Service Description (SD) System Registry Administration

**Abstract**

This document describes an Arrowhead compliant System Registry Administration service. System Registry Administration is used to list and manage system metadata in an Arrowhead cloud. This allows getting metadata about systems in runtime to help facilitate runtime governance. The administration service allows reading and updating metadata of all systems in a local cloud.

Table of Contents

[1. Service Description Overview 3](#_Toc105596969)

[1.1. Significant Prior Art 3](#_Toc105596970)

[1.2. How This Service Is Meant To Be Used 3](#_Toc105596971)

[1.3. Important delimitations and dependencies 3](#_Toc105596972)

[2. Service Interface 3](#_Toc105596974)

[2.1. Operation Register (RegistrationItem) 4](#_Toc105596979)

[2.2. Operation Unregister (Instance) 4](#_Toc105596981)

[2.3. Operation Lookup(MatchingInfo): InstanceList 4](#_Toc105596982)

[2.4. Service Discovery Sequence 4](#_Toc105596984)

[3. Information Model 5](#_Toc105596985)

[3.1. Struct RegistrationItem 5](#_Toc105596986)

[3.2. Struct Instance 5](#_Toc105596987)

[3.3. Struct MatchingInfo 5](#_Toc105596988)

[4. Non-functional Requirements 6](#_Toc105596989)

[5. References 6](#_Toc105596990)

[6. Revision history 6](#_Toc105596991)

[6.1. Amendments 6](#_Toc105596992)

[6.2. Quality Assurance 6](#_Toc105596993)

1. Service Description Overview

This document describes an Arrowhead compliant System Registry Administration service. The system registry administration service is to list and manage system metadata in an Arrowhead cloud which is crucial for day-to-day operations and runtime governance. The metadata contains information about the identity of the system, the owner and vendor of the system, SLAs, what service instances (service IDs) are provided by the system and more. The administration service allows reading and updating metadata of all systems in a local cloud.

# Significant Prior Art

The need for an inventory of systems in an IT environment is not new. However, with modern Microservice Architectures (MSA) the number of systems, services and APIs deployed to an environment has increased. Furthermore, the autonomy of teams in a large organization leads to this system setup being ever changing each day. This has led to a lot of developments in the field of system inventories, also known as software catalogs.

There exist several systems which solves this problem and, thus, provides the system discovery service among others. For example, Backstage by Spotify (Backstage Project Authors, 2022) is a software catalog and a developer portal that allows organizations to track ownership and system composition. The system model of Backstage also strongly resembles the system composition model of Arrowhead Framework.

There is also some overlap between the problem the system discovery service solves and API management. It is not uncommon for API management tools to include owner and vendor of the APIs and similar metadata, e.g., SLAs. An example of such tools could be the WSO2 API Management.

# How This Service Is Meant to Be Used

The service is meant to be used to manage system metadata of systems in an arrowhead local cloud that cannot manage it themselves due to being constraint devices, due to organizational policies enforcing metadata to be registered manually or due to registering system metadata using a third-party component, e.g., a source code scraping tool or similar. The service may also be used by an administrative user interface which enables browsing systems in the local cloud and thus enabling runtime governance.

# Important delimitations and dependencies

The system as a concept is not used for service operations, i.e., in the process to exchange information between systems. To discover services and how to access them (their address) is the role of the service registry. The system registry administration service only holds and serves the relationship between a system identity and provided service identities, i.e., it lists the service ID of provided services.

1. Service Interface

The abstract Service Registry Administration interface has five operations for managing the metadata of systems in the Arrowhead local cloud. The operations are the standard create, read, update and delete (CRUD) plus an extra list all systems operation.

# Operation ListSystems()

The list systems operation is used to discover what systems are available without any further knowledge about the local cloud. It can be used to traverse the metadata of all systems or to simply an overview in a consuming management user interface. It takes no input and must at least return a list of identifiers. An implementing service may want to return either the complete metadata of all systems or a subset of the metadata.

# Operation CreateSystem (SystemID, SystemMetadata)

The create operation is used to register a new system ID. This could also be done by the system itself using the System Discovery service. However, due to organizational policies or constrained devices it may be necessary to manually register a system before being deployed to the cloud and this is the purpose of this operation.

An implementing service may require all or a subset of the system metadata as input. If the metadata is not required, then the create system operation becomes a way to reserve a system ID in a local cloud and the metadata can then be provided using the update metadata operation.

# Operation ReadSystemMetadata (SystemID)

The read metadata operation is used to retrieve the currently registered metadata of a system. Given the systemID, the operation looks up the latest stored data of the system and sends it to the requester.

# Operation UpdateSystemMetadata (SystemID, SystemMetadata)

The update metadata operation is used by the owner of a system that want to modify existing data for a system.

# Operation DeleteSystem (SystemID)

The delete system operation is used to deregister a system. Depending on organizational policies this could be done after a system has been removed from the local cloud or not at all to, instead, maintain history. In the case of maintaining history, the deletion could instead be replaced by a “decommissioned” system metadata field that the system is updated with after being removed from the local cloud.

# System Lookup Sequence

# System Metadata Source Code Scraper Sequence

The following sequence shows a scenario where metadata about systems are stored in a version control system, e.g. git, and a “scraper” scans the repositories of the version control system and upserts the metadata using system discovery administration. This is a common scenario when an organization has most of their systems developed in-house or simply deploys their software, both their own and third-party, using automated pipelines and infrastructure as code (IaC).

Table

Description automatically generated

1. Information Model

# SystemID

The system identifier is a unique identifier for a system. An implementing service may choose how this identifier is represented but common techniques could be a simple string identifier, a sequential number, a URI or a public certificate.

# SystemMetadata

The data structure used to represent the metadata about a system. The metadata must contain a human readable name of the system. An implementing service could add more required metadata. Examples of metadata to consider are owner, vendor, SLAs, link to documentation, disaster recovery objectives, data classification (e.g. if the system handles personal data), GDPR related information etc.

1. Non-functional Requirements

No non-functional requirements have been defined.

1. References

No references have been identified.

1. Revision history

# Amendments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Version | Subject of Amendments | Author |
| 1 | 2022-09-16 | 0.1 | Initial draft of specification | David Rutqvist |
| 2 | 2022-10-17 | 0.2 | Updated operations | Per Olofsson |
| 3 | 2022-10-24 | 0.3 | Added scraper use case sequence | David Rutqvist |

# Quality Assurance

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Date | Version | Approved by |
| 1 |  |  |  |
| 2 |  |  |  |