Service Template Specification

STS id irrigation\_planning

Version 0.1.1

Status PUBLISHED

Author(s) I.D. Tsakmakis, V. Pisinaras, C. Brogi (LRI,FZJ)

**The "irrigation\_planning" Service Template**

Abstract

This document describes the specifications for irrigation\_planning services whose purpose is to provide irrigation maps, for a selected field, over the next few days, based on historic and forecast weather data. Irrigation\_planning services do not take into consideration the water supply system infrastructure (main and lateral pipelines or canals etc.) or the equipment that will implement the suggested irrigation (drip, sprinkler, micro-sprinkler etc.). They are based on soil water balance and/or crop models of varying complexity levels (from empirical to mechanistic), that are fed with data related to plant-soil-atmosphere continuum and derive advice on required net irrigation amount, which successively can be used by an irrigation management application.

Table of Contents

[1 Introduction 3](#_Toc87439973)

[2 Terminology 3](#_Toc87439974)

[3 Irrigation Planning Usage Scenarios 3](#_Toc87439975)

[3.1 Vineyard Irrigation 3](#_Toc87439976)

[4 Service Template Functions 4](#_Toc87439977)

[4.1 Irrigation Planning Functions 5](#_Toc87439978)

[4.1.1 Prepare Planning 5](#_Toc87439979)

[4.1.2 Get Planning Status 5](#_Toc87439980)

[4.1.3 Cancel Planning 6](#_Toc87439981)

[4.2 Irrigation Implementation Functions 6](#_Toc87439982)

[4.2.1 Get Irrigation Info 6](#_Toc87439983)

[5 Data Formats 6](#_Toc87439984)

[5.1 Product Application File Format 6](#_Toc87439985)

[6 Access and Authentication 8](#_Toc87439986)

[7 Dynamic Behaviour 8](#_Toc87439987)

[7.1 Irrigation Lifecycle 9](#_Toc87439988)

[7.2 Planning Completion 9](#_Toc87439989)

[7.2.1 Polling 9](#_Toc87439990)

[7.2.2 Notification 10](#_Toc87439991)

# Introduction

Shortage of plant available water may strongly reduce crop yield or cause crop production failure. Thus, in certain regions and for certain cultivations, irrigation is essential to achieve optimum crop yield. However, the “when” and “how much” to irrigate may vary significantly depending on crop variety, soil characteristics, local weather conditions, and on the existing soil water amount available for the plant.

An irrigation service determines the amount of water that is required to maintain the water levels within the rooting zone profile in satisfactory levels. This amount should not be miss-interpreted as the water needed to refill rooting zone back to field capacity. The satisfactory levels may vary significantly depending on the crop type or among a crop’s different growing stages, and can represent one of the following irrigation strategies: "no-water-stress", "regulated-mild water stress", "regulated-severe water stress", "sustained water stress" etc.

Farmers may use different irrigation services/options depending on the accuracy they want to achieve, their budget, the availability of ancillary data.

# 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 term "irrigation\_planning" refers to "irrigation\_planning service template". Instead, "irrigation\_planning service" is equivalent to "a service implementing the irrigation\_planning service template".

# Irrigation Planning Usage Scenarios

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

## Vineyard Irrigation

Wine grape vines are a delicate crop, that can produce new vegetation and ripening grapes at the same time, with substantially different water needs at the various growing stages.

An irrigation planning service can be designed to promote regulated deficit irrigation over full irrigation, in certain growing stages (e.g., veraison stage), aiming to control the vegetation growth and boost grapes ripening.

The service receives initially a request to prepare an irrigation map for a vineyard plantation. Based on the vines growing stage (linked to growing degree days after budburst), the soil's properties and previously implemented irrigations, the irrigation planning service determines the optimal irrigation amounts, for the next few days, that will maintain the vineyard plants under the proper water status (well irrigated or mildly stressed). The number of days that the service will run for, can be either passed as an argument during the initial request or can be based on the available weather forecast days.

These application maps are delivered on-demand to an irrigation management application vendor that is responsible (a) to decide the irrigation day; (b) converting the data to a format understandable by the equipment that will implement the suggested irrigation.

# Service Template Functions

This section provides a very high-level summary of the irrigation\_planning service template functions:

Irrigation Planning Functions

Prepare Planning

Get Planning Status

Cancel Planning

Irrigation Implementation

Get Irrigation 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.

irrigation\_planning 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.

## Irrigation Planning Functions

These functions relate to the development of irrigation planning. A planning may be requested to calculate the optimal water amounts that should be applied on the field for the next few days.

### Prepare Planning

This function allows for the preparation of the background information that may be necessary for the development of the irrigation planning. The preparation may take from some seconds to hours, depending on the complexity of the models integrated into the irrigation\_planning service and the corresponding data requirements.

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

| prepare\_planning |

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

| Inputs | field urn, planning\_days, |

| | notification URL |

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

| Outputs | planning info |

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

Planning\_days correspond to the number of days that the prepare\_planning service MUST run and can be provided by the irrigation management application vendor who sent the initial request. If not provided, the prepare\_planning function will be executed for a number of days equal to the weather forecast data. If the requested planning\_days number is larger than the available forecast days, the prepare\_planning service will return an error message.

irrigation\_planning services may retrieve any relevant information to preparing their planning from the field\_data service (e.g., boundaries, current crop, previous irrigations, sowing date etc.).

Clients may poll for the status of a planning (see Get Planning Status) or be notified of completion (successful or unsuccessful) if they supplied a notification URL.

### Get Planning Status

This function returns status information about a planning preparation.

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

| get\_planning\_status |

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

| Inputs | planning id |

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

| Outputs | status |

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

The planning status is one of "PREPARING", "READY", "CANCELED", "FAILED".

### Cancel Planning

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

| cancel\_planning |

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

| Inputs | planning id |

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

| Outputs | - |

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

## Irrigation Implementation Functions

Product application functions deal translating planning into irrigation maps that can be supplied on demand, typically by an irrigation management application vendor.

### Get Irrigation Info

This function may only be performed on irrigation plans whose status is "READY". It returns a download URL to the irrigation map and the total estimated amount of water required to implement the irrigation. All values are in millimetres of water. Please refer to "5.1 Irrigation Implementation File Format" for detailed specification of the downloaded irrigation map file format.

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

| get\_irrigation\_info |

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

| Inputs | planning id |

| | |

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

| Outputs | download URL, |

| | irrigation amount |

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

# Data Formats

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

## Product Application File Format

Irrigation Maps 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 |

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

| ROW | COLUMN | VALUE |

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

| | type | "irrigation\_planning" |

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

| | 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 irrigation\_planning 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".

An irrigation Map GeoPackage MUST contain the following additional tables:

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

| TABLE gpkg\_contents |

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

| ROW | COLUMN | VALUE |

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

| … |

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

| | table\_name | "day1" |

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

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

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

| | table\_name | "day1\_info" |

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

| | data\_type | "attributes" |

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

| | table\_name | "dayn" |

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

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

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

| | table\_name | "dayn\_info" |

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

| | data\_type | "attributes" |

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

The number of days n is defined either within the prepare\_planning function or if not is based on the available weather forecast days.

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

| TABLE product\_info |

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

| ROW | COLUMN | VALUE |

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

| | name | "irrigation\_amount" |

| | unit | "mm" |

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

# Access and Authentication

Irrigation management application vendors MUST have an account setup on a irrigation\_planning 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).

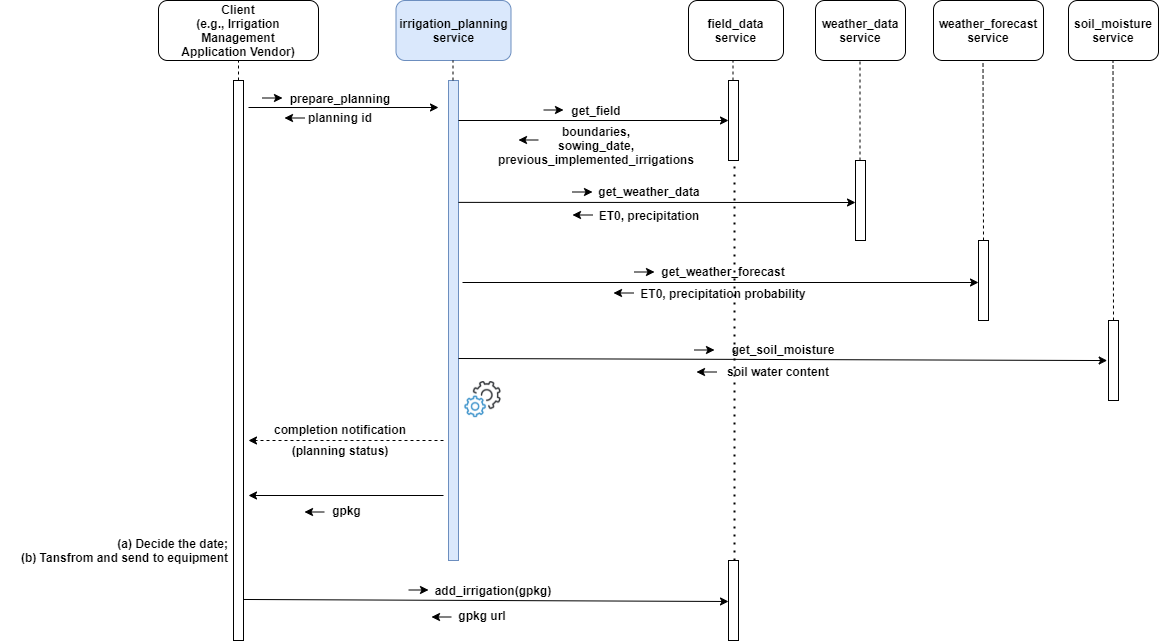
Some of the information held by irrigation\_planning services may be considered to be sensitive from a GDPR perspective. The service's authorization server SHOULD request the client's end-user consent at service pairing time in order to deliver an access token.

# 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.

## Irrigation Lifecycle

The "irrigation lifecycle" in ATLAS typically operates in a larger context involving up to five services and an end-user application such as an irrigation management application vendor which, in collaboration, provide an end-to-end solution from algorithmically derived irrigation maps to actual irrigation applications performed on a field.

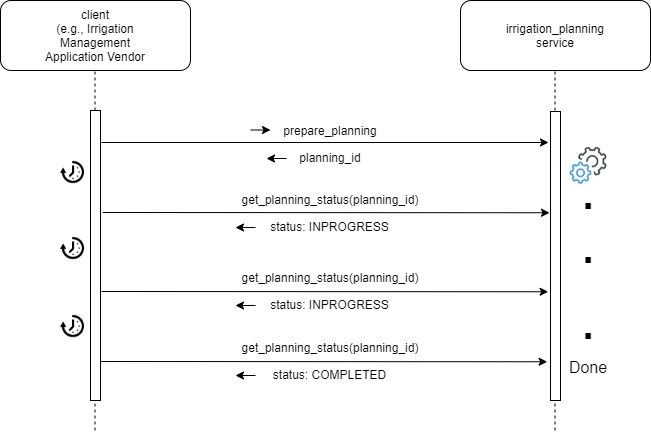


## Planning Completion

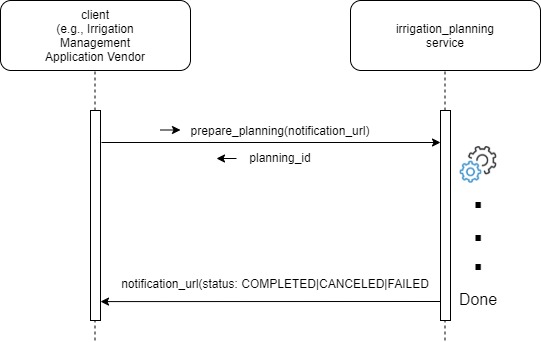
There are two methods for determining whether a planning preparation has completed (successfully or unsuccessfully): by polling the get\_planning\_status or by notification.

### Polling

After requesting a planning preparation, the client polls get\_planning\_status at regular intervals until the returned status is "READY", "CANCELED", or "FAILED".



### Notification



If a notification URL was supplied on the prepare\_planning function, it will be invoked by the irrigation\_planning when the preparation status changes.

Service 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 get\_planning\_status request.

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.