Service Template Specification

STS id fertilisation\_advisor

Version 0.2.4

Status PUBLISHED

Author(s) M. Caballero (AgriCircle AG)

**The "fertilisation\_advisor" Service Template**

Abstract

This document describes the specifications for fertilisation\_advisor services whose purpose is to provide product application maps for selected nutrients. fertilisation\_advisor services are agnostic of tractors or other equipment; they typically use agronomical algorithms to provide advice on fertilization which would then be typically carried out by the ATLAS Equipment Centre.

Table of Contents

[1 Introduction 3](#_Toc102577296)

[2 Terminology 3](#_Toc102577297)

[3 Fertilisation Advisor Usage Scenarios 3](#_Toc102577298)

[3.1 NPK Fertilisation 4](#_Toc102577299)

[4 Advisor Service Template API 5](#_Toc102577300)

[4.1 Field Registration Endpoints 5](#_Toc102577301)

[4.1.1 Register field 5](#_Toc102577302)

[4.1.2 Get field registration info 6](#_Toc102577303)

[4.1.3 Unregister field 6](#_Toc102577304)

[4.2 Advice Endpoints 6](#_Toc102577305)

[4.2.1 Prepare Advice 6](#_Toc102577306)

[4.2.2 Get Advice Info 7](#_Toc102577307)

[4.2.3 Cancel Advice 7](#_Toc102577308)

[4.2.4 Get Application Info 7](#_Toc102577309)

[5 Data Formats 8](#_Toc102577310)

[5.1 Product Application File Format 8](#_Toc102577311)

[6 Access and Authentication 11](#_Toc102577312)

[7 Dynamic Behaviour 11](#_Toc102577313)

[7.1 Fertilisation Lifecycle 12](#_Toc102577314)

[7.2 Field Registration 12](#_Toc102577315)

[7.2.1 Polling 12](#_Toc102577316)

[7.2.2 Notification 13](#_Toc102577317)

[7.3 Advice Preparation 14](#_Toc102577318)

[7.3.1 Polling 14](#_Toc102577319)

[7.3.2 Notification 15](#_Toc102577320)

[7.4 General Comments on Notifications 15](#_Toc102577321)

# Introduction

Fertilisation is essential for crop yield. The fertilisation requirements depend on the nutrient requirements for a particular crop type and the availability of these nutrients in the soil.

fertilisation\_advisor services determine product application maps to optimally satisfy specific nutrients requirements which can later be applied in one or more applications by the ATLAS Equipment Centre on a field.

Farmers may use different fertilisation\_advisor services depending on the specific set of nutrients they focus on, their budget, the accuracy and the granularity of information they require, etc.

# Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](https://datatracker.ietf.org/doc/html/rfc2119) [[RFC2119](https://datatracker.ietf.org/doc/html/rfc2119)] and indicate requirement levels for compliant implementations.

The notation "[xxx]" (xxx in square brackets) is equivalent to "array of xxx".

When used alone, the terms "fertilisation\_advisor" and "advisor" refer to "fertilisation\_advisor service template". Instead, "fertilisation\_advisor service and "advisor service" are equivalent to "a service implementing the fertilisation\_advisor service template".

# Fertilisation Advisor Usage Scenarios

The following sections describe some aspects of the use of the fertilisation\_advisor service template. The examples were chosen to illustrate the basic functions of applications using fertilisation\_advisor, not to limit what fertilisation\_advisor may be used for.

The typical use case scenarios with a fertilisation\_advisor service typically an FMIS that orchestrates operations with a field\_data and the ATLAS Equipment Centre as can be seen beloA screenshot of a computer

Description automatically generated with medium confidence

## NPK Fertilisation

Most crops have nitrogen (N), phosphorous (P) and potassium (K) nutrient requirements.

A fertilisation service can be designed so as to focus on NPK nutrients and privilege the use of organic over chemical fertilisers.

The service receives a request to prepare an NPK fertilization for a corn plantation using slurry, and two other products, one with high P dosage and another with high K dosage. Based on the corn NPK requirements, the soil's properties and previously applied products, the fertilization service determines the optimal amounts of each product's needs. These application maps are delivered on-demand to another service that is responsible for converting the data to a format understandable by the equipment that will perform the tasks on the field.

# Advisor Service Template API

This section provides a very high-level summary of the advisor service template API:

Field Registration Endpoints

Register field

Get field registration info

Unregister field

Advice Endpoints

Prepare Advice

Get Advice

Cancel Advice

Get Application Info

Within this section, functions are summarized with simple tables:

+--------------------------------------------+

| <logical function name> |

+-------------+------------------------------+

| Inputs | <URL parameters or |

| | request body attributes |

+-------------+------------------------------+

| Outputs | <response body attributes> |

+-------------+------------------------------+

Only the most meaningful parameters are discussed in this document. Please refer to the OpenAPI specifications for full details.

Advisor services are not required to handle intense traffic from a single client, especially for GET functions. Implementors MAY generate a 429 TOO MANY REQUESTS error response if the rate of calls exceed some pre-defined quota.

## Field Registration Endpoints

These functions relate to registering, and unregistering fields. Advisor implementation may provide the means to actively monitor fields and pro-actively generate advices.

### Register field

This endpoint registers a field with the fertilisation\_advisor along with the nutrients for which future advices may be requested or suggested.

+--------------------------------------------+

| register\_field |

+-------------+------------------------------+

| Inputs | field urn, nutrient |

+-------------+------------------------------+

| Outputs | registration status |

+-------------+------------------------------+

The registration process may be a lengthy process. Before planning advices, consumers must ascertain that a field’s registration status is READY by either polling or requesting a notification (see 7.2 Field Registration)

### Get field registration info

This endpoint returns the registration info for a field.

+--------------------------------------------+

| registration\_info |

+-------------+------------------------------+

| Inputs | field urn |

+-------------+------------------------------+

| Outputs | field registration info |

+-------------+------------------------------+

### Unregister field

This endpoint unregisters a field from the service.

+--------------------------------------------+

| unregister\_field |

+-------------+------------------------------+

| Inputs | field urn |

+-------------+------------------------------+

| Outputs | |

+-------------+------------------------------+

## Advice Endpoints

These endpoints relate to the creation of fertilization advices. An advice may be requested to plan and optimize fertilization for a one or more related nutrients (e.g. NPK) to be applied on a field.

### Prepare Advice

This endpoint is used to requests an advice to be prepared for specified nutrients and the products that are at disposition.

the applications that will be carried out (one or more) and their associated products, so that they may optimize advised application maps accordingly.

+--------------------------------------------+

| prepare\_advice |

+-------------+------------------------------+

| Inputs | field urn, application |

+-------------+------------------------------+

| Outputs | advice urn |

+-------------+------------------------------+

fertilisation\_advisor services may retrieve any relevant information for preparing advices from the field\_data service (e.g., boundaries, current crop, previous applications, etc.). Advices should be prepared by taking the current conditions into account. Advisors have the opportunity to refine their internal recommendations at the actual time the application for the advice is requested (see Get Application Info).

Advice preparation may be a lengthy process. Clients may either poll or request to be notified to determine that the advice is READY (see 7.3 Advice Preparation).

### Get Advice Info

This function returns status information about an advice.

+--------------------------------------------+

| advice\_info |

+-------------+------------------------------+

| Inputs | advice urn |

+-------------+------------------------------+

| Outputs | advice info |

+-------------+------------------------------+

In addition to the information submitted when preparing the advice, the advice info also contains a status about the advice preparation which is one of IN\_PROGRESS, CONFIGURATION\_REQUIRED, READY, FAILED.

### Cancel Advice

+--------------------------------------------+

| cancel\_advice |

+-------------+------------------------------+

| Inputs | advice urn |

+-------------+------------------------------+

| Outputs | - |

+-------------+------------------------------+

Invoking this endpoint causes the advice status to become FAILED. If an advice that is not yet in READY state is cancelled and if a notification URL was provided in the prepare\_advice request, a notification MUST be dispatched.

### Get Application Info

This endpoint may only be performed on advices whose status is READY. It returns a download URL to the application map and the total estimated amount of product needed carry out the application.

+--------------------------------------------+

| get\_application\_info |

+-------------+------------------------------+

| Inputs | advice id, application ref |

+-------------+------------------------------+

| Outputs | download URL, product amount |

+-------------+------------------------------+

All values are in the corresponding product units. Please refer to "5.1 Product Application File Format" for detailed specification of the downloaded product application map file format.

For maximum accuracy, the application map is requested in a "just-in-time" fashion by the ATLAS Equipment Centre (or any other consumer) to give an opportunity to advisors to make adjustments to their pre-computed preparation. However, since this information is usually requested just as farmers are about to go out on the field to perform a task, it is important that it is computed in a short time (maximum, few seconds). Any potentially time-consuming processing must be pre-computed and cached internally during the registration and/or preparation stages.

# Data Formats

This section focuses on the description of binary (file) data formats. Please refer to the fertilisation\_advisor OpenAPI specifications for details on all other payload and parameter descriptions.

## Product Application File Format

Nutrient Application data MUST be in GeoPackage (<https://www.geopackage.org/>) format version 1.2 or newer.

+=========================================================+

| TABLE gpkg\_contents |

+=====+=============+=====================================+

| ROW | COLUMN | VALUE |

+-----+-------------+-------------------------------------+

| | table\_name | "atlas" |

| +-------------+-------------------------------------+

| | data\_type | "attributes" |

+-----+-------------+-------------------------------------+

All Atlas GeoPackage files MUST contain an attributes table named "atlas" with a single row and the following structure:

+============================+

| TABLE atlas |

+================+===========+

| COLUMN | TYPE |

+----------------+-----------+

| id | INTEGER |

| type | TEXT |

| participant | TEXT |

| format\_version | TEXT |

+----------------+-----------+

and a single row:

+=========================================================+

| TABLE atlas |

+=====+================+==================================+

| ROW | COLUMN | VALUE |

+-----+----------------+----------------------------------+

| | type | "fertilisation\_application" |

| +----------------+----------------------------------+

| | participant | "<atlas participant id>" |

| 1 +----------------+----------------------------------+

| | application | "…" |

| +----------------+----------------------------------+

| | format\_version | "MAJOR.MINOR" |

+-----+----------------+----------------------------------+

Services SHALL validate that the "atlas.participant" field matches the information attached to the authentication context in which the file is uploaded.

If the GeoPackage was generated and uploaded by an Atlas service, then "atlas.application" SHOULD be "<service name>-<service version>", otherwise it MAY contain "<application name>-<application version>". In either case, no validation will be performed.

The "atlas.format\_version" field MUST be the version of the fertilisation\_advisor template that is targeted by the client uploading the file. GeoPackage files of a given type are guaranteed to be compatible (no breaking changes) for a same MAJOR version of "atlas.format\_version".

A fertilization application GeoPackage MUST contain the following additional tables:

+=========================================================+

| TABLE gpkg\_contents |

+=====+=============+=====================================+

| ROW | COLUMN | VALUE |

+-----+-------------+-------------------------------------+

| … |

+-----+-------------+-------------------------------------+

| | table\_name | "product" |

| +-------------+-------------------------------------+

| | data\_type | "2d-gridded-coverage|features" |

+-----+-------------+-------------------------------------+

| | table\_name | "application\_info" |

| +-------------+-------------------------------------+

| | data\_type | "attributes" |

+-----+-------------+-------------------------------------+

If the data\_type of the product table is "features", then the columns must be as follows:

+=========================================================+

| TABLE product |

+=====+=================+=================================+

| ROW | COLUMN | VALUE |

+-----+-----------------+---------------------------------+

| | id | "INTEGER" |

| +-----------------+---------------------------------+

| | geometry | "GeoPackage Geometry" |

| +-----------------+---------------------------------+

| | amount | "amount in product units (REAL)"|

|-----+-----------------+---------------------------------+

Instead, if the data\_type of the product table is "2d-gridded-coverage" then it must be defined as follows:

+=========================================================+

| TABLE product |

+=====+=================+=================================+

| ROW | COLUMN | VALUE |

+-----+-----------------+---------------------------------+

| | id | "INTEGER" |

| +-----------------+---------------------------------+

| | zoom\_level | "INTEGER" |

| +-----------------+---------------------------------+

| | tile\_column | "INTEGER" |

| +-----------------+---------------------------------+

| | tile\_row | "INTEGER" |

| +-----------------+---------------------------------+

| | tile\_data | "BLOB (TIFF)" |

|-----+-----------------+---------------------------------+

Where tile\_data is a tiff blob holding Float32 values as per GeoPackage specifications ( <http://docs.opengeospatial.org/is/17-066r1/17-066r1.html#_storage_formats_and_grid_cell_values>).

+============================+

| TABLE application\_info |

+================+===========+

| COLUMN | TYPE |

+----------------+-----------+

| id | INTEGER |

| type | TEXT |

| date | DATETIME |

| info | TEXT |

+----------------+-----------+

and a single row:

+=========================================================+

| TABLE application\_info |

+=====+=============+=====================================+

| ROW | COLUMN | VALUE |

+-----+-------------+-------------------------------------+

| | type | "fertilization" |

| 1 +-------------+-------------------------------------+

| | date | "yyyy-MM-ddTHH:mm:ssZ" |

| +-------------+-------------------------------------+

| | info | "<application\_info\_json>" |

+-----+-------------+-------------------------------------+

Fertilization "application\_info.info" is a JSON object serialized as a string, such as:

+-----------------------------------------------+

| { |

| "product": { |

| "name": "optional name", |

| "unit": "kg|l", |

| "amount": <amount\_in\_product\_unit>, |

| "nutrients": { |

| "N": <amount\_in\_grams\_per\_unit>, |

| "P2O5": <amount\_in\_grams\_per\_unit>, |

| "K2O": <amount\_in\_grams\_per\_unit>, |

| … |

| } |

| } |

| } |

+-----------------------------------------------+

The units of the "product" data table are in application\_info.info.product.unit" units.

# Access and Authentication

Farmers MUST have an account setup on an advisor service in order to authenticate and access API endpoints. The service implementor is responsible for the creation of accounts; it is not covered in the service template specifications.

Unless specifically documented in the OpenAPI specifications, all API calls must include credentials in form of Bearer authentication (also called token authentication). Clients can obtain an access token on behalf of their user from the service's authorization server (see ATLAS service pairing).

Some of the information held by advisor services may be considered sensitive from a GDPR perspective. The service's authorization server SHOULD request the client's end-user consent at service pairing time in order to deliver an access token.

# Dynamic Behaviour

The purpose of the diagrams in this section is to illustrate communication patterns, more complex than plain request/response API calls, that involve several interactions and/or asynchronous behaviour. Even though a sequence diagram representation is used, the diagrams are by no means to be interpreted as UML Sequence Diagrams. Specifically, in the spirit of focusing on functional behaviour and readability, error handling is deliberately not covered in the diagrams.

## Fertilisation Lifecycle

The "Fertilisation lifecycle" in ATLAS typically operates in a larger context involving three services and an end-user application such as an FMIS which, in collaboration, provide an end-to-end solution from algorithmically computed advices to actual operations performed on a field.

A picture containing chart

Description automatically generated

## Field Registration

There are two methods for determining whether a field registration has completed (successfully or unsuccessfully): by polling the registration\_info endpoint or by notification.

### Polling

After requesting a field registration, the client polls registration\_info endpoint at regular intervals until the returned status is CONFIGURATION\_REQUIRED, READY or FAILED.

A screenshot of a computer

Description automatically generated with low confidence

A CONFIGURATION\_REQUIRED status indicates that a user action is required on the user interface of the advisor. By convention, in case of CONFIGURATION\_REQUIRED status, the advisor MAY return a browser URL to the location where a user can provide additional parameters (as required for a specific service implementation). It is the user interface of the client application originating the request to display an appropriate message to the end-user with a clickable link to the provided URL.

### Notification

If a notification URL was supplied on the register\_field endpoint, it will be invoked by the advisor when the registration status changes.

A screenshot of a computer

Description automatically generated with low confidence

Advanced advisors may have the ability to monitor conditions on registered fields in the background and proactively generate an advice recommendation request on the notification URL provided at field registration.

Shape

Description automatically generated with medium confidence

Advisors MUST invoke the notification URL supplied by the client with an HTTPS POST command. The payload will be identical to the one that would be returned by a request to the registration\_info endpoint.

## Advice Preparation

There are two methods for determining whether an advice preparation has completed (successfully or unsuccessfully): by polling the advice\_info endpoint or by notification.

### Polling

After requesting an advice preparation, the client polls the advice\_info endpoint at regular intervals until the returned status is READY or FAILED.

The status may be CONFIGURATION\_REQUIRED which indicates that a user action is required on the user interface of the advisor. By convention, in case of CONFIGURATION\_REQUIRED status, the service SHOULD return a browser URL to the location where a user can provide additional parameters (as required for a specific service implementation). It is the user interface of the ADIS originating the request to display an appropriate message to the end-user with a clickable link to the provided URL.

A screenshot of a computer

Description automatically generated with low confidence

### Notification

If a notification URL was supplied on the prepare\_advice endpoint, it will be invoked by the advisor when the preparation status changes.

A screenshot of a computer

Description automatically generated with low confidence

Advisors MUST invoke the notification URL supplied by the client with an HTTPS POST command. The payload will be identical to the one that would be returned by the advice\_info request.

## General Comments on Notifications

Services must provide best efforts to deliver notifications. A notification is considered successful if the target returns an http result code 2XX.

Errors may occur during notification delivery. Depending on the type of error, services must react in different ways:

i) Network error - the connection to the client's host (from notification URL) cannot be established. The service MUST retry a certain number of times. The number of retries and possible backoff strategy is left at the discretion of the service implementer.

ii) Server errors (5XX result code) – these errors are potentially transient. The same strategy as for Network errors SHOULD be applied.

iii) Client errors (4XX result code) – typically when the notification URL is invalid, or the authentication is invalid/expired. 4XX errors should never be sent for transient client-side conditions and therefore services SHOULD NOT attempt retries.

Upon an excessive number of errors, services MAY give up further notification attempts. In that case, clients can only retrieve completion information via polling.