orchestration-service

Service Description

Abstract
This document provides service description for the **orchestration-service** service.

Contents

1 Overview

This document describes the **orchestration-service** service that provides runtime (late) binding between application systems. Its primary purpose is to provide application systems with orchestration information: where they need to connect to. The outcome of the service includes data that will tell the application system what service provider system(s) it should connect to and how (acting as a service consumer). Such orchestration data include:

- Accessibility information details of a service provider (e.g network address and port),
- Details of the service instance within the provider system (e.g. base URL, IDD specification and other metadata),
- Authorization-related information (e.g. access token and signature),
- Additional information that is necessary for establishing connection.

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.

1.1 How This Service Is Meant to Be Used

The given system should consume the Service Registry Core System's **query** service to get information about the **orchestration-service** service. Using this information the system can request the **orchestration-service** service with an orchestration form about the service the system wants to consume.

The service can be used in two ways. The first one uses predefined rules (coming from the fix or flexible database store) to find the appropriate providers for the consumer. The second option is the dynamic orchestration in which case the service searches the whole local cloud (and maybe some other clouds) to find matching providers.

1.1.1 Fix Store Orchestration

- requester system is mandatory,
- requested service and all the other parameters are optional,
- if requested service is not specified, then this service returns the top priority local provider of all services contained by the Orchestrator store database for the requester system. if requested service is specified, then you have to define the service definition and exactly one interface (all other service requirements are optional). In this case, it returns all accessible providers from the Orchestrator store database that provides the specified service via the specified interface to the specified consumer.

1.1.2 Flexible Store Orchestration

- requester system is mandatory,
- requested service is mandatory, but just the service definition part, all other parameters of the requested service are optional,
- all other parameters are optional

1.1.3 Dynamic Orchestration

- requester system is mandatory,
- requested service is mandatory, but just the service definition part, all other parameters of the requested service are optional,
- all other parameters are optional

1.1.4 Orchestration flags

The requester can modify the **orchestration-service**'s behaviour using these flags. These are the available options:

- `enableInterCloud`: the service can search another clouds for providers if none of the local cloud providers match the requirements,

- `enableQoS`: the orchestration process will use Quality-of-Service requirements and provider reservation requests (if the Orchestrator supports it),
- `externalServiceRequest`: the service is called by the Gatekeeper Core System on behalf of a consumer from an other cloud,
- `matchmaking`: the service automatically selects exactly one provider from the appropriate providers (if any),
- `metadataSearch`: query in the Service Registry uses metadata filtering,
- `onlyIPAddressResponse`: the service only returns providers whose address is an IP address (not a domain name),
- `onlyIPv4AddressResponse`: the service only returns providers whose address is a version 4 IP address (not a domain name and not a version 6 IP address),
- `onlyIPv6AddressResponse`: the service only returns providers whose address is a version 6 IP address (not a domain name and not a version 4 IP address),
- `onlyPreferred`: the service filters the results with the specified provider list,
- `overrideStore`: services uses dynamic orchestration if this flag is true, otherwise it uses the orchestration store,
- `pingProviders`: the service checks whether the returning providers are online and remove the unaccessible ones from the results,
- `triggerInterCloud`: the service skipped the search in the local cloud and tries to find providers in other clouds instead.

1.1.5 Quality-of-Service requirements

If the Orchestrator Core System supports it, the requester can specify Quality-of-Service requirements when sends an orchestration request. It has to set the `enableQoS` flag to true and also provide a `qosRequirements` map. The possible keys of the map are the following:

- `qosAvgRespTime`: the maximum acceptable value of the average (mean) response time, in milliseconds,
- `qosJitterThreshold`: the maximum acceptable value of jitter, in milliseconds,
- `qosMaxPacketLost`: the maximum acceptable packet loss, in percent,
- `qosMaxRecentPacketLoss`: the maximum acceptable recent packet loss, in percent,
- `qosMaxRespTimeThreshold`: the maximum acceptable value of the response time, in milliseconds,

The requester can also specify commands for the Orchestrator Core System. Currently, only one command is supported (and only if both the system and the requester enables QoS): `qosExclusivity`. With this command the requester can ask exclusive access for a provider's service for a specified time (in seconds). Exclusive access only means that the Orchestrator Core System will not include this particular provider to any orchestration result until the exclusivity expires.

1.1.6 The orchestration process

Figure 1 describes the orchestration process.

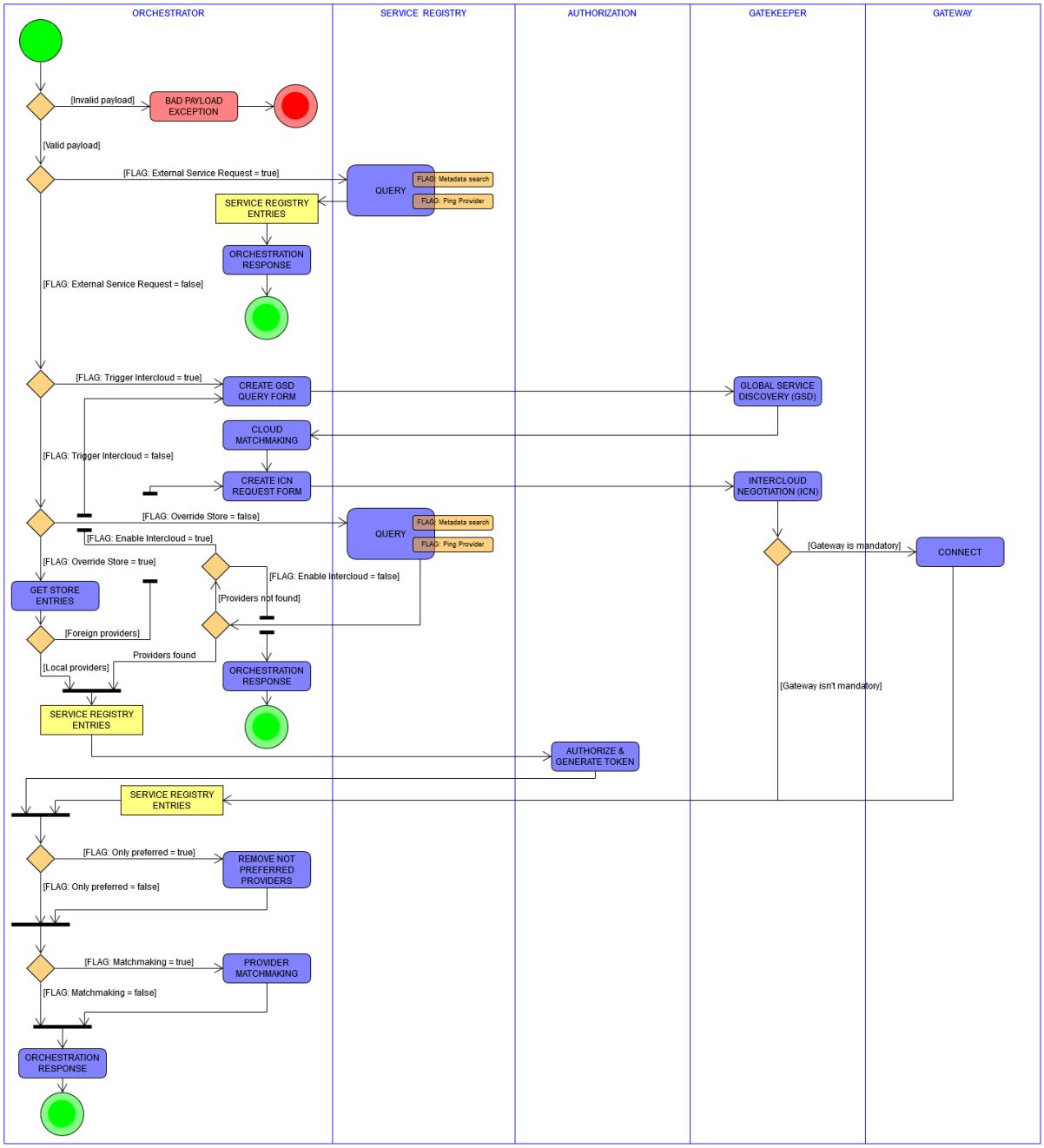

Figure 1: UML activity diagram of the orchestration process.

1.2 Important Delimitations

The flexible store mode has the following delimitations:

- It does not support inter-cloud rules.
- It does not support Quality-of-Service requirements.
- It does not support provider reservation.
- It does not use the Authorization Core System to check whether a system can consume a service. Authorization should be checked before a flexible rule is inserting into the Orchestration store.

The requested service data must meet the following criteria:

- Service definition can contain maximum 63 character of letters (english alphabet), numbers and dash (-), and has to start with a letter (also cannot ends with dash).
- Interface names have to follow the `Protocol-SecurityType-MimeType` format.
- Security types could be only `NOT_SECURE`, `CERTIFICATE` or `TOKEN` .

1.3 Access policy

Available for anyone within the local cloud, but in case of application systems, the system is allowed to orchestrate only for itself.

2 Service Interface

This section describes the interfaces to the service. The **orchestration-service** service is used to looking for matching providers. The various parameters are representing the necessary system and service input information. In particular, each subsection names an 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.

The following interfaces are available.

2.1 interface HTTP/TLS/JSON (OrchestrationForm) : OrchestrationResultList

Profile type	Type	Version
Transfer protocol	HTTP	1.1
Data encryption	TLS	1.3
Encoding	JSON	RFC 8259 [1]
Compression	N/A	-

Table 1: HTTP/TLS/JSON communication details.

3 Information Model

Here, all data objects that can be part of the **orchestration-service** 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 3.13, which are used to represent things like hashes and identifiers.

3.1 struct OrchestrationForm

Field	Type	Mandatory	Description
commands	Metadata	no	Additional commands to the Orchestrator, the only available command now is `qosExclusivity` (see above).
orchestrationFlags	OrchestrationFlags	no	A map of flags that changes the behaviour of the service. See details above.
preferredProviders	List<PreferredProvider>	no	A list of providers that takes precedence in matchmaking if they are available; if `onlyPreferred` flag is set, then the result can only be a subset of this list.
qosRequirements	Metadata	no	Quality-of-Service requirement map. See details above.
requestedService	ServiceQueryForm	no (yes)	Information about the requested service; mandatory in case of dynamic or flexible store orchestration.
requesterCloud	Cloud	no	Information about the cloud from which the request comes. Only specified when the request comes from an other cloud.
requesterSystem	System	yes	Information about the system that requests the service.

3.2 struct Metadata

An Object which maps String key-value pairs.

3.3 struct OrchestrationFlags

An Object which maps String keys to Boolean values.

3.4 struct PreferredProvider

Field	Type	Mandatory	Description
providerCloud	Cloud	no	Information about the cloud of the preferred system. Need only specified when the system is in an other cloud.
providerSystem	System	yes	Information about the preferred system.

3.5 struct Cloud

Field	Type	Mandatory	Description
name	Name	yes	Name of the cloud.
operator	Name	yes	Operator of the cloud.

3.6 struct System

Field	Type	Mandatory	Description
address	Address	yes	Network address of the system.
authenticationInfo	String	no	X.509 public key of the system.
metadata	Metadata	no	Additional information about the system.
port	PortNumber	yes	Port of the system.
systemName	Name	yes	Name of the system.

3.7 struct ServiceQueryForm

Field	Type	Mandatory	Description
interfaceRequirements	List<Interface>	no	Names of the required interfaces. If specified at least one of the interfaces must match for having result(s).
maxVersionRequirement	Number	no	Required maximum version of the service. If specified version must be equals or lower or having result(s). Ignored if `versionRequirement` is specified.
metadataRequirements	Metadata	no	Service metadata requirements. If specified the whole content of the map must match for having result(s). Only applied if the `metadataSearch` flag is set to true.
minVersionRequirement	Number	no	Required minimum version of the service. If specified version must be equals or higher or having result(s). Ignored if `versionRequirement` is specified.
pingProviders	Boolean	no	Whether or not the provider should be pinged. If true only the responding providers will comply. The orchestration flag `pingProviders` overrides this value.
securityRequirements	List<SecureType>	no	Types of the required security levels. If specified at least one of the types must match for having result(s).
serviceDefinitionRequirement	Name	yes	Identifier of the service.
versionRequirement	Number	no	Required version of the service. If specified version must match for having result(s).

3.8 struct OrchestrationResultList

Field	Type	Description
response	List<OrchestrationResult>	List of orchestration results.

3.9 struct OrchestrationResult

Field	Type	Description
authorizationTokens	Metadata	Tokens to use the service instance (one for every supported interface). Only filled if the security type is `TOKEN`.
interfaces	List<ServiceInterfaceRecord>	List of interfaces the service instance supports.
metadata	Metadata	Service instance metadata.
provider	SystemRecord	Descriptor of the provider system record.
secure	SecureType	Type of security the service instance uses.
service	ServiceDefinitionRecord	Descriptor of the service definition record.
serviceUri	String	Path of the service on the provider.
version	Version	Version of the service instance.
warnings	List<OrchestratorWarning>	List of warnings about the provider and/or its service instance.

3.10 struct ServiceInterfaceRecord

Field	Type	Description
createdAt	DateTime	Interface instance record was created at this UTC timestamp.
id	Number	Identifier of the interface instance.
interfaceName	Interface	Specified name of the interface.
updatedAt	DateTime	Interface instance record was modified at this UTC timestamp.

3.11 struct SystemRecord

Field	Type	Description
address	Address	Network address of the system.
authenticationInfo	String	X.509 public key of the system.
createdAt	DateTime	System instance record was created at this UTC timestamp.
id	Number	Identifier of the system instance.
metadata	Metadata	Additional information about the system.
port	PortNumber	Port of the system.
systemName	Name	Name of the system.
updatedAt	DateTime	System instance record was modified at this UTC timestamp.

3.12 struct ServiceDefinitionRecord

Field	Type	Description
createdAt	DateTime	Service definition instance record was created at this UTC timestamp.
id	Number	Identifier of the service definition instance.
serviceDefinition	Name	Name of the service definition.
updatedAt	DateTime	Service definition instance record was modified at this UTC timestamp.

3.13 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`.
DateTime	Pinpoints a specific moment in time.
Object	Set of primitives and possible further objects.
Interface	Any suitable type chosen by the implementor of service
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.
Number	Decimal number
OrchestratorWarning	A potentially interesting information about a provider and/or its service instance.
PortNumber	A Number between 0 and 65535.
SecureType	Any suitable type chosen by the implementor of service
String	A chain of characters.
Version	Specifies a service version.

4 References

[1] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 8259, Dec. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8259.txt

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	