# Orchestrator Core System

## System Description

**Abstract**
This document provides system description for the **Orchestrator Core System**.

# Contents

# 1 Overview

This document describes the Orchestrator Core System, which exists to find matching providers for the consumer's specification within an Eclipse Arrowhead Local Cloud (LC) and in other Arrowhead clouds by collaborating with other Core Systems. This can be achieved by using stored matching rules or applying a more dynamic strategy using the Service Registry Core System. This mandatory Core System provides the data storage functionality for the information related to matching rules and optionally provider reservation.

The rest of this document is organized as follows. In Section 1.1, we reference major prior art capabilities of the system. In Section 1.2, we describe the intended usage of the system. In Section 1.3, we describe fundamental properties provided by the system. In Section 1.4, we describe delimitations of capabilities of the system. In Section 2, we describe the abstract service operations produced by the system. In Section 3, we describe the security capabilities of the system.

## 1.1 Significant Prior Art

The strong development on cloud technology and various requirements for digitisation and automation has led to the concept of Local Clouds (LC).

*"The concept takes the view that specific geographically local automation tasks should be encapsulated and protected."* [1]

An orchestration system is a central component in any Service-Oriented Architecture (SOA). In applications the use of SOA for a massive distributed System of Systems requires orchestration. It is utilised to dynamically allow the re-use of existing services and systems in order to create new services and functionality.

## 1.2 How This System Is Meant to Be Used

Orchestration is a mandatory core system of Eclipse Arrowhead LC and is responsible for finding and pairing service consumers and providers.

An application that want to consume a service should ask the Orchestrator to find one or more accessible providers that met the necessary requirements (including Quality-of-Service requirements). The Orchestrator returns the information (address, port, context path, tokens) that the application needs to consume the specified service.

## 1.3 System functionalities and properties

### 1.3.1 Functional properties of the system

Orchestrator solves the following needs to fulfill the requirements of orchestration.

- Enables the application and other core systems to find the appropriate providers to consume their services.
- Enables the Workflow Choreographer core system to find appropriate providers for its executors to consume the services.
- Enables the Plant Description Engine core system to create and remove flexible matching rules.
- Enables the Gatekeeper core system to handle provider reservations in the name of an other cloud.

### 1.3.2 Non functional properties of the system

The Orchestrator implements certain authentication capabilites, meaning that Orchestrator makes decision whether a given system has right to use its services (i.e. only the Workflow Choreographer can orchestrate providers for some other systems).

### 1.3.3 Data stored by the system

In order to achieve the mentioned functionalities, Orchestrator is capable to store the information set described by figure 1.

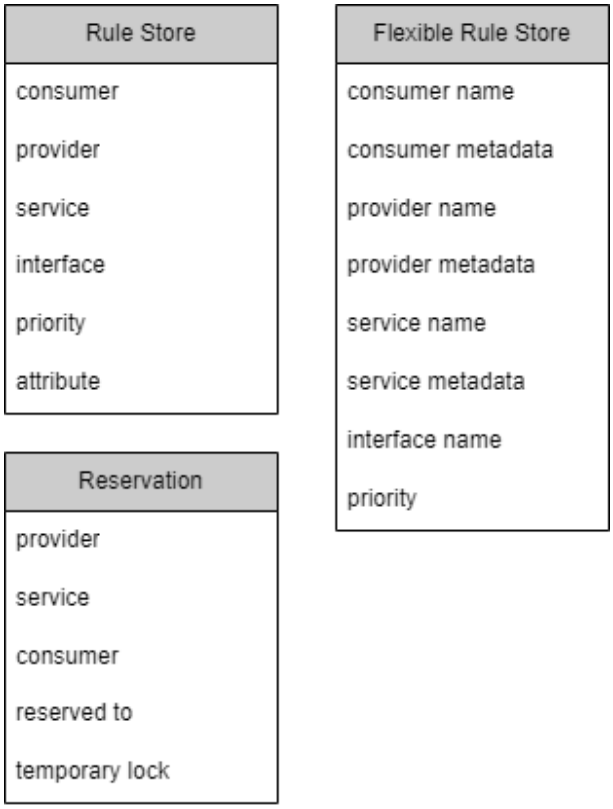


Figure 1: Overview of data stored by Orchestrator Core System.

## 1.4 Important Delimitations

While Orchestrator Core System is capable of using two types of matching rules (fix rules where every pairing is referencing a consumer and a provider instance, or flexible rules where both consumer and provider can be described with their names and/or their settings), can't use both simultaneously. Administators have to choose between the two modes when deploying the system.

Also, the Orchestrator can't work alone. It needs a Service Registry Core System and a Authorization Core System to operate. To enable inter-cloud orchestration, it also needs at least a running Gatekeeper Core System and depending on networking a Gateway Core System, too. To enable Quality-of-Service requirement matching, a QoS Monitor Core System is needed.

# 2 Services produced

## 2.1 service echo

The purpose of this service is to test the system availability. The service is offered for both application and core systems.

## 2.2 service orchestration-service

The purpose of this service is to get access information for providers that provides the required service and meets the additional requirements. The service is offered for both application and core systems.

## 2.3 service orchestration-service-by-proxy

The purpose of this service is to get access information for providers that provides the required service and meets the additional requirements. The main difference between this service and the previous one that here the consumer of the service is not the requester system. The service is offered only for the Workflow Choreographer core system.

## 2.4 service orchestration-service-by-id

With this service a consumer can get access information for all its necessary services based on the top priority stored matching rules. The service is offered for both application and core systems.

## 2.5 service orchestration-create-flexible-store-rules

The purpose of this service is to create a flexible matching rule. The service is offered only for the Plant Description Engine Core System.

## 2.6 service orchestration-remove-flexible-store-rule

The purpose of this service is to remove a flexible matching rule. The service is offered only for the Plant Description Engine Core System.

## 2.7 service orchestration-clean-flexible-store

The purpose of this service is to remove all flexible matching rules. The service is offered only for the Plant Description Engine Core System.

## 2.8 service orchestration-qos-enabled

The purpose of this service is to tell the requester whether the Orchestrator can handle Quality-of-Service requirements or not. The service is offered only for Gatekeeper Core System.

## 2.9 service orchestration-qos-reservations

The purpose of this service is to return all active provider reservations. The service is offered only for Gatekeeper Core System.

## 2.10 service orchestration-qos-temporary-lock

The purpose of this service is to lock a set of providers temporarily. The service is offered only for Gatekeeper Core System.

## 2.11 service orchestration-qos-confirm-reservation

The purpose of this service is to confirm a temporary provider reservation. All related not confirmed reservation are released. The service is offered only for Gatekeeper Core System.

# 3 Security

The security of Eclipse Arrowhead - and therefore the security of Orchestrator - is relying on X.509 certificate trust chains. The Arrowhead trust chain consists of three level:

- Master certificate: `arrowhead.eu`
- Cloud certificate: `my-cloud.my-company.arrowhead.eu`
- Client certificate: `my-client.my-cloud.my-company.arrowhead.eu`

For Arrowhead certificate profile see `https://github.com/eclipse-arrowhead/documentation`

# 4 References

[1] J. Delsing and P. Varga, *Local automation clouds*. Boca Raton: Taylor & Francis Group, 2017, p. 28. [Online]. Available: https://doi.org/10.1201/9781315367897

# 5 Revision History

## 5.1 Amendments

| No. | Date | Version | Subject of Amendments | Author |
|---|---|---|---|---|
| 1 | YYYY-MM-DD | 4.6.0 | | Xxx Yyy |

## 5.2 Quality Assurance

| No. | Date | Version | Approved by |
|---|---|---|---|
| 1 | YYYY-MM-DD | 4.6.0 | |