Orchestration Service IDD

**Abstract**

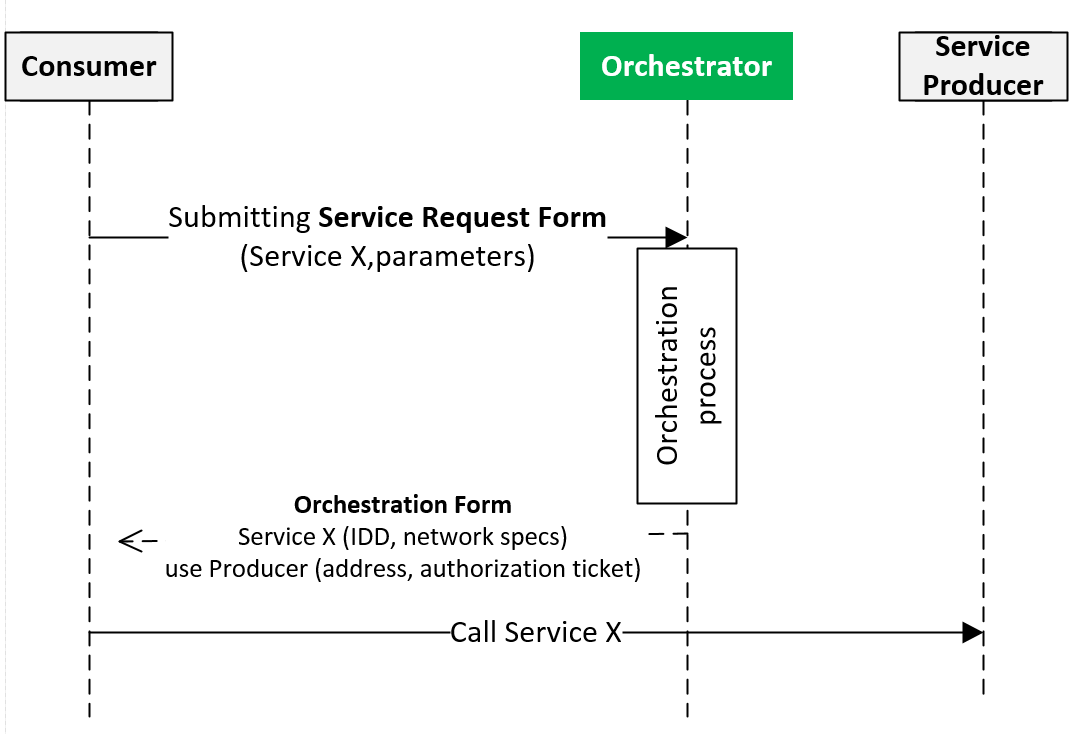
This document describes the interaction with the Advanced Orchestration Service using REST/HTTP with JSON payloads.

1. Introduction

This document describes the Orchestration Service and how it can be accessed using HTTP methods and JSON payloads.

1. Overview

This Service can be invoked with a simple REST call to the Orchestrator System to the specific Service URI.



1. Information model

The Orchestration Service is offered through 2 different interfaces. The default interface used by most Systems give options to the requester System to influence the Orchestration process, but also requires a complex request payload with mandatory fields. The 2nd interface is specifically implemented for low-level devices, where a request payload is not needed, and only default Orchestration Store based Orchestration process can happen. Table 1 describes these two interfaces.

Table 1 Function description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function | URL subpath | Method | Input | Output |
| Orchestration | “/orchestrator/orchestration” | POST | ServiceRequestForm | 200 OK,  BAD\_PAYLOAD,  DATA\_NOT\_FOUND |
| Orchestration | “/orchestrator /orchestration/ {systemName}” | GET | — | 200 OK,  BAD\_PAYLOAD,  DATA\_NOT\_FOUND |

The main Service interface is accessible via a POST request to the Service URI. The data is to be sent in JSON or XML.

The Orchestration URI is <http://LocalOrchestratorIP:port/orchestrator/orchestration> by default.

1. **Service Request Form**



Figure 1. The Service Request Form

|  |  |
| --- | --- |
| Field | Description |
| RequesterSystem: ArrowheadSystem | The ArrowheadSystem identifier that is requesting the orchestration. Mandatory field. |
| RequesterCloud: ArrowheadCloud | Not to be used by Application Systems. This field is only used when there is an inbound request from another Cloud. This can only be submitted by the Gatekeeper. |
| RequestedService: ArrowheadService | Not mandatory. When not filled out, the Orchestrator operate in the ”default configuration” mode. When filled out, other fields will affect how the orchestration will work – but the Service will be specified by the Consumer in advance. |
| OrchestrationFlags: Map<String, Boolean> | These flags can be set to tailor the orchestration process. |
| PreferredProviders | There can be a list from the Consumer stating which other Application Systems (even from other Clouds) it wishes to connect. If this field is filled out, the Orchestrator will try to facilitate these preferences. |
| RequestedQoS | This object will be presenting the Consumer’s selected QoS level. Not implemented. |

The Orchestration flags are optional in the payload, but have the following meaning if set (the color coding in Fig.1 suggests which flag affects which Core System):

|  |  |
| --- | --- |
| triggerInterCloud | Setting this flag will mean skipping the local orchestration process, and go straight to the inter-Cloud process with the Gatekeeper services (GSD + ICN). |
| externalServiceRequest | This flag can only be used by the Gatekeeper. It means that the Service Request is from another Cloud and an inbound inter-Cloud orchestration is happening. |
| enableInterCloud | The Consumer signals with this flag, that it accepts Provider systems from other clouds as well. The Orchestrator first tries to find a Provider inside the Local Cloud though. |
| metadataSearch | This flag is set if the Consumer wants filtering of Providers based on the submitted set of metadata key-value pairs. This happens in the Service Registry. |
| pingProviders | This is set if the Consumer wants the Service Registry to check whether the suitable Providers are online or their offer is invalid (forgot to revoke their entry). |
| overrideStore | If true, the Consumer wants dynamic orchestration, and not Orchestration Store-based. |
| matchmaking | This flag is set if the Consumer only wants one suitable Provider (the best). Requires custom matchmaker algorithm implemented in the Orchestrator (by default only the first one is returned). |
| onlyPreferred | If the Consumer can only accept preferred Providers (submitted in the SRF). |
| enableQoS | If true, QoS based filtering of the Providers is requested (not implemented). |

**Note:** Not all flags need to be in the Service Request Form, only the ones that are set to true, since all non-present flags will default to false value.

Service Request Form example:

{

"requesterSystem" : {

"systemName" : "client1",

"address" : "localhost",

"port" : 0,

"authenticationInfo" : "null"

},

"requestedService" : {

"serviceDefinition" : "IndoorTemperature",

"interfaces" : [ "json" ],

"serviceMetadata" : {

"unit" : "celsius"

}

},

"orchestrationFlags" : {

"onlyPreferred" : false,

"overrideStore" : true,

"externalServiceRequest" : false,

"enableInterCloud" : true,

"enableQoS" : false,

"matchmaking" : false,

"metadataSearch" : true,

"triggerInterCloud" : false,

"pingProviders" : false

},

"preferredProviders" : [ ],

"requestedQoS" : { },

"commands" : { }

}

1. **Orchestration Response**



Figure 2 The Orchestration Response

The Response consists of multiple Orchestration Forms.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Service: ArrowheadService | The exact Service that needs to be consumed. This response matches the ServiceRequestForm’s (if that was filled out), but it comes from the exact Service instance with proper metadata and interface identifier (only one). |
| Provider: ArrowheadSystem | The Application System description that will provide the Service. |
| ServiceURI: String | The base URL within the Service Provider that will contain the seeked Service. |
| Instruction: String | Optional additional data from the orchestration. |
| AuthorizationToken: String | The encrypted authorization token. |
| Signature: String | The signature from the Authorization System for the token (verifying the token issuing). |
| Warnings: List<OrchestratorWarnings> | Enumeration with values that signal certain things, such as the provider is from another Cloud, or the service offering might have expired already. |

Orchestration Response example:

{

"response" : [ {

"service" : {

"serviceDefinition" : "IndoorTemperature",

"interfaces" : [ "JSON", "XML" ],

"serviceMetadata" : {

"unit" : "celsius"

}

},

"provider" : {

"systemName" : "InsecureTemperatureSensor",

"address" : "0.0.0.0",

"port" : 8460

},

"serviceURI" : "temperature",

"warnings" : [ "TTL\_UNKNOWN" ]

} ]

}

1. Security

This Service can be configured to work insecured (plain HTTP) or through TLS with X.509 certificates.

In its secure mode, only those Application Systems can request orchestration that belong to its Local Cloud (identified and checked using the CN field of the certificate). The requesterSystem field inside the ServiceRequestForm must match with the certificate CN field.

1. Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Version | Subject of Amendments | Author |
| 1 | 2016-08-23 | 0.1 | First edition | Csaba Hegedűs |
| 2 | 2018-02-19 | M3 | Updated document to M3 | Csaba Hegedűs |
| 3 | 2018-05-23 | G4.0 | Updated to G4.0 | Zoltán Umlauf |

# Quality Assurance

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