ARROWHEAD

Document title
occupancyData HTTP
Date
2025-10-19
Author
Mattias Öhman
Contact
mathma-1@student.ltu.se

Document type
IDD
Version
1.0.0
Status
RELEASE
Page
1 (8)

occupancyData HTTP

Interface Design Description

Abstract

This Interface Design Description (IDD) defines the HTTP implementation of the occupancyData service. The interface uses secure HTTPS with JSON payloads to transmit aggregated parking occupancy information derived from boolean sensor inputs within a single parking lot.

Contents

1 Overview 3

2 Service Operations 4
2.1 operation getOccupancyData 4

3 Data Models 5
3.1 struct occupancyRequest 5
3.2 struct occupancyResponse 5
3.3 Primitives 5

4 References 7

5 Revision History 8
5.1 Amendments 8

1 Overview

This document describes the HTTP/JSON interface for the occupancyData service produced by the OccupancySystem. The interface allows other systems, such as the DynamicPricingSystem, to request the current aggregated occupancy data for a single parking lot. Communication occurs over HTTPS with JSON payloads.

Profile type	Type	Version
Transfer protocol	HTTPS	1.1
Data encryption	TLS	1.3
Encoding	JSON	RFC 8259 [**?**]
Compression	N/A	-
Semantics	SensML	RFC 9100
Ontology	N/A	-

Table 1: Communication profile for the occupancyData service interface

This document provides the Interface Design Description IDD to the *occupancyData – Service Description* document. For further details about how this service is meant to be used, please consult that document.

The rest of this document describes how to realize the occupancyData service interface in detail, both in terms of its operations (Section 2) and its information model (Section 3).

2 Service Operations

2.1 POST /occupancyData

Operation: getOccupancyData
Input: occupancyRequest
Output: occupancyResponse

Called by the DynamicPricingSystem to obtain aggregated occupancy data for the parking lot. The request and response are encoded in JSON format. The POST method is used to allow structured input, including optional filters such as timestamp or spot type.

```
POST /occupancyData HTTP/1.1


{
  "timestamp": "2025-10-14T08:30:00Z",
  "spotType": "EV"
}
```

Listing 1: Example request to getOccupancyData

```


{
  "totalSpots": 50,
  "occupiedSpots": 42,
  "occupancyRatio": 0.84,
  "timestamp": "2025-10-14T08:30:00Z",
  "metadata": {
    "lotName": "MainLot",
    "spotType": "EV"
  }
}
```

Listing 2: Example response from getOccupancyData

3 Data Models

3.1 struct occupancyRequest

Field	Type	Description
timestamp	DateTime	Optional timestamp filter for the data snapshot
spotType	Name	Optional filter by parking spot type (e.g., “EV”, “regular”)

3.2 struct occupancyResponse

Field	Type	Description
totalSpots	Integer	Total number of parking spots monitored
occupiedSpots	Integer	Number of currently occupied spots
occupancyRatio	Float	Ratio between occupied and total spots (0.0–1.0)
timestamp	DateTime	Time when data was aggregated
metadata	Metadata	Additional descriptive information (e.g., lot name, spot type)

3.3 Primitives

Type	Description
Name	String identifier for the parking spot or type
DateTime	Timestamp in UTC format
Integer	Whole number for counting parking spots
Float	Floating-point number for fractional values
Metadata	Object containing key–value descriptive data

3.3.1 alias Name = String

A string identifier that represents a parking spot type or category. Example: ”EV” or ”regular”.

3.3.2 alias DateTime = String

Pinpoints a moment in time in the format of ”YYYY-MM-DD HH:mm:ss”, where ”YYYY” denotes year (4 digits), ”MM” denotes month starting from 01, ”DD” denotes day starting from 01, ”HH” denotes hour in the 24-hour format (00-23), ”MM” denotes minute (00-59), ”SS” denotes second (00-59). ” ” is used as separator between the date and the time. An example of a valid date/time string is ”2020-12-05 12:00:00”

3.3.3 alias Integer = Number

Represents an integer number used for counting objects. Example: 42.

3.3.4 alias Float = Number

Represents a floating-point number, for example: 0.84.

3.3.5 alias Metadata = Object<String>

A JSON object containing string-based key–value pairs, such as lot names or types. Example: ”lotName”: ”MainLot”.

4 References

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2025-10-14	1.0.0	Initial release	Mattias Öhman