Orchestration

Service Description

Service ID: *"orchestration"*

Abstract

This document describes a variant of the Orchestration service.

ARTEMIS Innovation Pilot Project: Arrowhead
THEME [SP1-JTI-ARTEMIS-2012-AIPP4 SP1-JTI-ARTEMIS-2012-AIPP6]
[Production and Energy System Automation Intelligent-Built environment and urban infrastructure for sustainable and friendly cities]

Contents

1 Overview

This document describes the Orchestration Eclipse Arrowhead service, which provides Application Systems with orchestration information: where they need to connect to. The outcome of the Orchestration Service include rules that will tell the Application System what Service provider System(s) it should connect to and how (acting as a Service Consumer).

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

2 Service Interfaces

This section lists the functions that must be exposed by the Public Key service in alphabetical order. In particular, each subsection names an abstract interface, 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.

2.1 function Orchestration (Orchestration)

This function enables orchestration for application systems to use.

2.2 function OrchestrationStoreManagement (OrchestrationStoreManagement)

The function enables management of Store based orchestration rules.

3 Information Model

Here, all data objects that can be part of Orchestration service calls 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 ServiceRequestForm

This structure is used to check whether the consumer system can use a service from a list of provider systems.

Object Field	Value Type	Description
”requesterSystem”	System	Requester System.
”requestedService”	Service	Requested Service.
”securityRequirements”	Array<String>	Security requirements.
”metadataRequirements”	Object	Metadata requirements
”versionRequirement”	Number	Version requirements
”maxVersionRequirement”	Number	Max version requirements
”minVersionRequirement”	Number	Min version requirements
”preferredProviders”	Array<PreferredProvider>	Array of Preferred providers
”orchestrationFlags”	Object	Orchestration flags

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.

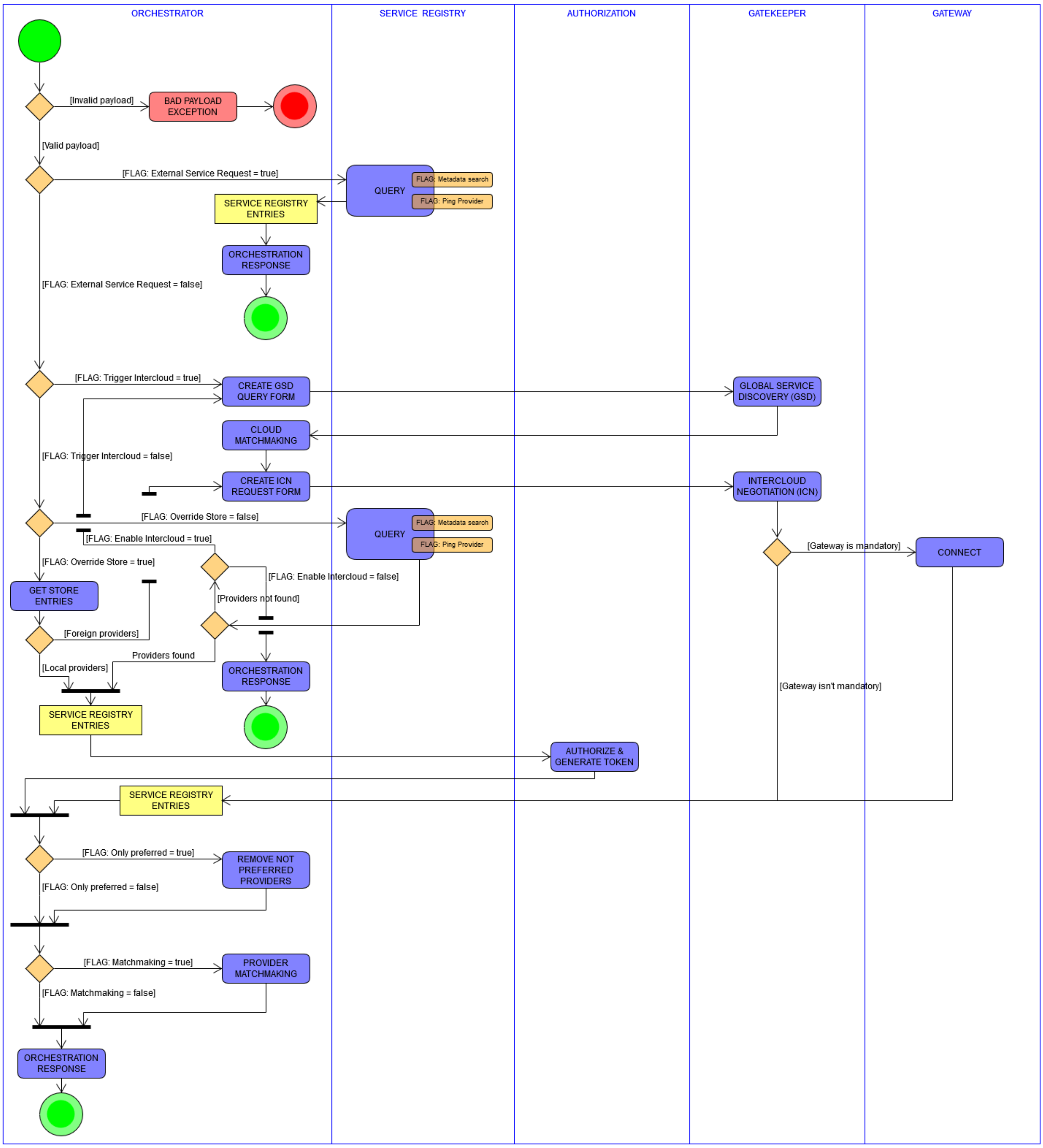

Figure 1: Information model as a UML activity diagram. Describes the process of Orchestration.

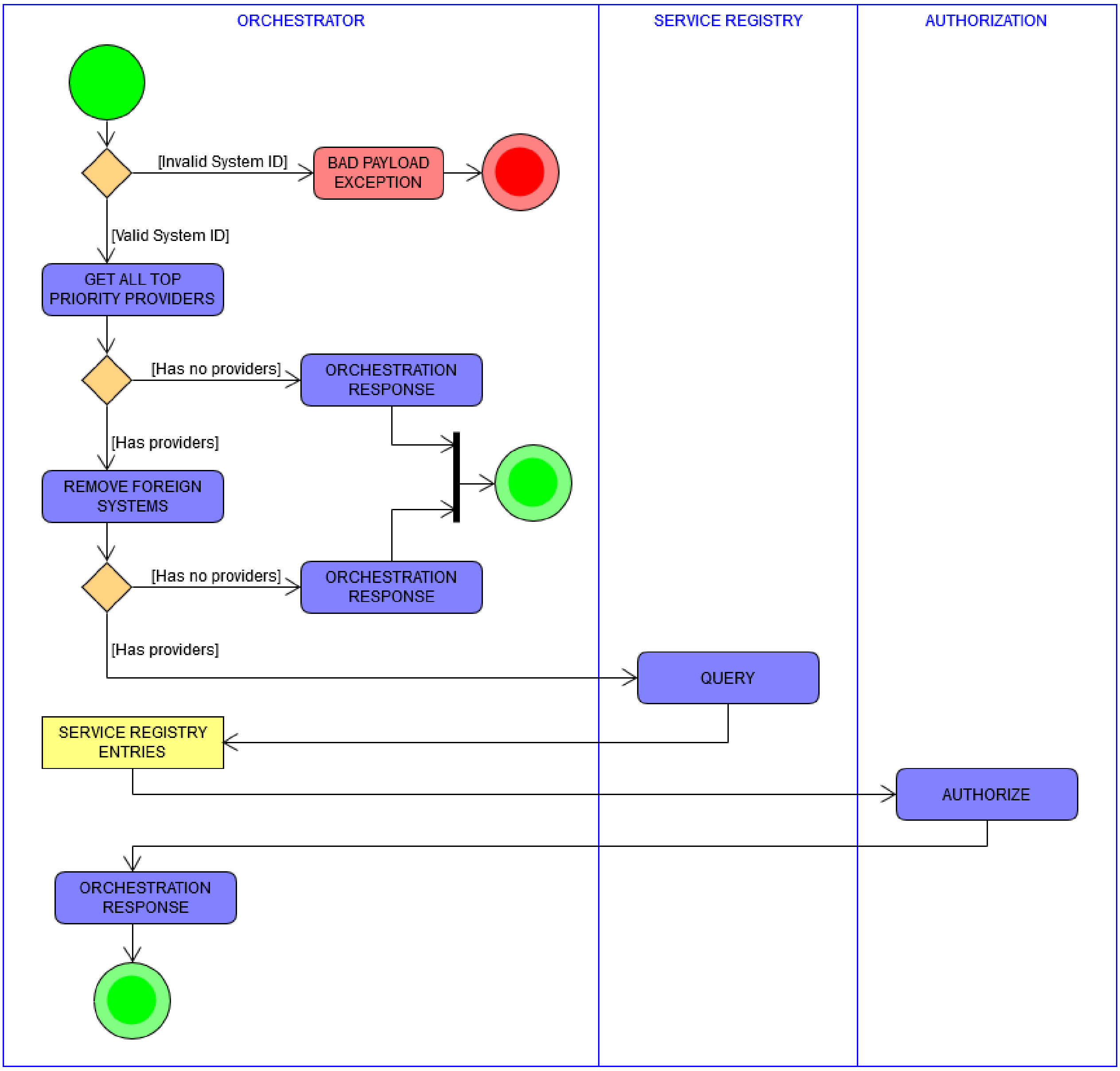

Figure 2: Information model as a UML activity diagram. Describes the process of Store Orchestration.

JSON Type	Description
Value	Any out of Object, Array, String, Number, Boolean or Null.
Object<A>	An unordered collection of [String: Value] pairs, where each Value conforms to type A.
Array<A>	An ordered collection of Value elements, where each element conforms to type A.
String	An arbitrary UTF-8 string.
Number	Any IEEE 754 binary64 floating point number [1], except for *+Inf*, *-Inf* and *NaN*.
Boolean	One out of `true` or `false`.
Null	Must be `null`.

4 References

[1] M. Cowlishaw, "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, July 2019. [Online]. Available: https://doi.org/10.1109/IEEESTD.2019.8766229

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2020-12-05	4.3.0		Tanyi Szvetlin

5.2 Quality Assurance

No.	Date	Version	Approved by
1	2021-01-29	4.3.0	Jerker Delsing