ARROWHEAD

Document title
GetAllLamps
Date
2025-01-09
Author
Juliana Sánchez
Contact
sanjul-4@student.ltu.se

Document type
SD
Version
X.Y.Z
Status
RELEASE
Page
1 (10)

GetAllLamps

Service Description

Abstract

This is the Service Description (SD document) for the ”getAllLamps model” service according to the Eclipse Arrowhead documentation structure.

Contents

1 Overview

This document describes the ”getAllLamps” service, which sends and receives informations of all lamps that changed status after the last time it was requested.

The rest of this document is organized as follows. In Section 2, we describe the abstract message operations provided by the service. In Section 3, we end the document by presenting the data types used by the mentioned operations.

1.1 How This Service Is Meant to Be Used

This service is used to have the updated status in order to turn off or on a lamp.

1.2 Important Delimitations and Dependencies

This service is specifically designed to work within a smart lighting system architecture, where lamps are controlled and monitored centrally. It assumes the existence of the following dependencies:

- A centralized lamp database (‘lampDB‘) that maintains the current status and metadata of all lamps.
- Reliable network communication between the street lighting system and the controller.
- The presence of a structured messaging protocol for exchanging data between components, ensuring compatibility with Arrowhead-compliant systems.

Furthermore, the service does not handle:

- Error recovery or retry mechanisms in case of network failures; such functionality must be implemented at the system level.
- Initial setup or configuration of the lamps in the system. It assumes all lamps are already registered in the database.
- Physical hardware issues, such as faulty lamps or communication modules, which must be managed by maintenance protocols external to this service.

2 Service Interface

This section describes the interfaces to the ”getAllLamps” service. In particular, each subsection names an abstract operation, an input type and an output type, in that order. The input type is named inside parentheses, while the output type is preceded by a colon. Input and output types are only denoted when accepted or returned, respectively, by the interface in question.

All abstract data types named in this section are defined in Section 3.

The following interface operations are available.

2.1 operation getAllLamps () : lampsList: List <LampResponseDTO>

The getAllLamps operation is used by the Controller to transmit the statuses of all the lamps.

2.2 operation getAllLampServiceOrchestrationAndConsumption ()

The getAllLampServiceOrchestrationAndConsumption operation is used to request the statuses of all the lamps from the Street Lighting System to the Controller.

3 Information Model

Here, all data objects that can be part something the ”getAllLamps” Service provides to the hosting System are listed in alphabetic order. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is used to denote a collection of named fields, each with its own data type. As a complement to the explicitly defined types in this section, there is also a list of implicit primitive types in Section 3.2, which are used to represent things like hashes and identifiers.

3.1 struct LampResponseDTO

This structure is used to store and send all updated lamps as List¡LampResponseDTO¿. It determines if the lamp was updated making sure the ”updated” parameter of the lamp object is set to true when getting all the lamps from the lampDB. This parameter is then set to false.

Field	Type	Description
id	int	Lamp ID.
status	int	Lamp status (1=ON, 0=OFF).

3.2 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

Type	Description
Address	A string representation of the address
Boolean	One out of `true` or `false`.
Interface	Any suitable type chosen by the implementor of the service.
DateTime	Pinpoints a specific moment in time.
List<A>	An *array* of a known number of items, each having type A.
Name	A string identifier that is intended to be both human and machine-readable.
PortNumber	Decimal number in the range of 0-65535
Version	Specifies a service version.

4 References

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2020-12-05	X.Y.Z		Tanyi Szvetlin
2	2021-07-14	X.Y.Z	Minor updates	Jerker Delsing
3	2022-01-10	X.Y.Z	Minor updates	Jerker Delsing

5.2 Quality Assurance

No.	Date	Version	Approved by
1	2022-01-10	X.Y.Z	