **RFC 5322#section-3.4**.
"INTEGER"	An integral number.
"PHONE"	A phone number.
"TEXT"	Arbitrary text.
"WWW"	An HTTP or HTTPS URL, as specified by **RFC 1738**.

Example Response:


```
{
  "name": "address_type",
  "path":
    "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/address_type/",
  "user_key": "Adressetype",
  "uuid": "e337bab4-635f-49ce-aa31-b44047a43aa1",
  "data": {
    "items": [
      {
        "example": "http://www.korsbaek.dk/",
        "name": "Hjemmeside",
        "scope": "WWW",
        "user_key": "URL",
        "uuid": "160ecaed-50b0-4800-bebc-0d0289a4f624"
      },
      {
        "example": "<UUID>",
        "name": "Lokation",
        "scope": "DAR",
        "user_key": "AdresseLokation",
        "uuid": "031f93c3-6bab-462e-a998-87cad6db3128"
      },
      {
        "example": "Mandag 10:00-12:00 Tirsdag 14:00-16:00",
        "name": "Åbningstid, telefon",
        "scope": "TEXT",
        "user_key": "Åbningstid Telefon",
        "uuid": "0836ffbf-3b3e-410f-8cbf-face7e6844ef"
      }
    ],
    "offset": 0,
    "total": 3
  }
}
```

3.8 IT Systems

This section describes how to interact with IT systems.

Resource	Operation	Description
IT system	*GET /service/o/(uuid:orgid)/it/*	List available systems

GET /service/o/(**uuid:** *orgid*)**/it/**

List the IT systems available within the given organisation.

Parameters

- **orgid** (*uuid*) – Restrict search to this organisation.

Response JSON Array of Objects

- **uuid** (*string*) – The universally unique identifier of the system.
- **name** (*string*) – The name of the system.
- **type** (*string*) – The type of the system.
- **user_key** (*string*) – A human-readable unique key for the system.

Status Codes

- 200 OK – Always.

Example Response:

```
[
  {
    "name": "Lokal Rammearkitektur",
    "type": null,
    "user_key": "LoRa",
    "uuid": "0872fb72-926d-4c5c-a063-ff800b8ee697"
  },
  {
    "name": "Active Directory",
    "type": null,
    "user_key": "AD",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb"
  }
]
```

3.9 Organisation

This section describes how to interact with organisations.

Resource	Operation	Description
Organisation	*GET /service/o/*	List
	GET /service/o/(uuid:orgid)/	Getter

GET /service/o/

List displayable organisations. We consider anything that has *at least one* root unit 'displayable'.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.

Request JSON Array of Objects

- **name** (*string*) – Human-readable name of the organisation.
- **user_key** (*string*) – Short, unique key identifying the unit.
- **uuid** (*string*) – Machine-friendly UUID of the organisation.

Status Codes

- 200 OK – Always.

Example Response:

```
[
  {
    "name": "Aarhus Kommune",
    "user_key": "AARHUS",
    "uuid": "59141156-ed0b-457c-9535-884447c5220b"
  },
  {
    "name": "Ballerup Kommune",
    "user_key": "BALLERUP",
    "uuid": "3a87187c-f25a-40a1-8d42-312b2e2b43bd"
  }
]
```

GET /service/o/(uuid: *orgid*)/
Obtain the initial level of an organisation.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.

Request JSON Object

- **name** (*string*) – Human-readable name of the organisation.
- **user_key** (*string*) – Short, unique key identifying the unit.
- **uuid** (*string*) – Machine-friendly UUID of the organisation.
- **child_count** (*int*) – Number of org. units nested immediately beneath the organisation.
- **person_count** (*int*) – Amount of people belonging to this organisation.
- **unit_count** (*int*) – Amount of units belonging to this organisation.
- **employment_count** (*int*) – Amount of employments in this organisation?

Status Codes

- 200 OK – Whenever the organisation exists and is readable.
- 404 Not Found – When no such organisation exists.

Example Response:

```
{
  "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62",
  "name": "Aarhus Universitet",
  "user_key": "AU",
  "person_count": 2,
  "child_count": 1,
  "unit_count": 6,
  "employment_count": 1,
  "association_count": 1,
  "leave_count": 1,
  "role_count": 1,
  "manager_count": 1
}
```

3.10 Organisational units

This section describes how to interact with organisational units.

Resource	Operation	Description
	GET /service/ou/(uuid:unitid)/history/	
Unit	*GET /service/(any:type)/(uuid:parentid)/children*	Children
	GET /service/o/(uuid:orgid)/ou/	List & search
	GET /service/ou/(uuid:unitid)/	Get
	POST /service/ou/(uuid:unitid)/create	Create relation
	POST /service/ou/(uuid:unitid)/edit	Edit
	POST /service/ou/(uuid:unitid)/terminate	Terminate
	POST /service/ou/create	Create

GET /service/(**any:** *type*)**/**
uuid: *parentid***/children** Obtain the list of nested units within an organisation or an organisational unit.

Parameters

- **type** – 'o' if the parent is an organistion, and 'ou' if it's a unit.
- **parentid** (*uuid*) – The UUID of the parent.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.

Request JSON Array of Objects

- **name** (*string*) – Human-readable name of the unit.
- **user_key** (*string*) – Short, unique key identifying the unit.
- **uuid** (*uuid*) – Machine-friendly UUID of the unit.
- **child_count** (*int*) – Number of org. units nested immediately beneath the organisation.

Status Codes

- 200 OK – Whenever the organisation or unit exists and is readable.
- 404 Not Found – When no such organisation or unit exists, or the parent was of the wrong type.

Example Response:

```
[
  {
    "name": "Humanistisk fakultet",
    "user_key": "hum",
    "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e",
    "child_count": 2
  },
  {
    "name": "Samfundsvidenskabelige fakultet",
    "user_key": "samf",
    "uuid": "b688513d-11f7-4efc-b679-ab082a2055d0",
    "child_count": 0
  }
]
```

GET /service/o/(**uuid:** *orgid*)**/ou/**
Query organisational units in an organisation.

Parameters

- **orgid** (*uuid*) – UUID of the organisation to search.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.
- **start** (*int*) – Index of first unit for paging.
- **limit** (*int*) – Maximum items
- **query** (*string*) – Filter by units matching this string.

Request JSON Object

- **items** (*string*) – The returned items.
- **offset** (*string*) – Pagination offset.
- **total** (*string*) – Total number of items available on this query.

Request JSON Array of Objects

- **name** (*string*) – Human-readable name.
- **uuid** (*string*) – Machine-friendly UUID.
- **user_key** (*string*) – Short, unique key identifying the unit.

Status Codes

- 200 OK – Always.

Example Response:

```
{
  "items": [
    {
      "name": "Samfundsvidenskabelige fakultet",
      "user_key": "samf",
      "uuid": "b688513d-11f7-4efc-b679-ab082a2055d0"
    }
  ],
  "offset": 0,
  "total": 1
}
```

GET /service/ou/(**uuid:** *unitid*)**/**
Query organisational units in an organisation.

Parameters

- **unitid** (*uuid*) – UUID of the unit to retrieve.

Query Parameters

- **at** (*date*) – Current time in ISO-8601 format.

Status Codes

- 200 OK – Whenever the object exists.
- 404 Not Found – Otherwise.

Example Response:

```
{
  "name": "Afdeling for Fortidshistorik",
  "user_key": "frem",
  "uuid": "04c78fc2-72d2-4d02-b55f-807af19eac48"
  "org": {
```

```
    "name": "Aarhus Universitet",
    "user_key": "AU",
    "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62"
  },
  "org_unit_type": {
    "example": null,
    "name": "Afdeling",
    "scope": null,
    "user_key": "afd",
    "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
  },
  "parent": {
    "name": "Historisk Institut",
    "user_key": "hist",
    "uuid": "da77153e-30f3-4dc2-a611-ee912a28d8aa"
  }
}
```


POST /service/ou/(**uuid:** *unitid*)**/create**

Creates new unit relations

Status Codes

- 200 OK – Creation succeeded.

Parameters

- **employee_uuid** – The UUID of the employee.

All requests contain validity objects on the following form:

Request JSON Array of Objects

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.


```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```


Request payload contains a list of creation objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Address:

Request JSON Array of Objects

- **type** (*string*) – "address"
- **address_type** (*object*) – The type of the address, exactly as returned by returned by *GET /service/o/(uuid:orgid)/f/(facet)/*.
- **value** (*string*) – The value of the address field. Please note that as a special case, this should be a UUID for *DAR* addresses.


```
[
  {
    "value": "1234567890",
    "address_type": {
      "example": "5712345000014",
      "name": "EAN",
      "scope": "EAN",
      "user_key": "EAN",
      "uuid": "e34d4426-9845-4c72-b31e-709be85d6fa2"
    },
```

```
    "type": "address",
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

POST /service/ou/(**uuid:** *unitid*)**/edit**

Edits an organisational unit

Status Codes

- 200 OK – The edit succeeded.

Parameters

- **unitid** – The UUID of the organisational unit.

Request JSON Array of Objects

- **type** (*string*) – The type of the operation, defaulting to `org_unit`.
- **original** (*object*) – An **optional** object containing the original state of the org unit to be overwritten. If supplied, the change will modify the existing registration on the org unit object. Detailed below.
- **data** (*object*) – An object containing the changes to be made to the org unit. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

Request JSON Object

- **name** (*string*) – The name of the org unit
- **parent** (*string*) – The parent org unit
- **org_unit_type** (*string*) – The type of org unit
- **validity** (*object*) – The validities of the changes.

Validity objects are defined as such:

Request JSON Object

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.

Example Request:

```
[
  {
    "original": {
      "name": "Pandekagehuset",
      "parent": {
        "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
      },
      "org_unit_type": {
        "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
      },
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": null
      }
    },
    "data": {
```

```
      "name": "Vaffelhuset",
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
      }
    }
  }
]
```

See also *POST /service/e/(uuid:employee_uuid)/edit* for further examples for the individual types.

GET /service/ou/(uuid: *unitid*) **/history/**
Get the history of an org unit :param unitid: The UUID of the org unit

Example response:

Request JSON Array of Objects

- **from** (*string*) – When the change is active from
- **to** (*string*) – When the change is active to
- **action** (*string*) – The action performed
- **life_cycle_code** (*string*) – The type of action performed
- **user_ref** (*string*) – A reference to the user who made the change

```
[
  {
    "from": "2018-02-21T13:25:24.391793+01:00",
    "to": "infinity",
    "action": "Afslut enhed",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.343010+01:00",
    "to": "2018-02-21T13:25:24.391793+01:00",
    "action": "Rediger organisationsenhed",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.271516+01:00",
    "to": "2018-02-21T13:25:24.343010+01:00",
    "action": "Rediger organisationsenhed",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.214514+01:00",
    "to": "2018-02-21T13:25:24.271516+01:00",
    "action": "Oprettet i MO",
    "life_cycle_code": "Opstaaet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  }
]
```

POST /service/ou/(uuid: *unitid*) **/terminate**
Terminates an organisational unit from a specified date.

Status Codes

- 200 OK – The termination succeeded.

Parameters

- **unitid** – The UUID of the organisational unit to be terminated.

Request JSON Object

- **validity** (*object*) – The date on which the termination should happen, in ISO 8601.

Example Request:

```
{
  "validity": {
    "from": "2016-01-01T00:00:00+00:00"
  }
}
```

POST /service/ou/create

Creates new organisational unit

Status Codes

- 200 OK – Creation succeeded.

Example Request:

Request JSON Object

- **name** (*string*) – The name of the org unit
- **parent** (*string*) – The parent org unit
- **org_unit_type** (*string*) – The type of org unit
- **addresses** (*list*) – A list of address objects.
- **validity** (*object*) – The validity of the created object.

Validity objects are defined as such:

Request JSON Array of Objects

- **from** (*string*) – The from date, in ISO 8601.
- **to** (*string*) – The to date, in ISO 8601.

```
{
  "name": "Name",
  "parent": {
    "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
  },
  "org_unit_type": {
    "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
  },
  "valid_from": "2016-01-01T00:00:00+00:00",
  "valid_to": "2018-01-01T00:00:00+00:00"
}
```

Returns UUID of created org unit

3.11 Server-side codebase

3.11.1 Legacy service layer

This is the legacy service API retained for compatibility with the original UI.

mora.api.**create_employee_role**(*employeeid*, *role=None*)
Catch-all function for creating Employees roles

Parameters **employeeid** – Employee ID from MO. Not used.

Returns The uuid of the employee and a HTTP status code.

mora.api.**create_organisation_unit**(*orgid=None*)
Create a new org unit.

Parameters **orgid** – The UUID of the organisation (not used, but given by the frontend).

Returns JSON containing the new org unit UUID and the response status code.

mora.api.**full_hierarchy**(*orgid*)

mora.api.**get_contact_facet_properties_classes**()

mora.api.**get_contact_facet_types_classes**()

mora.api.**get_employee**(*id*)

mora.api.**get_employee_by_cpr**(*cpr_number*)

mora.api.**get_engagement_classes**(*facet*)

mora.api.**get_geographical_addresses**()

mora.api.**get_organisation**(*id*)

mora.api.**get_orgunit**(*unitid*, *orgid=None*)

mora.api.**get_orgunit_history**(*unitid*, *orgid=None*)

mora.api.**get_role**(*role*, ***kwargs*)
Get the roles of an employee or organisational unit.

Queryparam string validity Only yield entries relevant to the past, present or future.

Queryparam date effective-date Set the "current" date for the time machine.

Queryparam boolean unique Retained for compatibility with original UI.

Queryparam int t Retained for compatibility with original UI.

Parameters

- **role** (*string*) – The relevant role type, see *GET /mo/role-types/*
- **userid** (*uuid*) – Optional employee UUID.
- **unitid** (*uuid*) – Optional unit UUID.
- **orgid** (*uuid*) – Optional organisation UUID, retained for compatibility with original UI.

mora.api.**inactivate_org_unit**(*unitid*, *orgid=None*)
Inactivate an org unit.

Parameters

- **orgid** – The UUID of the organisation (not used, but given by the frontend).
- **unitid** – The UUID of the org unit.

Returns JSON containing the org unit UUID and the response status code.

mora.api.**list_classes**()

mora.api.**list_employees**()

mora.api.**list_itsystems**(*orgid=None*)
List the available IT systems.

Parameters **orgid** (*string*) – optional organisation UUID restricting the search

>jsonarr string name user-friendly name

>jsonarr string userKey unique, human-readable identifyer

>jsonarr string uuid unique, machine-friendly identifyer

Status 200 Always, even if nothing found.

Status 404 Not used.

mora.api.**list_organisations**()

mora.api.**list_orgunits**(*orgid*)

mora.api.**list_roles**()
List the supported role types.

mora.api.**move_org_unit**(*unitid*, *orgid=None*)
Move an org unit.

Parameters

- **orgid** – The UUID of the organisation (not used, but given by the frontend).
- **unitid** – The UUID of the org unit.

Returns JSON containing the org unit UUID and the response status code.

mora.api.**rename_or_retype_org_unit**(*unitid*, *orgid=None*)
Change the name or the type of an org unit.

Parameters

- **orgid** – The UUID of the organisation.
- **unitid** – The UUID of the org unit.

Returns JSON containing the org unit UUID and the response status code.

mora.api.**update_organisation_unit_location**(*orgid=None*, *unitid=None*, *locid=None*)
Add a location or contact channel or update existing ones.

Parameters

- **orgid** – The UUID of the organisation.
- **unitid** – The UUID of the org unit.
- **locid** – The UUID of the location (i.e. the address UUID).

Returns JSON containing the org unit UUID and the response status code.

mora.app.**handle_invalid_usage**(*error*)
Handles errors in case an exception is raised.

Parameters **error** – The error raised.

Returns JSON describing the problem and the apropriate status code.

mora.app.**root**()

mora.app.**send_scripts**(*path*)

mora.app.**send_styles**(*path*)

mora.app.**v2_root**(*path=''*)

3.11.2 Authentication

This section describes how to authenticate with MO. The API is work in progress.

class mora.auth.**SAMLAuth**(*assertion=None*)

mora.auth.**login**(*username*)
Attempt a login as the given user name. The internals of this login will be kept from the JavaScript by using httpOnly cookies.

Statuscode 200 The login succeeded.

Statuscode 401 The login failed.

Parameters **username** – The user ID to login as.

<json string password The password of the user.

<json boolean rememberme Whether to persist the login — currently ignored.

>json string user The name of the user.

>json string token Retained for compatibility with original UI.

>json string role Retained for compatibility with original UI.

mora.auth.**logout**(*user=None*)
Attempt to log out as the given user name.

Parameters **username** – The user ID to logout as.

Returns Nothing.

Statuscode 200 The logout succeeded — which it almost always does.

mora.cli.**load_cli**(*app*)

mora.cli.**requires_auth**(*func*)

mora.converters.addr.**autocomplete_address**(*query: str*, *orgid: str*)

mora.converters.addr.**find_address**(*query: str*, *municipality: str = None*)

mora.converters.addr.**get_address**(*addrid: str*)
Obtain an address from DAWA

mora.converters.importing.**convert**(*paths*, *include=None*, *exact=False*)

mora.converters.importing.**convert_bruger**(*obj*)

mora.converters.importing.**convert_facet**(*obj*)

mora.converters.importing.**convert_itsystem**(*obj*)

mora.converters.importing.**convert_klasse**(*obj*)

mora.converters.importing.**convert_klassifikation**(*obj*)

mora.converters.importing.**convert_organisation**(*obj*)

mora.converters.importing.**convert_organisationenhed**(*obj*)

mora.converters.importing.**convert_organisationfunktion**(*obj*)

mora.converters.importing.**load_data**(*sheets*, *exact=False*)

mora.converters.importing.**read_paths**(*paths*)

class mora.converters.meta.**Address**

    classmethod fromdict(*d*)

    classmethod fromstring(*s*)

class mora.converters.meta.**PhoneNumber**

    classmethod fromstring(*s*)

mora.converters.reading.**full_hierarchies**(*orgid: str*, *parentid: str*, *include_children=True*, ***loraparams*)

mora.converters.reading.**full_hierarchy**(*unitid: str*, *include_children=True*, ***loraparams*)

mora.converters.reading.**get_class**(*uuid*)

mora.converters.reading.**get_classes**(*facet_name: str*)

mora.converters.reading.**get_contact_channels**(*userid=None*, *orgid=None*, *unitid=None*, ***loraparams*)

mora.converters.reading.**get_contact_properties**() → list

mora.converters.reading.**get_contact_types**() → list

mora.converters.reading.**get_employees**(*uuids*, ***loraparams*)

mora.converters.reading.**get_engagements**(*orgid=None*, *unitid=None*, *userid=None*, ***loraparams*)

mora.converters.reading.**get_it_systems**(*userid: str = None*, ***loraparams*) → list

mora.converters.reading.**get_locations**(*userid: str = None*, *orgid: str = None*, *unitid: str = None*, ***loraparams*) → list

mora.converters.reading.**get_organisations**(**orgids*)

mora.converters.reading.**get_orgunit**(*unitid: str*, *include_parents=True*, ***loraparams*)

mora.converters.reading.**get_orgunits**(*unitids*, ***loraparams*)

mora.converters.reading.**get_unit_type**(*typerel*) → dict

mora.converters.reading.**list_employees**(***, *limit=1000*, *start=0*, ***loraparams*)

mora.converters.reading.**list_it_systems**(*orgid: str = None*) → list
List all it systems.

mora.converters.reading.**list_orgunits**(***, *limit=1000*, *start=0*, ***loraparams*)

mora.converters.reading.**unit_history**(*unitid*)

mora.converters.reading.**wrap_in_org**(*connector*, *orgid*, *value*, *org=None*)

mora.converters.writing.**create_contact**(*req*)

mora.converters.writing.**create_org_unit**(*req: dict*) → dict
Create org unit data to send to LoRa.

Parameters **req** – Dictionary representation of JSON request from the frontend.

Returns Dictionary representation of the org unit JSON object to send to LoRa.

mora.converters.writing.**create_update_kwargs**(*req: dict*) → dict
Pick out the necessary data from the frontend request depending on the roletype - the frontend handles location updates in a very funny way…

Parameters **req** – The frontend request.

Returns The necessary data depending on the roletype.

mora.converters.writing.**get_remaining_org_funk_fields**(*obj_paths: typing.List[typing.List[str]]*)

mora.converters.writing.**inactivate_org_funktion**(*startdate*, *enddate*)

mora.converters.writing.**inactivate_org_unit**(*startdate: str*, *enddate: str*) → dict
Inactivate an org unit.

Parameters

- **startend** – The date from which the org unit is active.

- **enddate** – The date to inactivate the org unit from.

Returns The payload JSON used to update LoRa.

mora.converters.writing.**move_org_funktion_payload**(*move_date*, *from_time*, *to_time*, *overwrite*, *org_unit_uuid*, *org-funk*)

mora.converters.writing.**move_org_unit**(*req: dict*) → dict
Move an org unit to a new parent unit.

Parameters **req** – The JSON reqeust from the frontend.

Returns The payload JSON used to update LoRa.

mora.converters.writing.**rename_org_unit**(*req: dict*) → dict
Rename an org unit.

Parameters **req** – The JSON request sent from the frontend.

Returns The payload JSON used to update LoRa.

mora.converters.writing.**retype_org_unit**(*req: dict*) → dict
Change the type or start-date of the org unit.

Parameters **req** – The JSON request sent from the frontend.

Returns The payload JSON used to update LoRa.

mora.converters.writing.**update_employee_addresses**(*emplid: str*, *roletype: str*, ***kwargs*)

mora.converters.writing.**update_org_funktion_payload**(*from_time*, *to_time*, *note*, *fields*, *original*, *payload*)

mora.converters.writing.**update_org_unit_addresses**(*unitid: str*, *roletype: str*, ***kwargs*)
Update or add an org unit address or contact channel.

Parameters

- **unitid** – The org unit UUID.
- **roletype** – The roletype (contact-channel, location, None) to use - this is handled in a funny way by the frontend!?
- **kwargs** – The required data from the frontend request.

Returns The payload to send (PUT) to LoRa.

exception mora.exceptions.**IllegalArgumentException**

mora.integrations.serviceplatformen.**get_citizen**(*cpr*)

class mora.lora.**Connector**(***defaults*)

defaults

get_date_chunks(*dates*)

is_effect_relevant(*effect*)

scope_map = {'bruger': 'organisation/bruger', 'facet': 'klassifikation/facet', 'it

validity

class mora.lora.**Scope**(*connector*, *path*)

base_path

create(*obj*, *uuid=None*)

delete(*uuid*)

fetch(***params*)

get(*uuid*, ***params*)

get_all(**, start=0, limit=1000, **params*)

get_effects(*obj*, *relevant*, *also=None*, ***params*)

paged_get(*func*, **, start=0, limit=1000, **params*)

update(*obj*, *uuid*)

mora.lora.**create**(*path*, *obj*, *uuid=None*)

mora.lora.**delete**(*path*, *uuid*)

mora.lora.**fetch**(*path*, ***params*)

mora.lora.**get**(*path*, *uuid*, ***params*)

mora.lora.**get_org_unit**(*unitid: str*) → dict
Get an org unit for in the time span from -infinity to +infinity :param unitid: the UUID of the org unit :return: the org unit

mora.lora.**update**(*path*, *obj*)

3.11.3 Address

This section describes how to interact with addresses.

class mora.service.address.**Addresses**(*scope*)

create(*id*, *req*)

edit(*id*, *req*)

get(*id*)

static has(*reg*)

mora.service.address.**address_autocomplete**(*orgid*)
Perform address autocomplete :param orgid: The UUID of the organisation

Queryparam str q A query string to be used for lookup

Queryparam boolean global Whether or not the lookup should be in the entire country, or contained to the municipality of the organisation

Example Response:

<jsonarr uuid uuid A UUID of a DAR address

<jsonarr str name A human readable name for the address

```
[
  {
    "location": {
      "uuid": "f0396d0f-ef2d-41e5-a420-b4507b26b6fa",
      "name": "Rybergsvej 1, Sønderby, 5631 Ebberup"
    }
  },
  {
    "location": {
      "uuid": "0a3f50cb-05eb-32b8-e044-0003ba298018",
      "name": "Wild Westvej 1, 9310 Vodskov"
    }
```

```
  }
]
```

mora.service.address.**get_address_class**(*c*, *addrrel*, *class_cache*)

mora.service.address.**get_one_address**(*c*, *addrrel*, *class_cache=None*)

mora.service.address.**get_relation_for**(*classobj*, *value*)

3.11.4 Associations

This section describes how to interact with employee associations.

mora.service.association.**create_association**(*employee_uuid*, *req*)

mora.service.association.**edit_association**(*employee_uuid*, *req*)

3.11.5 Common utilities

class mora.service.common.**AbstractRelationDetail**(*scope*)

create(*id: str*, *req: dict*)

edit(*id: str*, *req: dict*)

get(*objid*)

has(*registration*)

scope

mora.service.common.**FieldTuple**
alias of mora.service.common.PropTuple

class mora.service.common.**FieldTypes**
An enumeration.

ADAPTED_ZERO_TO_MANY = (2,)

ZERO_TO_MANY = (1,)

ZERO_TO_ONE = (0,)

mora.service.common.**add_bruger_history_entry**(*employee_uuid*, *note: str*)
Add a history entry to a given employee. The idea is to write an update to the employee whenever an object associated to him is created or changed, as to easily be able to get an overview of the history of the modifications to both the employee but also the employee's associated objects.

We have to make some sort of 'meaningful' change to data to be able to update the 'note' field - which for now amounts to just updating the virkning notetekst of gyldighed with a garbage value

Parameters

- **employee_uuid** – The UUID of the employee
- **note** – A note to be associated with the entry

class mora.service.common.**cache**(*func*, **args*, ***kwargs*)
combination of functools.partial & defaultdict into one

mora.service.common.**checked_get**(*mapping: typing.Dict[K, V]*, *key: K*, *default: V*, *fallback: typing.Dict[K, V] = None*, *required: bool = False*) → V

mora.service.common.**convert_reg_to_history**(*reg*)

mora.service.common.**create_bruger_payload**(*valid_from: str*, *valid_to: str*, *brugernavn: str*, *brugervendtnoegle: str*, *tilhoerer: str*, *cpr: str*)

mora.service.common.**create_organisationsenhed_payload**(*enhedsnavn: str*, *valid_from: str*, *valid_to: str*, *brugervendtnoegle: str*, *tilhoerer: str*, *enhedstype: str*, *overordnet: str*, *adresser: typing.List[dict] = None*) → dict

mora.service.common.**create_organisationsfunktion_payload**(*funktionsnavn: str, valid_from: str, valid_to: str, brugervendtnoegle: str, tilknyttedebrugere: typing.List[str], tilknyttedeorganisationer: typing.List[str], tilknyttedeenheder: typing.List[str] = None, funktionstype: str = None, opgaver: typing.List[dict] = None, adresser: typing.List[str] = None*) → dict

mora.service.common.**ensure_bounds**(*valid_from: datetime.datetime, valid_to: datetime.datetime, props: typing.List[mora.service.common.PropTuple], obj: dict, payload: dict*)

mora.service.common.**get_args_flag**(*name: str*)

Get an argument from the Flask request as a boolean flag.

A 'flag' argument is false either when not set or one of the values '0', 'false' and 'False'. Anything else is true.

mora.service.common.**get_connector**()

mora.service.common.**get_effect_from**(*effect: dict*) → datetime.datetime

mora.service.common.**get_effect_to**(*effect: dict*) → datetime.datetime

mora.service.common.**get_effect_validity**(*effect*)

mora.service.common.**get_field_value**(*field: mora.service.common.PropTuple*, *obj*)

mora.service.common.**get_obj_value**(*obj*, *path: tuple*, *filter_fn: typing.Callable = None*)

mora.service.common.**get_uuid**(*mapping: typing.Dict[K, V], fallback: typing.Dict[K, V] = None, *, key: collections.abc.Hashable = 'uuid'*) → str

mora.service.common.**get_valid_from**(*obj*, *fallback=None*) → datetime.datetime

mora.service.common.**get_valid_to**(*obj*, *fallback=None*) → datetime.datetime

mora.service.common.**get_validity_effect**(*entry*, *fallback=None*)

mora.service.common.**inactivate_old_interval**(*old_from: str*, *old_to: str*, *new_from: str*, *new_to: str*, *payload: dict*, *path: tuple*) → dict

Create 'inactivation' updates based on two sets of from/to dates :param old_from: The old 'from' time, in ISO-8601 :param old_to: The old 'to' time, in ISO-8601 :param new_from: The new 'from' time, in

ISO-8601 :param new_to: The new 'to' time, in ISO-8601 :param payload: An existing payload to add the updates to :param path: The path to where the object's 'gyldighed' is located :return: The payload with the inactivation updates added, if relevant

mora.service.common.**inactivate_org_funktion_payload**(*enddate*, *note*)

mora.service.common.**is_reg_valid**(*reg*)

Check if a given registration is valid i.e. that the registration contains a 'gyldighed' that is 'Aktiv'

Parameters **reg** – A registration object

mora.service.common.**replace_relation_value**(*relations: typing.List[dict], old_entry: dict, new_entry: dict = None*) → typing.List[dict]

mora.service.common.**set_object_value**(*obj: dict, path: tuple, val: typing.List[dict]*)

mora.service.common.**update_payload**(*valid_from: datetime.datetime, valid_to: datetime.datetime, relevant_fields: typing.List[typing.Tuple[mora.service.common.PropTuple, dict]], obj: dict, payload: dict*)

3.11.6 CPR

This section describes functionality for retrieving information about people based on their CPR number.

mora.service.cpr.**format_cpr_response**(*sp_data: dict*, *cpr: str*)

mora.service.cpr.**search_cpr**()

Search for a CPR number in Serviceplatformen and retrieve the associated information

Queryparam q The CPR no. of a person to be searched

<jsonarr string name The name of the person

<jsonarr string cpr_no The person's CPR number.

Example Response:

```
{
  "name": "John Doe",
  "cpr_no": "1234567890"
}
```

3.11.7 Details

This section describes how to interact with employee and organisational unit metadata, referred to as *details* within this API.

class mora.service.details.**DetailType**(*search*, *scope*, *relation_types*)

relation_types
Alias for field number 2

scope
Alias for field number 1

search
Alias for field number 0

mora.service.details.**get_detail**(*type*, *id*, *function*)

Obtain the list of engagements, associations, roles, etc. corresponding to a user or organisational unit. See *GET /service/(any:type)/(uuid:id)/details/* for the available list of endpoints.

Most of these endpoints are broadly similar to engagements, with the notable exception being IT systems.

All requests contain validity objects on the following form:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

Queryparam date at Current time in ISO-8601 format.

Queryparam string validity Only show *past*, *present* or *future* values – which the default being to show *present* values.

Queryparam int start Index of first item for paging.

Queryparam int limit Maximum items.

Parameters

- **type** – 'ou' for querying a unit; 'e' for querying an employee.
- **id** (*uuid*) – The UUID to query, i.e. the ID of the employee or unit.
- **function** – See *GET /service/(any:type)/(uuid:id)/details/* for the available values for this field.

Status 200 Always.

Example engagement response:

<jsonarr object job_function See *GET /service/o/(uuid:orgid)/f/(facet)/*.

<jsonarr object type See *GET /service/o/(uuid:orgid)/f/(facet)/*.

<jsonarr object org_unit See *GET /service/o/(uuid:orgid)/f/(facet)/*.

<jsonarr string uuid Machine-friendly UUID.

<jsonarr string validity The validity times of the object.


```
[
      {
          "job_function": {
              "example": null,
              "name": "Fakultet",
              "scope": null,
              "user_key": "fak",
              "uuid": "4311e351-6a3c-4e7e-ae60-8a3b2938fbd6"
          },
          "org_unit": {
              "name": "Humanistisk fakultet",
              "user_key": "hum",
              "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e"
          },
          "person": {
              "name": "Anders And",
              "uuid": "53181ed2-f1de-4c4a-a8fd-ab358c2c454a"
          },
          "engagement_type": {
              "example": null,
              "name": "Afdeling",
              "scope": null,
              "user_key": "afd",
              "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
          },
```

```
            "uuid": "d000591f-8705-4324-897a-075e3623f37b",
            "validity": {
                "from": "2017-01-01T00:00:00+01:00",
                "to": null
            },
        }
    ]
```


Example association response:


```
[
  {
    "address": {
      "href": "https://www.openstreetmap.org/"
              "?mlon=12.57924839&mlat=55.68113676&zoom=16",
      "name": "Pilestræde 43, 3., 1112 København K",
      "value": "0a3f50a0-23c9-32b8-e044-0003ba298018"
    },
    "address_type": {
      "example": "<UUID>",
      "name": "Adresse",
      "scope": "DAR",
      "user_key": "Adresse",
      "uuid": "4e337d8e-1fd2-4449-8110-e0c8a22958ed"
    },
    "association_type": {
      "example": null,
      "name": "Medlem",
      "scope": null,
      "user_key": "medl",
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "job_function": {
      "example": null,
      "name": "Hund",
      "scope": null,
      "user_key": "hund",
      "uuid": "c2b23c43-87c6-48bb-a99c-53396bfa99fb"
    },
    "org_unit": {
      "name": "Humanistisk fakultet",
      "user_key": "hum",
      "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e"
    },
    "person": {
      "name": "Fedtmule",
      "uuid": "6ee24785-ee9a-4502-81c2-7697009c9053"
    },
    "uuid": "30cd25e1-b21d-46fe-b299-1c1265e9be66",
    "validity": {
      "from": "2017-01-01T00:00:00+01:00",
      "to": "2018-01-01T00:00:00+01:00"
    }
  }
]
```


Example IT response:


```
[
  {
    "name": "Active Directory",
    "user_name": "Fedtmule",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb",
```

```
    "validity": {
      "from": "2002-02-14T00:00:00+01:00",
      "to": null
    }
  }
]
```

Example address response:

```
[
  {
    "name": "Christiansborg Slotsplads 1, 1218 København K",
    "value": "bae093df-3b06-4f23-90a8-92eabedb3622"
    "href": "https://www.openstreetmap.org/"
        "?mlon=12.58176945&mlat=55.67563739&zoom=16",
    "address_type": {
      "scope": "DAR"
    },
    "validity": {
      "from": "2002-02-14T00:00:00+01:00",
      "to": null
    },
  },
  {
    "name": "goofy@example.com",
    "href": "mailto:goofy@example.com",
    "value": "urn:mailto:goofy@example.com"
    "address_type": {
      "example": "test@example.com",
      "name": "Emailadresse",
      "scope": "EMAIL",
      "user_key": "Email",
      "uuid": "c78eb6f7-8a9e-40b3-ac80-36b9f371c3e0"
    },
    "validity": {
      "from": "2002-02-14T00:00:00+01:00",
      "to": null
    },
  },
  {
    "name": "goofy@example.com",
    "href": "mailto:goofy@example.com",
    "value": "urn:mailto:goofy@example.com"
    "address_type": {
      "example": "test@example.com",
      "name": "Emailadresse",
      "scope": "EMAIL",
      "user_key": "Email",
      "uuid": "c78eb6f7-8a9e-40b3-ac80-36b9f371c3e0"
    },
    "validity": {
      "from": "2002-02-14T00:00:00+01:00",
      "to": null
    },
  }
]
```

Example org_unit response:

An array of objects as returned by *GET /service/ou/(uuid:unitid)/*.

```
[
  {
```

```
    "name": "Afdeling for Fortidshistorik",
    "user_key": "frem",
    "uuid": "04c78fc2-72d2-4d02-b55f-807af19eac48"
    "org": {
      "name": "Aarhus Universitet",
      "user_key": "AU",
      "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62"
    },
    "org_unit_type": {
      "example": null,
      "name": "Afdeling",
      "scope": null,
      "user_key": "afd",
      "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
    },
    "parent": {
      "name": "Historisk Institut",
      "user_key": "hist",
      "uuid": "da77153e-30f3-4dc2-a611-ee912a28d8aa"
    },
    "validity": {
      "from": "2018-01-01T00:00:00+01:00",
      "to": "2019-01-01T00:00:00+01:00"
    }
  }
]
```


Example manager response:

mora.service.details.**list_details**(*type*, *id*)
List the available 'detail' types under this entry.

Example response:


```
{
  "address": false,
  "association": false,
  "engagement": true,
  "it": false,
  "leave": true,
  "manager": false,
  "role": false
}
```


The value above informs you that at least one entry exists for each of 'engagement' and 'leave' either in the past, present or future.

3.11.8 Employees

This section describes how to interact with employees.

mora.service.employee.**create_employee**()
Create a new employee

Statuscode 200 Creation succeeded.

Example Request:

<json string name The name of the employee

<json string cpr_no The CPR no of the employee

<json object org The organisation with which the employee is associated

```
{
  "name": "Name Name",
  "cpr_no": "1234567890",
  "org": {
    "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
  }
}
```

Returns UUID of created employee

mora.service.employee.**create_employee_relation**(*employee_uuid*)

Creates new employee relations

Statuscode 200 Creation succeeded.

Parameters **employee_uuid** – The UUID of the employee.

All requests contain validity objects on the following form:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

Request payload contains a list of creation objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Engagement:

<jsonarr string type "engagement"

<jsonarr string org_unit The associated org unit

<jsonarr string job_function The job function of the association

<jsonarr string engagement_type The engagement type

<jsonarr object validity The validities of the created object.

```
[
  {
    "type": "engagement",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "job_function": {
      "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
    },
    "engagement_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Association:

<jsonarr string type "association"

<jsonarr string org_unit The associated org unit

<jsonarr string job_function The job function of the association

<jsonarr string association_type The association type

<jsonarr string location The associated location.

<jsonarr object validity The validities of the created object.

```
[
  {
    "type": "association",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "job_function": {
      "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
    },
    "association_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "location": {
      "uuid": "89faa44c-f37a-4e4a-9cd8-b25f67cfd7bc"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    },
  }
]
```

IT system:

<json string type `"it"`

<json object itsystem The IT system to create a relation to, as returned by *GET /service/o/(uuid:orgid)/it/*. The only mandatory field is `uuid`.

```
[
    {
        "type": "it",
        "itsystem": {
            "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb"
        },
        "validity": {
            "from": "2017-12-01T00:00:00+01",
            "to": null
        }
    }
]
```

Role:

<jsonarr string type "role"

<jsonarr string org_unit The associated org unit

<jsonarr string role_type The role type

<jsonarr object validity The validities of the created object.

```
[
  {
    "type": "role",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
```

```
    },
    "role_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Manager:

<jsonarr string type **"manager"**

<jsonarr string org_unit The associated org unit

<jsonarr string manager_type The manager type

<jsonarr string responsibility The manager responsibility

<jsonarr string manager_level The manager level

<jsonarr object validity The validities of the created object.

```
[
  {
    "type": "manager",
    "org_unit": {
      "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
    },
    "manager_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "responsibility": {
      "uuid": "e6b24f90-b056-433b-ad65-e6ab95d25826"
    },
    "manager_level": {
      "uuid": "f17f2d60-9750-4577-a367-8a5f065b63fa"
    },
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

Leave:

<jsonarr string type **"leave"**

<jsonarr string leave_type The leave type

<jsonarr object validity The validities of the created object.

```
[
  {
    "type": "leave",
    "leave_type": {
      "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
    },
    "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
    },
```

```
  }
]
```

Address:

<jsonarr string type `"address"`

<jsonarr object address_type The type of the address, exactly as returned by returned by *GET /service/o/(uuid:orgid)/f/(facet)/*.

<jsonarr string address The value of the address field. Please note that as a special case, this should be a UUID for *DAR* addresses.

```
[
  {
    "value": "1234567890",
    "address_type": {
      "example": "5712345000014",
      "name": "EAN",
      "scope": "EAN",
      "user_key": "EAN",
      "uuid": "e34d4426-9845-4c72-b31e-709be85d6fa2"
    },
    "type": "address",
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

mora.service.employee.**edit_employee**(*employee_uuid*)

Edits an employee

Statuscode 200 The edit succeeded.

All requests contain validity objects on the following form:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00"
}
```

Request payload contains a list of edit objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Engagement:

Parameters **employee_uuid** – The UUID of the employee.

<json string type "engagement"

<json string uuid The UUID of the engagement,

<json object original An **optional** object containing the original state of the engagement to be overwritten. If supplied, the change will modify the existing registration on the engagement object. Detailed below.

<json object data An object containing the changes to be made to the engagement. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string org_unit The associated org unit

<jsonarr string job_function The job function of the association

<jsonarr string engagement_type The engagement type

<jsonarr object validity The validities of the changes.

```
[
  {
    "type": "engagement",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2018-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      },
      "engagement_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      }
    },
    "data": {
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": "2019-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      }
    }
  }
]
```

Association:

Parameters **employee_uuid** – The UUID of the employee.

<json string type "association"

<json string uuid The UUID of the association,

<json object original An **optional** object containing the original state of the association to be overwritten. If supplied, the change will modify the existing registration on the association object. Detailed below.

<json object data An object containing the changes to be made to the association. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string org_unit The associated org unit

<jsonarr string job_function The job function of the association

<jsonarr string association_type The association type

<jsonarr string location The associated location.

<jsonarr object validity The validities of the changes.

```
[
  {
    "type": "association",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      },
      "association_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      },
      "location": {
        "uuid": "89faa44c-f37a-4e4a-9cd8-b25f67cfd7bc"
      }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "job_function": {
        "uuid": "5b56432c-f289-4d81-a328-b878ea0a4e1b"
      }
    }
  }
]
```

IT system:

Parameters **employee_uuid** – The UUID of the employee.

<json string type `"it"`

<json string uuid The UUID of the IT system,

<json object original An **optional** object containing the original state of the role to be overwritten. If supplied, the change will modify the existing registration on the role object. Detailed below.

<json object data An object containing the changes to be made to the role. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string uuid Change the IT system to another.

```
[
  {
    "type": "it",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb",
    "original": {
      "name": "Active Directory",
      "user_name": "Fedtmule",
      "uuid": "00000000-0000-0000-0000-000000000000",
      "validity": {
        "from": "2002-02-14T00:00:00+01:00",
        "to": null
      }
```

```
    },
    "data": {
      "uuid": "11111111-1111-1111-1111-111111111111",
      "validity": {
        "to": "2020-01-01T00:00:00+01:00"
      }
    }
  }
]
```

Role:

Parameters **employee_uuid** – The UUID of the employee.

<json string type "role"

<json string uuid The UUID of the role,

<json object original An **optional** object containing the original state of the role to be overwritten. If supplied, the change will modify the existing registration on the role object. Detailed below.

<json object data An object containing the changes to be made to the role. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string org_unit The associated org unit

<jsonarr string role_type The role type

<jsonarr string location The associated location.

<jsonarr object validity The validities of the changes.

```
[
  {
    "type": "role",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "role_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      },
      "org_unit": {
        "uuid": "04f73c63-1e01-4529-af2b-dee36f7c83cb"
      }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "role_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```

Leave:

Parameters **employee_uuid** – The UUID of the employee.

<json string type "leave"

<json string uuid The UUID of the leave,

<json object original An **optional** object containing the original state of the leave to be overwritten. If supplied, the change will modify the existing registration on the leave object. Detailed below.

<json object data An object containing the changes to be made to the leave. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string leave_type The leave type

<jsonarr object validity The validities of the changes.

```
[
  {
    "type": "leave",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2018-01-01T00:00:00+00:00"
      },
      "leave_type": {
        "uuid": "743a6448-2b0b-48cf-8a2e-bf938a6181ee"
      }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "leave_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```

Manager:

Parameters **employee_uuid** – The UUID of the employee.

<json string type "manager"

<json string uuid The UUID of the manager,

<json object original An **optional** object containing the original state of the leave to be overwritten. If supplied, the change will modify the existing registration on the leave object. Detailed below.

<json object data An object containing the changes to be made to the leave. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<jsonarr string manager_type The manager type

<jsonarr string org_unit The associated org unit

<jsonarr string manager_type The manager type

<jsonarr string responsibility The manager responsibility

<jsonarr string manager_level The manager level

<jsonarr object validity The validities of the changes.


```
[
  {
    "type": "manager",
    "uuid": "de9e7513-1934-481f-b8c8-45336387e9cb",
    "original": {
        "org_unit": {
          "uuid": "a30f5f68-9c0d-44e9-afc9-04e58f52dfec"
        },
        "manager_type": {
          "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
        },
        "responsibility": {
          "uuid": "e6b24f90-b056-433b-ad65-e6ab95d25826"
        },
        "manager_level": {
          "uuid": "f17f2d60-9750-4577-a367-8a5f065b63fa"
        },
        "validity": {
            "from": "2016-01-01T00:00:00+00:00",
            "to": "2018-01-01T00:00:00+00:00"
        }
    },
    "data": {
      "validity": {
          "from": "2016-01-01T00:00:00+00:00",
          "to": "2019-01-01T00:00:00+00:00"
      },
      "manager_type": {
        "uuid": "eee27f47-8355-4ae2-b223-0ee0fdad81be"
      }
    }
  }
]
```


mora.service.employee.**get_employee**(*id*)

Retrieve an employee.

Queryparam date at Current time in ISO-8601 format.

<json string name Human-readable name.

<json string uuid Machine-friendly UUID.

<json object org The organisation that this employee belongs to, as yielded by *GET /service/o/*.

<json string cpr_no CPR number of for the corresponding person. Please note that this is the only means for obtaining the CPR number; due to confidentiality requirements, all other end points omit it.

Status 200 Whenever the user ID is valid and corresponds to an existing user.

Status 404 Otherwise.

Example Response:


```
{
  "cpr_no": "1011101010",
  "name": "Hans Bruger",
  "uuid": "9917e91c-e3ee-41bf-9a60-b024c23b5fe3",
  "org": {
    "name": "Magenta ApS",
    "user_key": "Magenta ApS",
    "uuid": "8efbd074-ad2a-4e6a-afec-1d0b1891f566"
```

```
  }
}
```

mora.service.employee.**get_employee_history**(*employee_uuid*)

Get the history of an employee :param employee_uuid: The UUID of the employee

Example response:

<jsonarr string from When the change is active from

<jsonarr string to When the change is active to

<jsonarr string action The action performed

<jsonarr string life_cycle_code The type of action performed

<jsonarr string user_ref A reference to the user who made the change

```
[
  {
    "from": "2018-02-21T11:27:20.909206+01:00",
    "to": "infinity",
    "action": "Opret orlov",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T11:27:20.803682+01:00",
    "to": "2018-02-21T11:27:20.909206+01:00",
    "action": "Rediger engagement",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T11:27:20.619990+01:00",
    "to": "2018-02-21T11:27:20.803682+01:00",
    "action": null,
    "life_cycle_code": "Importeret",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  }
]
```

mora.service.employee.**get_one_employee**(*c*, *userid*, *user=None*, *full=False*)

mora.service.employee.**list_employees**(*orgid*)

Query employees in an organisation.

Parameters **orgid** (*uuid*) – UUID of the organisation to search.

Queryparam date at Current time in ISO-8601 format.

Queryparam int start Index of first unit for paging.

Queryparam int limit Maximum items

Queryparam string query Filter by employees matching this string. Please note that this only applies to attributes of the user, not the relations or engagements they have.

<json string items The returned items.

<json string offset Pagination offset.

<json string total Total number of items available on this query.

<jsonarr string name Human-readable name.

<jsonarr string uuid Machine-friendly UUID.

Status 200 Always.

Example Response:

```
{
  "items": [
    {
      "name": "Hans Bruger",
      "uuid": "9917e91c-e3ee-41bf-9a60-b024c23b5fe3"
    },
    {
      "name": "Joe User",
      "uuid": "cd2dcfad-6d34-4553-9fee-a7023139a9e8"
    }
  ],
  "offset": 0,
  "total": 1
}
```

mora.service.employee.**terminate_employee**(*employee_uuid*)

Terminates an employee and all of his roles from a specified date.

Statuscode 200 The termination succeeded.

Parameters **employee_uuid** – The UUID of the employee to be terminated.

<json string valid_from The date on which the termination should happen, in ISO 8601.

Example Request:

```
{
  "validity": {
    "from": "2016-01-01T00:00:00+00:00"
  }
}
```

3.11.9 Engagements

This section describes how to interact with engagements linking employees and organisational units.

mora.service.engagement.**create_engagement**(*employee_uuid*, *req*)

mora.service.engagement.**edit_engagement**(*employee_uuid*, *req*)

3.11.10 Facets

This sections describes how to interact with facets, i.e. the types of objects.

mora.service.facet.**get_classes**(*orgid: uuid.UUID*, *facet: str*)

List classes available in the given facet.

Parameters

- **orgid** (*uuid*) – Restrict search to this organisation.
- **facet** (*string*) – One of the facet names listed by *GET /service/o/(uuid:orgid)/f/*

Queryparam int start Index of first item for paging.

Queryparam int limit Maximum items.

>jsonarr string name Human-readable name.

>jsonarr string uuid Machine-friendly UUID.

>jsonarr string user_key Short, unique key.

>jsonarr string example An example value. In most cases the value is meant to be presented to the user as an aid.

>jsonarr string scope A representation of the type of value, see the table below for details.

Status 200 On success.

Status 404 Whenever the facet isn't found.

Scopes:

Key	Description
"DAR"	UUID of a DAR address, as found through the API. Please note that this requires performing separate calls to convert this value to and from human-readable strings.
"EMAIL"	An email address, as specified by RFC 5322#section-3.4.
"INTEGER"	An integral number.
"PHONE"	A phone number.
"TEXT"	Arbitrary text.
"WWW"	An HTTP or HTTPS URL, as specified by RFC 1738.

Example Response:


```
{
  "name": "address_type",
  "path":
    "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/address_type/",
  "user_key": "Adressetype",
  "uuid": "e337bab4-635f-49ce-aa31-b44047a43aa1",
  "data": {
    "items": [
      {
        "example": "http://www.korsbaek.dk/",
        "name": "Hjemmeside",
        "scope": "WWW",
        "user_key": "URL",
        "uuid": "160ecaed-50b0-4800-bebc-0d0289a4f624"
      },
      {
        "example": "<UUID>",
        "name": "Lokation",
        "scope": "DAR",
        "user_key": "AdresseLokation",
        "uuid": "031f93c3-6bab-462e-a998-87cad6db3128"
      },
      {
        "example": "Mandag 10:00-12:00 Tirsdag 14:00-16:00",
        "name": "Åbningstid, telefon",
        "scope": "TEXT",
        "user_key": "Åbningstid Telefon",
        "uuid": "0836ffbf-3b3e-410f-8cbf-face7e6844ef"
      }
    ],
    "offset": 0,
    "total": 3
  }
}
```


mora.service.facet.**get_one_class**(*c*, *classid*, *clazz=None*)

mora.service.facet.**get_one_facet**(*c*, *facetid*, *facet_name*, *orgid*, *facet=None*, *data=None*)

mora.service.facet.**list_facets**(*orgid*)
List the facet types available in a given organisation.

Parameters **orgid** (*uuid*) – Restrict search to this organisation.

>jsonarr string name The facet name.

>jsonarr string path The location on the web server.

>jsonarr string desc Human readable description.

Status 200 Always.

Example Response:

```
[
  {
    "name": "address",
    "path": "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/address/",
    "user_key": "Adressetype",
    "uuid": "e337bab4-635f-49ce-aa31-b44047a43aa1"
  },
  {
    "name": "ou",
    "path": "/service/o/456362c4-0ee4-4e5e-a72c-751239745e62/f/ou/",
    "user_key": "Enhedstype",
    "uuid": "fc917e7c-fc3b-47c2-8aa5-a0383342a280"
  }
]
```

3.11.11 IT Systems

This section describes how to interact with IT systems.

class mora.service.itsystem.**ITSystems**(*scope*)

create(*id*, *req*)

edit(*id*, *req*)

get(*id*)
Obtain the list of engagements corresponding to a user.

Queryparam date at Current time in ISO-8601 format.

Queryparam string validity Only show *past*, *present* or *future* values – which the default being to show *present* values.

Parameters **id** (*uuid*) – The UUID to query, i.e. the ID of the employee or unit.

All requests contain validity objects on the following form:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

<jsonarr string name The name of the IT system in question.

<jsonarr string user_name The user name on the IT system, sort of.

<jsonarr string uuid Machine-friendly UUID.

<jsonarr string validity The validity times of the object.

Status 200 Always.

Example response:

```
[
  {
    "name": "Lokal Rammearkitektur",
    "user_name": "Fedtmule",
    "uuid": "0872fb72-926d-4c5c-a063-ff800b8ee697",
    "validity": {
        "from": "2016-01-01T00:00:00+01:00",
        "to": "2018-01-01T00:00:00+01:00"
    },
  },
  {
    "name": "Active Directory",
    "user_name": "Fedtmule",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb",
    "validity": {
        "from": "2002-02-14T00:00:00+01:00",
        "to": null
    },
  }
]
```

static get_relation_for(*value*, *start*, *end*)

has(*reg*)

mora.service.itsystem.**list_it_systems**(*orgid: uuid.UUID*)

List the IT systems available within the given organisation.

Parameters **orgid** (*uuid*) – Restrict search to this organisation.

>jsonarr string uuid The universally unique identifier of the system.

>jsonarr string name The name of the system.

>jsonarr string type The type of the system.

>jsonarr string user_key A human-readable unique key for the system.

Status 200 Always.

Example Response:

```
[
  {
    "name": "Lokal Rammearkitektur",
    "type": null,
    "user_key": "LoRa",
    "uuid": "0872fb72-926d-4c5c-a063-ff800b8ee697"
  },
  {
    "name": "Active Directory",
    "type": null,
    "user_key": "AD",
    "uuid": "59c135c9-2b15-41cc-97c8-b5dff7180beb"
  }
]
```

3.11.12 Leave

This section describes how to interact with employee leave.

mora.service.leave.**create_leave**(*employee_uuid*, *req*)

mora.service.leave.**edit_leave**(*employee_uuid*, *req*)

3.11.13 Manager

This section describes how to interact with employee manager roles.

mora.service.manager.**create_manager**(*employee_uuid*, *req*)

mora.service.manager.**edit_manager**(*employee_uuid*, *req*)

3.11.14 Organisation

This section describes how to interact with organisations.

mora.service.org.**get_one_organisation**(*c*, *orgid*, *org=None*)

mora.service.org.**get_organisation**(*orgid*)

Obtain the initial level of an organisation.

Queryparam date at Current time in ISO-8601 format.

<json string name Human-readable name of the organisation.

<json string user_key Short, unique key identifying the unit.

<json string uuid Machine-friendly UUID of the organisation.

<json int child_count Number of org. units nested immediately beneath the organisation.

<json int person_count Amount of people belonging to this organisation.

<json int unit_count Amount of units belonging to this organisation.

<json int employment_count Amount of employments in this organisation?

Status 200 Whenever the organisation exists and is readable.

Status 404 When no such organisation exists.

Example Response:

```
{
  "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62",
  "name": "Aarhus Universitet",
  "user_key": "AU",
  "person_count": 2,
  "child_count": 1,
  "unit_count": 6,
  "employment_count": 1,
  "association_count": 1,
  "leave_count": 1,
  "role_count": 1,
  "manager_count": 1
}
```

mora.service.org.**list_organisations**()

List displayable organisations. We consider anything that has *at least one* root unit 'displayable'.

Queryparam date at Current time in ISO-8601 format.

<jsonarr string name Human-readable name of the organisation.

<jsonarr string user_key Short, unique key identifying the unit.

<jsonarr string uuid Machine-friendly UUID of the organisation.

Status 200 Always.

Example Response:

```
[
  {
    "name": "Aarhus Kommune",
    "user_key": "AARHUS",
    "uuid": "59141156-ed0b-457c-9535-884447c5220b"
  },
  {
    "name": "Ballerup Kommune",
    "user_key": "BALLERUP",
    "uuid": "3a87187c-f25a-40a1-8d42-312b2e2b43bd"
  }
]
```

3.11.15 Organisational units

This section describes how to interact with organisational units.

class mora.service.orgunit.**OrgUnit**(*scope*)

create(*id*, *req*)

edit(*unitid*, *req*)

get(*objid*)

has(*reg*)

class mora.service.orgunit.**UnitDetails**
An enumeration.

FULL = 2

MINIMAL = 0

NCHILDREN = 1

mora.service.orgunit.**create_org_unit**()
Creates new organisational unit

Statuscode 200 Creation succeeded.

Example Request:

<json string name The name of the org unit

<json string parent The parent org unit

<json string org_unit_type The type of org unit

<json list addresses A list of address objects.

<json object validity The validity of the created object.

Validity objects are defined as such:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "name": "Name",
  "parent": {
    "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
  },
  "org_unit_type": {
    "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
  },
  "valid_from": "2016-01-01T00:00:00+00:00",
  "valid_to": "2018-01-01T00:00:00+00:00"
}
```

Returns UUID of created org unit

mora.service.orgunit.**create_org_unit_relation**(*unitid*)

Creates new unit relations

Statuscode 200 Creation succeeded.

Parameters **employee_uuid** – The UUID of the employee.

All requests contain validity objects on the following form:

<jsonarr string from The from date, in ISO 8601.

<jsonarr string to The to date, in ISO 8601.

```
{
  "from": "2016-01-01T00:00:00+00:00",
  "to": "2018-01-01T00:00:00+00:00",
}
```

Request payload contains a list of creation objects, each differentiated by the attribute 'type'. Each of these object types are detailed below:

Address:

<jsonarr string type `"address"`

<jsonarr object address_type The type of the address, exactly as returned by returned by *GET /service/o/(uuid:orgid)/f/(facet)/*.

<jsonarr string value The value of the address field. Please note that as a special case, this should be a UUID for *DAR* addresses.

```
[
  {
    "value": "1234567890",
    "address_type": {
      "example": "5712345000014",
      "name": "EAN",
      "scope": "EAN",
      "user_key": "EAN",
      "uuid": "e34d4426-9845-4c72-b31e-709be85d6fa2"
    },
    "type": "address",
    "validity": {
      "from": "2016-01-01T00:00:00+00:00",
      "to": "2018-01-01T00:00:00+00:00"
    }
  }
]
```

mora.service.orgunit.**edit_org_unit**(*unitid*)

Edits an organisational unit

Statuscode 200 The edit succeeded.

Parameters **unitid** – The UUID of the organisational unit.

<jsonarr string type The type of the operation, defaulting to `org_unit`.

<jsonarr object original An **optional** object containing the original state of the org unit to be overwritten. If supplied, the change will modify the existing registration on the org unit object. Detailed below.

<jsonarr object data An object containing the changes to be made to the org unit. Detailed below.

The **original** and **data** objects follow the same structure. Every field in **original** is required, whereas **data** only needs to contain the fields that need to change along with the validity dates.

<json string name The name of the org unit

<json string parent The parent org unit

<json string org_unit_type The type of org unit

<json object validity The validities of the changes.

Validity objects are defined as such:

<json string from The from date, in ISO 8601.

<json string to The to date, in ISO 8601.

Example Request:

```
[
  {
    "original": {
      "name": "Pandekagehuset",
      "parent": {
        "uuid": "62ec821f-4179-4758-bfdf-134529d186e9"
      },
      "org_unit_type": {
        "uuid": "3ef81e52-0deb-487d-9d0e-a69bbe0277d8"
      },
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
        "to": null
      }
    },
    "data": {
      "name": "Vaffelhuset",
      "validity": {
        "from": "2016-01-01T00:00:00+00:00",
      }
    }
  }
]
```

See also *`POST /service/e/(uuid:employee_uuid)/edit`* for further examples for the individual types.

`mora.service.orgunit.`**`get_children`**(*type*, *parentid*)

Obtain the list of nested units within an organisation or an organisational unit.

Parameters

- **`type`** – 'o' if the parent is an organistion, and 'ou' if it's a unit.
- **`parentid`** (*`uuid`*) – The UUID of the parent.

Queryparam date at Current time in ISO-8601 format.

<jsonarr string name Human-readable name of the unit.

<jsonarr string user_key Short, unique key identifying the unit.

<jsonarr uuid uuid Machine-friendly UUID of the unit.

<jsonarr int child_count Number of org. units nested immediately beneath the organisation.

Status 200 Whenever the organisation or unit exists and is readable.

Status 404 When no such organisation or unit exists, or the parent was of the wrong type.

Example Response:

```
[
  {
    "name": "Humanistisk fakultet",
    "user_key": "hum",
    "uuid": "9d07123e-47ac-4a9a-88c8-da82e3a4bc9e",
    "child_count": 2
  },
  {
    "name": "Samfundsvidenskabelige fakultet",
    "user_key": "samf",
    "uuid": "b688513d-11f7-4efc-b679-ab082a2055d0",
    "child_count": 0
  }
]
```

mora.service.orgunit.**get_one_orgunit**(*c*, *unitid*, *unit=None*, *details=<UnitDetails.NCHILDREN: 1>*, *validity=None*) → dict

Internal API for returning one organisation unit.

mora.service.orgunit.**get_org_unit_history**(*unitid*)

Get the history of an org unit :param unitid: The UUID of the org unit

Example response:

<jsonarr string from When the change is active from

<jsonarr string to When the change is active to

<jsonarr string action The action performed

<jsonarr string life_cycle_code The type of action performed

<jsonarr string user_ref A reference to the user who made the change

```
[
  {
    "from": "2018-02-21T13:25:24.391793+01:00",
    "to": "infinity",
    "action": "Afslut enhed",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.343010+01:00",
    "to": "2018-02-21T13:25:24.391793+01:00",
    "action": "Rediger organisationsenhed",
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.271516+01:00",
    "to": "2018-02-21T13:25:24.343010+01:00",
    "action": "Rediger organisationsenhed",
```

```
    "life_cycle_code": "Rettet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  },
  {
    "from": "2018-02-21T13:25:24.214514+01:00",
    "to": "2018-02-21T13:25:24.271516+01:00",
    "action": "Oprettet i MO",
    "life_cycle_code": "Opstaaet",
    "user_ref": "42c432e8-9c4a-11e6-9f62-873cf34a735f"
  }
]
```


mora.service.orgunit.**get_orgunit**(*unitid*)

Query organisational units in an organisation.

Parameters **unitid** (*uuid*) – UUID of the unit to retrieve.

Queryparam date at Current time in ISO-8601 format.

Status 200 Whenever the object exists.

Status 404 Otherwise.

Example Response:


```
{
  "name": "Afdeling for Fortidshistorik",
  "user_key": "frem",
  "uuid": "04c78fc2-72d2-4d02-b55f-807af19eac48"
  "org": {
    "name": "Aarhus Universitet",
    "user_key": "AU",
    "uuid": "456362c4-0ee4-4e5e-a72c-751239745e62"
  },
  "org_unit_type": {
    "example": null,
    "name": "Afdeling",
    "scope": null,
    "user_key": "afd",
    "uuid": "32547559-cfc1-4d97-94c6-70b192eff825"
  },
  "parent": {
    "name": "Historisk Institut",
    "user_key": "hist",
    "uuid": "da77153e-30f3-4dc2-a611-ee912a28d8aa"
  }
}
```


mora.service.orgunit.**list_orgunits**(*orgid*)

Query organisational units in an organisation.

Parameters **orgid** (*uuid*) – UUID of the organisation to search.

Queryparam date at Current time in ISO-8601 format.

Queryparam int start Index of first unit for paging.

Queryparam int limit Maximum items

Queryparam string query Filter by units matching this string.

<json string items The returned items.

<json string offset Pagination offset.

<json string total Total number of items available on this query.

<jsonarr string name Human-readable name.

<jsonarr string uuid Machine-friendly UUID.

<jsonarr string user_key Short, unique key identifying the unit.

Status 200 Always.

Example Response:

```
{
  "items": [
    {
      "name": "Samfundsvidenskabelige fakultet",
      "user_key": "samf",
      "uuid": "b688513d-11f7-4efc-b679-ab082a2055d0"
    }
  ],
  "offset": 0,
  "total": 1
}
```

mora.service.orgunit.**terminate_org_unit**(*unitid*)
Terminates an organisational unit from a specified date.

Statuscode 200 The termination succeeded.

Parameters **unitid** – The UUID of the organisational unit to be terminated.

<json object validity The date on which the termination should happen, in ISO 8601.

Example Request:

```
{
  "validity": {
    "from": "2016-01-01T00:00:00+00:00"
  }
}
```

3.11.16 Roles

This section describes how to interact with employee roles.

mora.service.role.**create_role**(*employee_uuid*, *req*)

mora.service.role.**edit_role**(*employee_uuid*, *req*)

mora.tokens.**get_token**(*username*, *passwd*, *raw=False*, *verbose=False*, *insecure=None*)
Request a SAML authentication token from the given host and endpoint.

Windows Server typically returns a 500 Internal Server Error on authentication errors; this function simply raises a httplib.HTTPError in these cases. In other cases, it returns a KeyError. WSO2 tends to yield more meaningful errors.

mora.util.**do_ranges_overlap**(*first_start*, *first_end*, *second_start*, *second_end*)

mora.util.**from_iso_time**(*s*)

mora.util.**is_cpr_number**(*v*)

mora.util.**is_urn**(*v*)

mora.util.**is_uuid**(*v*)

mora.util.**now**() → datetime.datetime
Get the current time, localized to the current time zone.

mora.util.**parsedatetime**(*s: str*) → datetime.datetime

mora.util.**restrictargs**(**allowed: str*, *required: typing.Iterable[str] = []*)
Function decorator for checking and verifying Flask request arguments

If any argument other than those listed is set and has a value, the function logs an error and return HTTP 501.

mora.util.**splitlist**(*xs*, *size*)

mora.util.**to_frontend_time**(*s*)

mora.util.**to_iso_time**(*s*)
Return an ISO 8601 string representing the time and date given by *s*.

We always localise this to our 'default' timezone, since LoRA might be running under something silly such as UTC.

mora.util.**to_lora_time**(*s*)

mora.util.**today**() → datetime.datetime
Get midnight of current date, localized to the current time zone.

mora.util.**uniqueify**(*xs*)
return the contents of xs as a list, but stable

mora.util.**unparsedate**(*d: datetime.date*) → str

mora.util.**update_config**(*mapping*, *config_path*, *allow_environment=True*)
load the JSON configuration at the given path

We disregard all entries in the configuration that lack a default within the mapping.

mora.validator.**is_candidate_parent_valid**(*unitid: str*, *req: dict*) → bool
For moving an org unit. Check if the candidate parent is in the subtree of the org unit itself. Note: it is (and should be) allowed to move an org unit to its own parent - since it can be moved back and forth on different dates.

Parameters

- **unitid** – The UUID of the current org unit.
- **req** – The frontend request.

Returns True if the candidate parent is valid and False otherwise.

mora.validator.**is_create_org_unit_request_valid**(*req: dict*) → bool
Check if the create org unit request is valid.

Parameters **req** – The frontend request.

Returns True if the request is valid and false otherwise.

mora.validator.**is_inactivation_date_valid**(*unitid: str*, *end_date: str*) → bool
Check if the inactivation date is valid.

Parameters

- **unitid** – The UUID of the org unit.
- **end_date** – The candidate end-date.

Returns True if the inactivation date is valid and false otherwise.

mora.validator.**is_location_update_valid**(*req: dict*) → bool
Check if the location update frontend request is valid.

Parameters **req** – The request send from the frontend.

Returns True if the location update is valid and false otherwise.

3.12 Proposal for new REST API

3.12.1 Overview

```
/o
/o/<uuid>
/o/<uuid>/children
/ou/<uuid>
/ou/<uuid>/children
/ou/<uuid>/history
/ou/<uuid>/details
/ou/<uuid>/details/<detail_name>
/ou/create
/ou/<uuid>/edit
/ou/<uuid>/terminate
/e
/e/<uuid>
/e/<uuid>/details
/e/<uuid>/detail/<detail_name>
/e/<uuid>/history
/e/<uuid>/create
/e/<uuid>/edit
/e/<uuid>terminate
/f
/f/<facet_name>
/search?q=<query>
/auth/login
/auth/logout
```

3.12.2 Organisation

Get all organisations

```
GET /o
```

Return a list of all organisations

RESPONSE

```
[
    {
        uuid:  <uuid>
        name: <String>
        has_children: <Boolean>
    },
```

```
    ...
]
```

Get organisation

`GET /o/<uuid>`

Return stats about the organisation

RESPONSE

```
{
    uuid: <uuid>
    name: <String>
    employees: <int>
    units: <int>
    ...
}
```

Get organisation children

`GET /o/<uuid>/children`

Return a list of children for an organisation

RESPONSE

```
{
    uuid:  <uuid>
    name: <String>
    children: [
        {
            uuid: <uuid>
            name: <String>
            has_children: <Boolean>
        },
        ...
    ]
}
```

3.12.3 Organisation unit

Get organisation unit

`GET /ou/<uuid>`

Return basic info about an organisation unit

RESPONSE

```
{
    uuid: <uuid>
    name: <String>
    organisation: <organisation object>
    parent: <parent_org_unit object>
    type: <type object>
    valid_from: <ISO8601>
    valid_to: <ISO8601>
}
```

Get organisation unit children

```
GET /ou/<uuid>/children
```

Return a list of children for an organisation unit

RESPONSE

```
{
    uuid:  <uuid>
    name: <String>
    children: [
        {
            uuid: <uuid>
            name: <String>
            has_children: <Boolean>
        },
        ...
    ]
}
```

Get organisation unit history

```
GET /ou/<uuid>/history
```

Return a list of all changes on the organisation unit

RESPONSE

```
[
    {
        uuid:  <uuid>
        date: <ISO8601>
        section: <section object>
        object: <object object>
        action: <action object>
        valid_from: <ISO8601>
        user: {
            uuid: <uuid>
            name: <String>
        }
    },
    ...
]
```

Get available organisation unit details

```
GET /ou/<uuid>/details
```

```
{
    unit: <Boolean>
    location: <Boolean>
    contact_channel: <Boolean>
    engagements: <Boolean>
    ...
}
```

Get organisation unit unit details

```
GET /ou/<uuid>/details/unit
```

RESPONSE

```
{
    present: [
        {
            uuid: <uuid>
            name: <String>
            unit_type: <unit_type object>
            org_unit: <org_unit object>
            valid_from: <ISO8601>
            valid_to: <ISO8601>
        },
        ...
    ]
    past: [...]
    future: [...]
}
```

Get organisation unit `type` details

`GET /ou/<uuid>/details/location`

`GET /ou/<uuid>/details/contact_channel`

`GET /ou/<uuid>/details/engagements`

Create a new organisation unit

`POST /ou/create`

Create a new organisation unit

REQUEST OBJECT

```
{
    //to be determined
}
```

RESPONSE

```
{
    //to be determined
}
```

3.12.4 Edit an organisation unit

`POST /ou/<uuid>/edit`

Edit an organisation unit

REQUEST OBJECT

```
{
    //to be determined
}
```

RESPONSE

```
{
    //to be determined
}
```

3.12.5 Terminate an organisation unit

```
DELETE /ou/<uuid>/terminate
```

REQUEST OBJECT

```
{
    valid_from: <ISO8601>
}
```

RESPONSE

```
{
    //to be determined
}
```

3.12.6 Employees

Get all employees

```
GET /e
```

Return a list of all employees

```
[
    {
        uuid: <uuid>
        name: <String>
    },
    ...
]
```

Get employee

```
GET /e/<uuid>
```

Return an employee

```
{
    uuid: <uuid>
    name: <String>
    cpr_no: <int>
}
```

Get available employee details

```
GET /e/<uuid>/details
```

Return a list of details available for an employee

```
{
    engagement: <Boolean>
    association: <Boolean>
    it: <Boolean>
    contact: <Boolean>
    ...
}
```

Get employee engagement details

`GET /e/<uuid>/details/engagement`

Get a list of all engagement details for an employee

```
{
    present: [
        {
            uuid: <uuid>
            org_unit: <org_unit object>
            job_function: <job_function object>
            engagement_type: <engagement_type object>
            valid_from: <ISO8601>
            valid_to: <ISO8601>
        },
        ...
    ]
    past: [...]
    future: [...]
}
```

Get employee `type` details

`GET /e/<uuid>/details/association`

`GET /e/<uuid>/details/it`

`GET /e/<uuid>/details/contact`

Get employee history

`GET /e/<uuid>/history`

Return a list of all changes on the employee

```
[
    {
        uuid: <uuid>
        date: <ISO8601>
        section: <section object>
        object: <object object>
        action: <action object>
        valid_from: <ISO8601>
        user: {
            uuid: <uuid>
            name: <String>
        }
    },
    ...
]
```

Create a new employee employment

`POST /e/<uuid>/create`

Create a new employment

REQUEST OBJECT

```
[
    {
        type: engagement
        org_unit_uuid: <uuid>
        job_title_uuid: <uuid>
        engagement_type_uuid: <uuid>
        valid_from: <ISO8601>
        valid_to: <ISO8601>
    },
    {
        type: associaton
        ...
    },
    {
        type: it
        ...
    },
    ...
]
```

RESPONSE

```
{
    //to be determined
}
```

Edit employee details

`POST /e/<uuid>/edit`

Edit an employee

REQUEST OBJECT

```
[
    {
        type: engagement
        uuid: <uuid>
    },
    {
        type: move,
        destination_uuid: <uuid>
        valid_from: <ISO8601>
        valid_to: <ISO8601>
    },
    {
        type: contact,
        channel_uuid: <uuid>
        value: <String>
        valid_from: <ISO8601>
        valid_to: <ISO8601>
    }
]
```

RESPONSE

```
{
    //to be determined
}
```

Terminate an employee

`DELETE /e/<uuid>/terminate`

Terminate an employee

REQUEST OBJECT

```
{
    valid_from: <ISO8601>
}
```

RESPONSE

```
{
    //to be determined
}
```

3.12.7 Facets

Get a list of all facets

`GET /f`

Get a list of all facets

RESPONSE

```
[
    {
        name: org_unit_types
    },
    {
        name: contact_types
    },
    ...
]
```

Get `facet_name` facets

`GET /f/<facet_name>`

Get a list containing the facet values

RESPONSE

```
[
    {
        name: <String>
        uuid: <uuid>
    },
    ...
]
```

3.12.8 Search

`GET /search?q=<query>`

Return a list of results based on the query

It should be able to return both organisations and employees

```
[
    {
        type: employee
        name: <String>
        uuid: <uuid>
    },
    {
        type: organisation
        name: <String>
        uuid: <uuid>
    }
]
```

3.12.9 Authentication

Login

`POST /auth/login`

POST OBJECT

```
{
    username: <String>
    password: <String>
    remember_me: <Boolean>
}
```

RESPONSE

```
{
    //to be determined
}
```

Logout

`POST /auth/logout`

POST PBJECT

```
{
    username: <String>
}
```

RESPONSE

```
{
    //to be determined
}
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mora, 44
mora.api, 14
mora.app, 46
mora.auth, 17
mora.cli, 47
mora.converters, 47
mora.converters.addr, 47
mora.converters.importing, 47
mora.converters.meta, 47
mora.converters.reading, 47
mora.converters.writing, 48
mora.exceptions, 49
mora.integrations, 49
mora.integrations.serviceplatformen, 49
mora.lora, 49
mora.service, 50
mora.service.address, 13
mora.service.association, 51
mora.service.common, 51
mora.service.cpr, 18
mora.service.details, 18
mora.service.employee, 22
mora.service.engagement, 68
mora.service.facet, 34
mora.service.itsystem, 36
mora.service.keys, 71
mora.service.leave, 71
mora.service.manager, 72
mora.service.mapping, 72
mora.service.org, 37
mora.service.orgunit, 38
mora.service.role, 78
mora.settings, 78
mora.tokens, 78
mora.util, 78
mora.validator, 79

HTTP Routing Table

/mo

GET /mo/acl/, 17
GET /mo/addressws/geographical-location, 14
GET /mo/e/, 14
GET /mo/e/(int:cpr_number)/, 14
GET /mo/e/(uuid:id)/, 14
GET /mo/e/(uuid:userid)/role-types/(any:role)/, 16
GET /mo/it-system/, 15
GET /mo/it/, 15
GET /mo/o/, 15
GET /mo/o/(uuid:id)/, 15
GET /mo/o/(uuid:orgid)/full-hierarchy, 15
GET /mo/o/(uuid:orgid)/it/, 15
GET /mo/o/(uuid:orgid)/org-unit/, 15
GET /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/, 15
GET /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/history/, 16
GET /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/role-types/(any:role)/, 16
GET /mo/org-unit/(uuid:unitid)/, 16
GET /mo/org-unit/(uuid:unitid)/history/, 16
GET /mo/org-unit/(uuid:unitid)/role-types/(any:role)/, 16
GET /mo/org-unit/type, 16
GET /mo/role-types/, 17
GET /mo/role-types/contact/facets/properties/classes/, 17
GET /mo/role-types/contact/facets/type/classes/, 17
GET /mo/role-types/engagement/facets/(any:facet)/classes/, 17
POST /mo/acl/, 17
POST /mo/e/(uuid:employeeid)/actions/role, 14
POST /mo/e/(uuid:employeeid)/role-types/(any:role), 14
POST /mo/mo/e/(uuid:employeeid)/actions/role, 14
POST /mo/o/(uuid:orgid)/org-unit, 15
POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid), 15
POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/ac 16
POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/ro 16
POST /mo/o/(uuid:orgid)/org-unit/(uuid:unitid)/ro 16
POST /mo/org-unit, 15
POST /mo/org-unit/(uuid:unitid), 15
POST /mo/org-unit/(uuid:unitid)/actions/move, 16
POST /mo/org-unit/(uuid:unitid)/role-types/locati 16
POST /mo/service/user/(user)/logout, 17
POST /mo/service/user/(username)/login, 17
DELETE /mo/o/(uuid:orgid)/org-unit/(uuid:unitid), 15
DELETE /mo/org-unit/(uuid:unitid), 15

/service

GET /service/(any:type)/(uuid:id)/details/, 18
GET /service/(any:type)/(uuid:id)/details/(functi 18
GET /service/(any:type)/(uuid:parentid)/children, 39
GET /service/e/(uuid:employee_uuid)/history/, 32
GET /service/e/(uuid:id)/, 33
GET /service/e/cpr_lookup/, 18
GET /service/o/, 37
GET /service/o/(uuid:orgid)/, 38
GET /service/o/(uuid:orgid)/address_autocomplete/ 13
GET /service/o/(uuid:orgid)/e/, 34
GET /service/o/(uuid:orgid)/f/, 35
GET /service/o/(uuid:orgid)/f/(facet)/, 35
GET /service/o/(uuid:orgid)/it/, 37
GET /service/o/(uuid:orgid)/ou/, 39

GET /service/ou/(uuid:unitid)/, 40
GET /service/ou/(uuid:unitid)/history/,
43
POST /service/e/(uuid:employee_uuid)/create,
22
POST /service/e/(uuid:employee_uuid)/edit,
26
POST /service/e/(uuid:employee_uuid)/terminate,
32
POST /service/e/create, 33
POST /service/ou/(uuid:unitid)/create,
41
POST /service/ou/(uuid:unitid)/edit,
42
POST /service/ou/(uuid:unitid)/terminate,
43
POST /service/ou/create, 44

Index

A

AbstractRelationDetail (class in mora.service.common), 51
ADAPTED_ZERO_TO_MANY (mora.service.common.FieldTypes attribute), 51
add_bruger_history_entry() (in module mora.service.common), 51
Address (class in mora.converters.meta), 47
address_autocomplete() (in module mora.service.address), 50
Addresses (class in mora.service.address), 50
autocomplete_address() (in module mora.converters.addr), 47

B

base_path (mora.lora.Scope attribute), 49

C

cache (class in mora.service.common), 51
checked_get() (in module mora.service.common), 51
Connector (class in mora.lora), 49
convert() (in module mora.converters.importing), 47
convert_bruger() (in module mora.converters.importing), 47
convert_facet() (in module mora.converters.importing), 47
convert_itsystem() (in module mora.converters.importing), 47
convert_klasse() (in module mora.converters.importing), 47
convert_klassifikation() (in module mora.converters.importing), 47
convert_organisation() (in module mora.converters.importing), 47
convert_organisationenhed() (in module mora.converters.importing), 47
convert_organisationfunktion() (in module mora.converters.importing), 47
convert_reg_to_history() (in module mora.service.common), 51
create() (in module mora.lora), 50
create() (mora.lora.Scope method), 49
create() (mora.service.address.Addresses method), 50
create() (mora.service.common.AbstractRelationDetail method), 51
create() (mora.service.itsystem.ITSystems method), 70
create() (mora.service.orgunit.OrgUnit method), 73
create_association() (in module mora.service.association), 51
create_bruger_payload() (in module mora.service.common), 51
create_contact() (in module mora.converters.writing), 48
create_employee() (in module mora.service.employee), 57
create_employee_relation() (in module mora.service.employee), 58
create_employee_role() (in module mora.api), 44
create_engagement() (in module mora.service.engagement), 68
create_leave() (in module mora.service.leave), 72
create_manager() (in module mora.service.manager), 72
create_org_unit() (in module mora.converters.writing), 48
create_org_unit() (in module mora.service.orgunit), 73
create_org_unit_relation() (in module mora.service.orgunit), 74
create_organisation_unit() (in module mora.api), 45
create_organisationsenhed_payload() (in module mora.service.common), 52
create_organisationsfunktion_payload() (in module mora.service.common), 52
create_role() (in module mora.service.role), 78
create_update_kwargs() (in module mora.converters.writing), 48

D

defaults (mora.lora.Connector attribute), 49
delete() (in module mora.lora), 50
delete() (mora.lora.Scope method), 49
DetailType (class in mora.service.details), 53
do_ranges_overlap() (in module mora.util), 78

E

edit() (mora.service.address.Addresses method), 50

edit() (mora.service.common.AbstractRelationDetail method), 51
edit() (mora.service.itsystem.ITSystems method), 70
edit() (mora.service.orgunit.OrgUnit method), 73
edit_association() (in module mora.service.association), 51
edit_employee() (in module mora.service.employee), 61
edit_engagement() (in module mora.service.engagement), 68
edit_leave() (in module mora.service.leave), 72
edit_manager() (in module mora.service.manager), 72
edit_org_unit() (in module mora.service.orgunit), 74
edit_role() (in module mora.service.role), 78
ensure_bounds() (in module mora.service.common), 52

F

fetch() (in module mora.lora), 50
fetch() (mora.lora.Scope method), 50
FieldTuple (in module mora.service.common), 51
FieldTypes (class in mora.service.common), 51
find_address() (in module mora.converters.addr), 47
format_cpr_response() (in module mora.service.cpr), 53
from_iso_time() (in module mora.util), 78
fromdict() (mora.converters.meta.Address class method), 47
fromstring() (mora.converters.meta.Address class method), 47
fromstring() (mora.converters.meta.PhoneNumber class method), 47
FULL (mora.service.orgunit.UnitDetails attribute), 73
full_hierarchies() (in module mora.converters.reading), 47
full_hierarchy() (in module mora.api), 45
full_hierarchy() (in module mora.converters.reading), 48

G

get() (in module mora.lora), 50
get() (mora.lora.Scope method), 50
get() (mora.service.address.Addresses method), 50
get() (mora.service.common.AbstractRelationDetail method), 51
get() (mora.service.itsystem.ITSystems method), 70
get() (mora.service.orgunit.OrgUnit method), 73
get_address() (in module mora.converters.addr), 47
get_address_class() (in module mora.service.address), 51
get_all() (mora.lora.Scope method), 50
get_args_flag() (in module mora.service.common), 52
get_children() (in module mora.service.orgunit), 75
get_citizen() (in module mora.integrations.serviceplatformen), 49
get_class() (in module mora.converters.reading), 48
get_classes() (in module mora.converters.reading), 48
get_classes() (in module mora.service.facet), 68
get_connector() (in module mora.service.common), 52
get_contact_channels() (in module mora.converters.reading), 48
get_contact_facet_properties_classes() (in module mora.api), 45
get_contact_facet_types_classes() (in module mora.api), 45
get_contact_properties() (in module mora.converters.reading), 48
get_contact_types() (in module mora.converters.reading), 48
get_date_chunks() (mora.lora.Connector method), 49
get_detail() (in module mora.service.details), 53
get_effect_from() (in module mora.service.common), 52
get_effect_to() (in module mora.service.common), 52
get_effect_validity() (in module mora.service.common), 52
get_effects() (mora.lora.Scope method), 50
get_employee() (in module mora.api), 45
get_employee() (in module mora.service.employee), 66
get_employee_by_cpr() (in module mora.api), 45
get_employee_history() (in module mora.service.employee), 67
get_employees() (in module mora.converters.reading), 48
get_engagement_classes() (in module mora.api), 45
get_engagements() (in module mora.converters.reading), 48
get_field_value() (in module mora.service.common), 52
get_geographical_addresses() (in module mora.api), 45
get_it_systems() (in module mora.converters.reading), 48
get_locations() (in module mora.converters.reading), 48
get_obj_value() (in module mora.service.common), 52
get_one_address() (in module mora.service.address), 51
get_one_class() (in module mora.service.facet), 69
get_one_employee() (in module mora.service.employee), 67
get_one_facet() (in module mora.service.facet), 69
get_one_organisation() (in module mora.service.org), 72
get_one_orgunit() (in module mora.service.orgunit), 76
get_org_unit() (in module mora.lora), 50
get_org_unit_history() (in module mora.service.orgunit), 76
get_organisation() (in module mora.api), 45
get_organisation() (in module mora.service.org), 72
get_organisations() (in module mora.converters.reading), 48
get_orgunit() (in module mora.api), 45
get_orgunit() (in module mora.converters.reading), 48
get_orgunit() (in module mora.service.orgunit), 77
get_orgunit_history() (in module mora.api), 45
get_orgunits() (in module mora.converters.reading), 48
get_relation_for() (in module mora.service.address), 51
get_relation_for() (mora.service.itsystem.ITSystems

static method), 71
get_remaining_org_funk_fields() (in module mora.converters.writing), 48
get_role() (in module mora.api), 45
get_token() (in module mora.tokens), 78
get_unit_type() (in module mora.converters.reading), 48
get_uuid() (in module mora.service.common), 52
get_valid_from() (in module mora.service.common), 52
get_valid_to() (in module mora.service.common), 52
get_validity_effect() (in module mora.service.common), 52

H

handle_invalid_usage() (in module mora.app), 46
has() (mora.service.address.Addresses static method), 50
has() (mora.service.common.AbstractRelationDetail method), 51
has() (mora.service.itsystem.ITSystems method), 71
has() (mora.service.orgunit.OrgUnit method), 73

I

IllegalArgumentException, 49
inactivate_old_interval() (in module mora.service.common), 52
inactivate_org_funktion() (in module mora.converters.writing), 48
inactivate_org_funktion_payload() (in module mora.service.common), 53
inactivate_org_unit() (in module mora.api), 45
inactivate_org_unit() (in module mora.converters.writing), 48
is_candidate_parent_valid() (in module mora.validator), 79
is_cpr_number() (in module mora.util), 78
is_create_org_unit_request_valid() (in module mora.validator), 79
is_effect_relevant() (mora.lora.Connector method), 49
is_inactivation_date_valid() (in module mora.validator), 79
is_location_update_valid() (in module mora.validator), 79
is_reg_valid() (in module mora.service.common), 53
is_urn() (in module mora.util), 78
is_uuid() (in module mora.util), 78
ITSystems (class in mora.service.itsystem), 70

L

list_classes() (in module mora.api), 45
list_details() (in module mora.service.details), 57
list_employees() (in module mora.api), 45
list_employees() (in module mora.converters.reading), 48
list_employees() (in module mora.service.employee), 67
list_facets() (in module mora.service.facet), 69
list_it_systems() (in module mora.converters.reading), 48
list_it_systems() (in module mora.service.itsystem), 71
list_itsystems() (in module mora.api), 45
list_organisations() (in module mora.api), 46
list_organisations() (in module mora.service.org), 72
list_orgunits() (in module mora.api), 46
list_orgunits() (in module mora.converters.reading), 48
list_orgunits() (in module mora.service.orgunit), 77
list_roles() (in module mora.api), 46
load_cli() (in module mora.cli), 47
load_data() (in module mora.converters.importing), 47
login() (in module mora.auth), 46
logout() (in module mora.auth), 47

M

MINIMAL (mora.service.orgunit.UnitDetails attribute), 73
mora (module), 44
mora.api (module), 14, 44
mora.app (module), 46
mora.auth (module), 17, 46
mora.cli (module), 47
mora.converters (module), 47
mora.converters.addr (module), 47
mora.converters.importing (module), 47
mora.converters.meta (module), 47
mora.converters.reading (module), 47
mora.converters.writing (module), 48
mora.exceptions (module), 49
mora.integrations (module), 49
mora.integrations.serviceplatformen (module), 49
mora.lora (module), 49
mora.service (module), 50
mora.service.address (module), 13, 50
mora.service.association (module), 51
mora.service.common (module), 51
mora.service.cpr (module), 18, 53
mora.service.details (module), 18, 53
mora.service.employee (module), 22, 57
mora.service.engagement (module), 68
mora.service.facet (module), 34, 68
mora.service.itsystem (module), 36, 70
mora.service.keys (module), 71
mora.service.leave (module), 71
mora.service.manager (module), 72
mora.service.mapping (module), 72
mora.service.org (module), 37, 72
mora.service.orgunit (module), 38, 73
mora.service.role (module), 78
mora.settings (module), 78
mora.tokens (module), 78
mora.util (module), 78
mora.validator (module), 79
move_org_funktion_payload() (in module mora.converters.writing), 49
move_org_unit() (in module mora.api), 46

move_org_unit() (in module mora.converters.writing), 49

N

NCHILDREN (mora.service.orgunit.UnitDetails attribute), 73
now() (in module mora.util), 78

O

OrgUnit (class in mora.service.orgunit), 73

P

paged_get() (mora.lora.Scope method), 50
parsedatetime() (in module mora.util), 78
PhoneNumber (class in mora.converters.meta), 47

R

read_paths() (in module mora.converters.importing), 47
relation_types (mora.service.details.DetailType attribute), 53
rename_or_retype_org_unit() (in module mora.api), 46
rename_org_unit() (in module mora.converters.writing), 49
replace_relation_value() (in module mora.service.common), 53
requires_auth() (in module mora.cli), 47
restrictargs() (in module mora.util), 78
retype_org_unit() (in module mora.converters.writing), 49
RFC
    RFC 1738, 36, 69
    RFC 5322#section-3.4, 36, 69
root() (in module mora.app), 46

S

SAMLAuth (class in mora.auth), 46
Scope (class in mora.lora), 49
scope (mora.service.common.AbstractRelationDetail attribute), 51
scope (mora.service.details.DetailType attribute), 53
scope_map (mora.lora.Connector attribute), 49
search (mora.service.details.DetailType attribute), 53
search_cpr() (in module mora.service.cpr), 53
send_scripts() (in module mora.app), 46
send_styles() (in module mora.app), 46
set_object_value() (in module mora.service.common), 53
splitlist() (in module mora.util), 79

T

terminate_employee() (in module mora.service.employee), 68
terminate_org_unit() (in module mora.service.orgunit), 78
to_frontend_time() (in module mora.util), 79
to_iso_time() (in module mora.util), 79
to_lora_time() (in module mora.util), 79
today() (in module mora.util), 79

U

uniqueify() (in module mora.util), 79
unit_history() (in module mora.converters.reading), 48
UnitDetails (class in mora.service.orgunit), 73
unparsedate() (in module mora.util), 79
update() (in module mora.lora), 50
update() (mora.lora.Scope method), 50
update_config() (in module mora.util), 79
update_employee_addresses() (in module mora.converters.writing), 49
update_org_funktion_payload() (in module mora.converters.writing), 49
update_org_unit_addresses() (in module mora.converters.writing), 49
update_organisation_unit_location() (in module mora.api), 46
update_payload() (in module mora.service.common), 53

V

v2_root() (in module mora.app), 46
validity (mora.lora.Connector attribute), 49

W

wrap_in_org() (in module mora.converters.reading), 48

Z

ZERO_TO_MANY (mora.service.common.FieldTypes attribute), 51
ZERO_TO_ONE (mora.service.common.FieldTypes attribute), 51