KNOWLEDGE CENTRE FOR AGRICULTURE

Farmtracking API description

1 Introduction

This document describes the API for the Farmtracking service which is part of the Farmtracking application.

Also read the *Farmtracking developer introduction* document for a description of the whole Farmtracking application

The description below is a first draft and will likely change as implementation progress.

2 Entities

The following entities will be sent via the API

2.1 Task

A task (also called notification in the *Farmtracking developer introduction* document) is the entity returned when the device is polling the backend with a new GPS position. It describes the message the user will see in the phones notifications system.

Name	Data type	Description	Editable
Id	String	Unique id	No
ServiceId	String	Origin subsystem	No
UserId	String	User the Task addressed to	No
Title	String	Title of the task	No
Text	String	Message in the task	No
Url	String	Related url the user can visit	No
IsRead	Boolean	User has read the task	Yes
Postponed	Nullable<DateTime>	User has postponed task to this date (UTC)	Yes
FieldTasks	Collection<FieldTasks>	List of fields the task is attached to	No

2.2 FieldTask

A FieldTask is only used as a child entity to a Task.

Name	Data type	Description	Editable
Id	String	Unique id	No
TaskId	String	Parent Task Id	No
Title	String	Title of the FieldTask (Field number and name)	No
FieldId	Int32	Id of the Field this FieldTask represent	No

IsDone	Boolean	The Task has been carried out on this field	Yes

2.3 Farm

Represents a farm owned or in another way available for the user.

Name	Data type	Description	Editable
Id	Int32	Unique id of the farm	No
Name	String	Farm name	No

2.4 Field

Represents a field owned or in another way available for the user.

The field entity includes the field geometry in Well-known Text (WKT) format.

Name	Data type	Description	Editable
Id	Int32	Unique id	No
FarmId	Int32	Id of the farm	No
FarmName	String	Farm name	No
HarvestYear	Int32	The harvest year (season) this field is available in	No
FieldNumber	String	Displayed field number e.g. ”15-1”	No
FieldName	String	Field name	No
PreCropName	String	Previous year crop name	No
Geometry	String	Geometry of the field. Well-known Text format.	No

2.5 Crop

A field can have one or more crops.

Name	Data type	Description	Editable
Id	Int32	Unique id	No
FieldId	Int32	Id of parent Field	No
FarmId	Int32	Id of the farm	No
HarvestYear	Int32	The harvest year (season) this crop is available in	No
CropName	String	Crop name	No
SuccessionNo	String	The crop with the highest succession number on a field is the main crop	No
VarietyName	String	Variety name	No

2.6 Hotspot

A hotspot is a registered place of interest with a geo position and type

Name	Data type	Description	Editable
Id	Int32	Unique id	No
LastUpdatedDate	DateTime	Date for last change (UTC). Is automatic updated on POST and PATCH requests	No
HotspotTypeId	Int32	Hotspot type	Yes
HotspotSubTypeId	Collection<Int32>	List of hotspot subtypes e.g. weed type	Yes
Geometry	String	Geometry of hotspot. Well-known Text format.	Yes
Description	String	User written description	Yes
Images	Collec-tion<HotspotImage>	List of images the user has taken of the hotspot (Base64 encoded jpeg)	Yes
FarmId	Int32	Farm this hotspot is belonging to	Yes

2.7 HotspotImage

Jpeg image from the hotspot

Name	Data type		
Id	Int32	Unique id	No
HotspotId	Int32	Id of the parent Hotspot	Yes
ImageData	String	Base64 encoded image data	Yes

2.8 HotspotType

Hotspot type (e.g. weed, stone) to be used when registering a hotspot

Name	Data type		
Id	Int32	Unique id	No
Name	String	Hotspot type name	No

2.9 HotspotSubType

A subtype to the hotspot type (e.g. weedtypes)

Name	Data type		
Id	Int32	Unique id	No
HotspotTypeId	Int32	Id of the parent hotspot type	No
Name	String	Hotspot type name	No

3 API usage

The API is implemented as a Web API and has the following OData endpoints:

Url	Commands
https://farmtracking.dlbr.dk/api/Tasks	GET, PATCH
https://farmtracking.dlbr.dk/api/FieldTasks	PATCH
https://farmtracking.dlbr.dk/api/Farms	GET
https://farmtracking.dlbr.dk/api/Fields	GET
https://farmtracking.dlbr.dk/api/Crops	GET
https://farmtracking.dlbr.dk/api/Hotspots	GET, POST, PATCH, DELETE
https://farmtracking.dlbr.dk/api/HotspotImages	GET, POST, DELETE
https://farmtracking.dlbr.dk/api/HotspotTypes	GET
https://farmtracking.dlbr.dk/api/HotspotSubTypes	GET

Notes:

- The entity and property names are case sensitive
- Currently the API has no authorization. For now every request will automatically be authenticated as the user fttest1@PROD.DLI
- The odata requests can be filtered, ordered etc. with use of the odata syntax. See for example http://www.odata.org/getting-started/basic-tutorial/
-

3.1 Task

To get all Tasks for the user use the following request

```
GET https://devtest-farmtracking.vfltest.dk/api/Tasks
```

Used on reminders main page

The following examples shows the use of the odata syntax for ordering and filtering

```
GET https://devtest-farmtracking.vfltest.dk/api/Tasks?$orderby=Title desc
```

```
GET https://devtest-farmtracking.vfltest.dk/api/Tasks?$filter=startswith(Title,'Test')
```

The following request will return the list of Tasks and include all child FieldTasks

```
GET https://devtest-farmtracking.vfltest.dk/api/Tasks?$expand=FieldTasks
```

To get a single Task by its id, use this request

```
GET https://devtest-farmtracking.vfltest.dk/api/Tasks('Plant_Task22')
```

This can be combined with the expand command to get a single Task with expanded child FieldTasks

```
GET https://devtest-
farmtracking.vfltest.dk/api/Tasks('Plant_Task22')?$expand=FieldTasks
```

Used on reminder details page

To update a Task use the PATCH method. This will only update the properties for the entity sent in the request body.

To update Task postponed date:

```
PATCH https://devtest-farmtracking.vfltest.dk/api/Tasks('Plant_Task22') HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "Postponed":"2014-09-23T00:00:00Z"
}
```

Used on reminder details page

Clear the postponed date:

```
PATCH https://devtest-farmtracking.vfltest.dk/api/Tasks('Plant_Task22') HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "Postponed":null
}
```

Used on reminder details page

Update IsRead property:

```
PATCH https://devtest-farmtracking.vfltest.dk/api/Tasks('Plant_Task22') HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "IsRead":true
}
```

Used on reminder details page

Update multiple properties:

```
PATCH https://devtest-farmtracking.vfltest.dk/api/Tasks('Plant_Task22') HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "Postponed":"2014-09-23T00:00:00Z",
   "IsRead":true
}
```

Used on reminder details page

The following request the will poll for new Taks in relation to the current position. It will return a list of new taskIds – if any. In case of any the tasks, the client should download a fresh Task list

```
POST https://devtest-
farmtracking.vfltest.dk/api/Tasks/FT.TaskChangesAvailableForLocation HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "latitude": 1,
   "longitude": 1
}
```

Used in background polling thread

3.2 FieldTask

FieldTasks is only used on the reminder details screen. As described above the only way to get a list of FieldTask is by expanding the Task entity.

This request will return a single FieldTask:

```
GET https://devtest-farmtracking.vfltest.dk/api/FieldTasks('Plant_Task22_FieldId12345')
```

This request will update the IsDone property on a FieldTask entity:

```
PATCH https://devtest-
farmtracking.vfltest.dk/api/FieldTasks('Plant_Task22_FieldId12345') HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "IsDone":true
}
```

3.3 Farm

The available farms for the current user can be received with the following request:

```
GET https://devtest-farmtracking.vfltest.dk/api/Farms
```

3.4 Field

When requesting a Field list you always have to specify FarmId(s) and HarvestYear parameters in the odata filter string.

Get all Fields for a specific farm in a specific harvest year :

```
GET https://devtest-farmtracking.vfltest.dk/api/Fields?$filter=FarmId eq 71965 and Har-
vestYear eq 2014
```

Get all Fields for two farms in a specific harvest year :

```
GET https://devtest-farmtracking.vfltest.dk/api/Fields?$filter=(FarmId eq 71965 or Far-
mId eq 29275) and HarvestYear eq 2014
```

Get all Fields for a specific farm in a specific harvest year – and get child crops for each field:

```
GET https://devtest-farmtracking.vfltest.dk/api/Fields?$filter=FarmId eq 71965 and Har-
vestYear eq 2014&$expand=Crops
```

3.5 Crop

When requesting a Crop list you always have to specify FarmId(s) and HarvestYear parameters in the odata filter string.

Get all Crop for a specific farm in a specific harvest year :

```
GET https://devtest-farmtracking.vfltest.dk/api/Crops?$filter=FarmId eq 71965 and Har-
vestYear eq 2014
```

Note: It's also possible to get the Crops by using the Expand parameter on the Fields endpoint (as described in 3.3)

3.6 Hotspot

Get Hotspots for specific farms

```
GET https://devtest-farmtracking.vfltest.dk/api/Hotspots?$filter=(FarmId eq 71965 or
FarmId eq 29275)
```

Get single Hotspot by Id:

```
GET https://devtest-farmtracking.vfltest.dk/api/Hotspots(5)
```

To insert a new Hotpost use POST. The Geometry is in WKT format

```
POST https://devtest-farmtracking.vfltest.dk/api/Hotspots HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "HotspotTypeId":1,
   "HotspotSubTypeIds":[12,25,48],
   "FarmId":29275,
   "RegisteredDate":"2014-09-23T00:00:00Z",
   "Description":"Hotspot 33",
   "Geometry":"POLYGON ((10.142741203099936 56.202911526834306, 10.142987966329269
56.202962254387657, 10.142977237493215 56.202857815234104, 10.142741203099936
56.202911526834306))"
}
```

Use PATCH to update a Hotspot:

```
PATCH https://devtest-farmtracking.vfltest.dk/api/Hotspots(6) HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "Description":"Hotspot 33 – Updated from API",
   "Geometry":"POLYGON ((10.142741203099936 56.202911526834306, 10.142987966329269
56.202962254387657, 10.142977237493215 56.202857815234104, 10.142741203099936
56.202911526834306))"
}
```

A Hotspot can be deleted with the following request:

```
DELETE https://devtest-farmtracking.vfltest.dk/api/Hotspots(6)
```

3.7 HotspotImage

All images are expected in jpeg-format

A list of HotspotImages can be fetch by expanding the Images property of Hotspots

```
GET https://devtest-farmtracking.vfltest.dk/api/Hotspots?$expand=Images
```

An example for a single Hotspot with imgaes

```
GET https://devtest-farmtracking.vfltest.dk/api/Hotspots(7)?$expand=Images
```

If you know the HotspoitImage id, you can get it from the following request:

```
GET https://devtest-farmtracking.vfltest.dk/api/HotspotImages(6)
```

To insert at new HotspotImage use:

```
POST https://devtest-farmtracking.vfltest.dk/api/HotspotImages HTTP/1.1
Host: devtest-farmtracking.vfltest.dk
Content-Type: application/json

{
   "HotspotId":1,
   "ImageData":"base64-encoded-string"
}
```

Delete a HotspotImage:

```
DELETE https://devtest-farmtracking.vfltest.dk/api/HotspotImages(6)
```

3.8 HotspotType

Get all HotspotTypes with this request:

```
GET https://devtest-farmtracking.vfltest.dk/api/HotspotTypes
```

3.9 HotspotSubType

Get all HotspotSubTypes with this request:

```
GET https://devtest-farmtracking.vfltest.dk/api/HotspotSubTypes
```

To filter the HotspotSubTypes on a specific HotspotType use odata filtering:

```
GET https://devtest-farmtracking.vfltest.dk/api/HotspotSubTypes?$filter=HotspotTypeId
eq 1
```

4 Environments

For development the Web API will be available in a test environment

4.1 Test

https://devtest-farmtracking.vfltest.dk/

4.2 Production

https://farmtracking.dlbr.dk/

5 Authentication / authorization

The authentication model for the Farmtracking application is currently not decided. It can be either ADFS or OAuth.

5.1 Authorization

Each request must include an Authorization header with the value “Bearer <encodedToken>”

<encodedToken> is produced by base64 encode the SAML token obtain through the login flow

5.2 Authorization Error Codes

When authorization fails the API will respond with http code “401 Unauthorized”. If possible a JSON error entity is also sent with detailed information.

Error entity

Name	Data type	Description
Code	String	Error code
Message	String	Detailed error description

Error codes

Code	Description
401.50	Error decoding token
401.51	Error reading token
401.52	Token expired
401.99	Unknown error