ARROWHEAD

Document title
Orchestrator Core System
Date
2023-02-16
Author
Rajmund Bocsi
Contact
rbocsi@aitia.ai

Document type
SysDD
Version
4.6.0
Status
RELEASE
Page
1 (11)

Orchestrator Core System

System Design Description

Abstract
This document provides system design 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. In Section 2, we describe implementation details of the system. In Section 3, we summarize the services produced by the system.

2 Implementation

2.1 Implementation language and tools

- *Programming Language:* **Java 11**
- *Programming Framework:* **Spring-Boot 2.1.5**
- *Building Tool:* **Maven 3.5+**
- *Database Management System:* **MySQL 5.7**
- *State:* **Stateful**

2.2 Functional properties implementation

2.2.1 Database structure

Implementation of data storage functionality was done as described by Figure 1.

Please note that the Űrchestrator is only responsible for handling the data that belongs to the tables inside the red rectangle. The *system_*, *service_definition* and *service_interface* tables are controlled by the Service Registry Core System, and the *cloud* table is belongs to the Gatekeeper Core System.

Please also note the the *provider_system_id* field of the *orchestrator_store* table is either store the id of a *system_* record (local provider) or the id of a *foreign_system* record (provider from an other cloud). In the latter case, the *foreign_* flag is set to true.

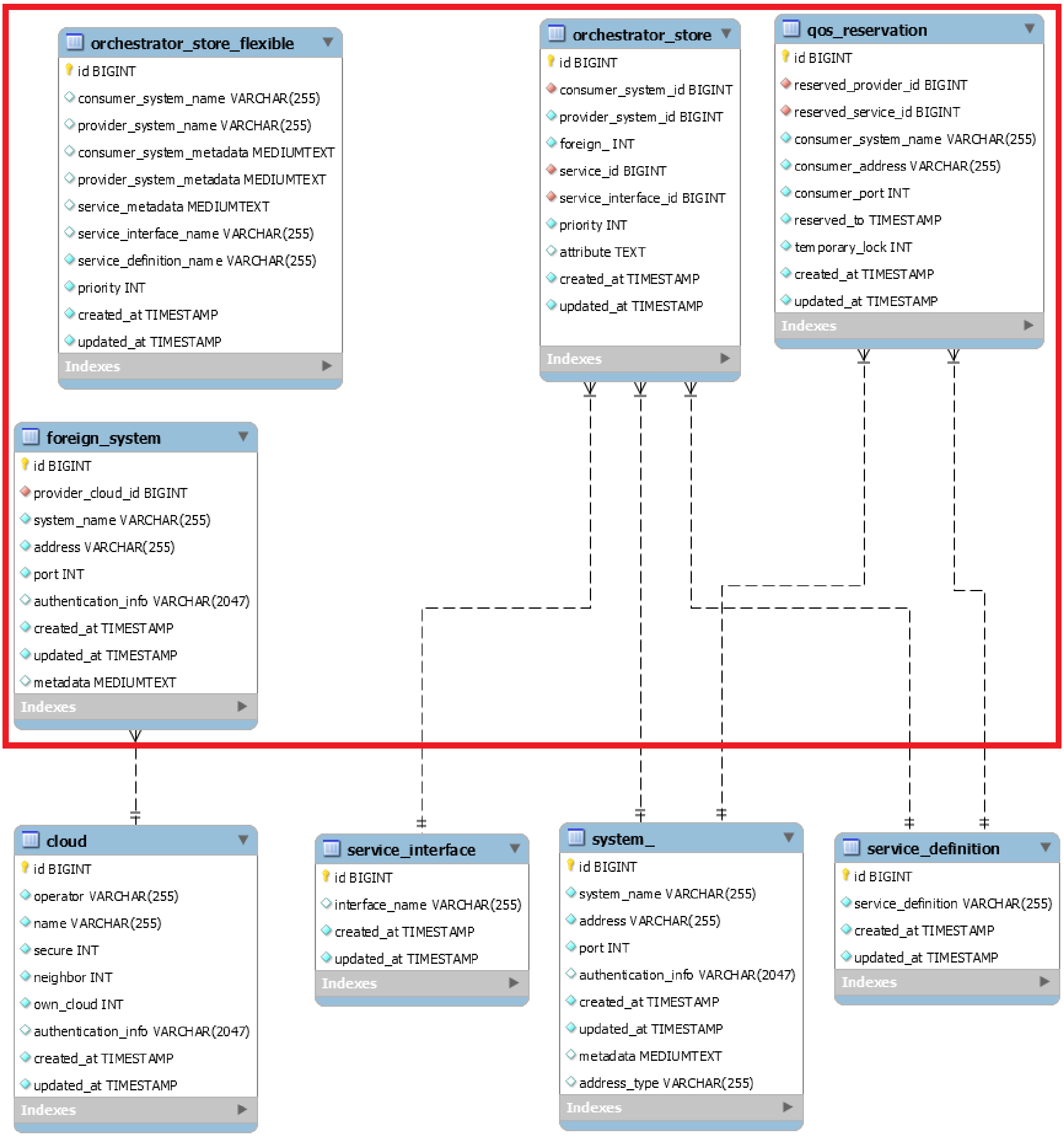

Figure 1: Database model of Orchestrator Core System.

2.2.2 Configuration

The system configuration properties can be found in the `application.properties` file which is located at `src/main/resources` folder.

Note: During the build process this file is going to be built into the executable jar, but also going to be copied next to the jar file. Any modification in the configuration file located next to the executable jar file will overide the built in configuration property value.

- **sr_address**

  The address of the Service Registry Core System in the local cloud.
- **sr_port**

  The port of the Service Registry Core System in the local cloud.
- **use_flexible_store**

  If 'true' the Orchestrator uses the newer, more flexible store orchestration. If 'false' it uses the original, more strict store orchestration.
- **use_strict_service_definition_verifier**

  If 'true', service definitions has to follow these rules: They only contains letters (english alphabet), numbers and dash (-), and have to start with a letter (also cannot end with dash).
- **gatekeeper_is_present**

  If the local cloud supports inter-cloud orchestration, a working Gatekeeper Core System must be in the LC and this flag must be 'true', so the Orchestrator can communicate with the Gatekeeper when needed.
- **enable_qos**

  Set this 'true' to enable orchestration based on Quality-of-Service measurements and provider reservations. For using this feature, a working QoS Monitor Core System is also needed.
- **qos_reservation_check_interval**

  How frequently the database should be checked for expired reservations, in seconds.
- **qos_ping_measurement_cache_threshold**

  How long a cached ping measurement can use without refreshing it using the QoS Monitor's webservice, in seconds.
- **use_strict_service_intf_name_verifier**

  Interface names has to follow this format `PROTOCOL-SECURITY-FORMAT`, where security can be `SECURE` or `INSECURE` and protocol and format must be a sequence of letters, numbers and underscore. A regexp checker will verify that. If this setting is set to true then the `PROTOCOL` and `FORMAT` must come from a predefined set.

2.3 Non functional properties implementation

2.3.1 Security

The system's security - when it is enabled - is relying on SSL 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`

The trust chain is created by issuing the cloud certificate from the master certificate and the client certificate from the cloud certificate. With other words, the cloud certificate is signed by the master certificate's private key and the client certificate is signed by the cloud certificate's private key which makes the whole chain trustworthy.

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

2.3.2 Access control

The services provided by Orchestrator Core System are applying various access policies, which are described in the related service description documents.

2.3.3 Configuration

The system configuration properties can be found in the `application.properties` file which is located at `src/main/resources` folder.

Note: During the build process this file is going to be built into the executable jar, but also going to be copied next to the jar file. Any modification in the configuration file located next to the executable jar file will overide the built in configuration property value.

- **spring.datasource.url**
  URL to the database.
- **spring.datasource.username**
  Username to the database.
- **spring.datasource.password**
  Password to the database.
- **spring.datasource.driver-class-name**
  The driver provides the connection to the database and implements the protocol for transferring the query and result between client and database.
- **spring.jpa.database-platform**
  Specify the database dialect for Java Persistence API.
- **spring.jpa.show-sql**
  Set to true in order to log out the mysql queries.

- **spring.jpa.properties.hibernate.format_sql**
  Set to true to log out mysql queries in pretty format. (Effective only when 'spring.jpa.show-sql' is 'true')
- **spring.jpa.hibernate.ddl-auto**
  Auto initialization of database tables. Value must be always 'none'.
- **server.address**
  IP address of the server.
- **server.port**
  Port number of the server.
- **domain.name**
  Set this when the system is available via domain name within the network.
- **domain.port**
  Set this when the system is available via domain port within the network.
- **coap.server.enabled**
  There is an experimental option in the Orchestrator Core System that expose some of the system's features via CoAP. Set this true to enable this option
- **coap.server.address**
  Address of the CoAP server.
- **coap.server.port**
  Port of the CoAP server.
- **core_system_name**
  Name of the system. Must be always 'ORCHESTRATOR'.
- **log_all_request_and_response**
  Set to 'true' in order to show all request/response in debug log.
- **server.ssl.enabled**
  Set to 'false' in order to disable https mode.
- **server.ssl.key-store-type**
  Type of the key store.
- **server.ssl.key-store**
  Path to the key store.
- **server.ssl.key-store-password**
  Password to the key store..
- **server.ssl.key-alias**
  Alias name of the certificate.
- **server.ssl.key-password**
  Password to the certificate.
- **server.ssl.client-auth**
  Must be always 'need' which means that SSL client authentication is necessary when SSL is enabled.

- **server.ssl.trust-store-type**
  Type of the trust store.
- **server.ssl.trust-store**
  Path to trust store.
- **server.ssl.trust-store-password**
  Password to trust store.
- **disable.hostname.verifier**
  If true, http client does not check whether the hostname is match one of the server's SAN in its certificate.

The logging configuration properties can be found in the `log4j2.xml` file located at `src/main/resources` folder.

Note: During the build process this file is going to be built into the executable jar, but it is also possible to override it from by an external file. For that use the following command when starting the system: `java -jar arrowhead-orchestrator-x.x.x -Dlog4j.configurationFile=path-to-external-file`

- **JDBC_LEVEL**
  Set this to change the level of log messages in the database. Levels: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF.
- **CONSOLE_FILE_LEVEL**
  Set this to change the level of log messages in console and the log file. Levels: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF.
- **LOG_DIR**
  Set this to change the directory of log files.

3 Services

Table 1: Services produced.

Services produced	Scope	Published
echo	Application + Core Systems	no
orchestration-service	Application + Core Systems	yes
orchestration-service-by-proxy	Workflow Choreographer Core System	yes
orchestration-service-by-id	Application + Core Systems	yes
orchestration-create-flexible-store-rules	Plant Description Engine Core System	yes
orchestration-remove-flexible-store-rules	Plant Description Engine Core System	yes
orchestration-clean-flexible-store	Plant Description Engine Core System	yes
orchestration-qos-enabled	Gatekeeper Core System	yes
orchestration-qos-reservations	Gatekeeper Core System	yes
orchestration-qos-temporary-lock	Gatekeeper Core System	yes
orchestration-confirm-reservation	Gatekeeper Core System	yes

Table 2: Services consumed.

Services consumed	Interface
query (Service Registry)	HTTP/TLS/JSON
query-multi (Service Registry)	HTTP/TLS/JSON
query-by-system-id (Service Registry)	HTTP/TLS/JSON
query-by-system (Service Registry)	HTTP/TLS/JSON
token-generation (Authorization)	HTTP/TLS/JSON
authorization-control-intra (Authorization)	HTTP/TLS/JSON
global-service-discovery (Gatekeeper)	HTTP/TLS/JSON
inter-cloud-negotiations (Gatekeeper)	HTTP/TLS/JSON
gatekeeper-get-cloud (Gatekeeper)	HTTP/TLS/JSON
qos-monitor-intra-ping-measurement (QoS Monitor)	HTTP/TLS/JSON
qos-monitor-intra-ping-median-measurement (QoS Monitor)	HTTP/TLS/JSON
qos-monitor-inter-direct-ping-measurement (QoS Monitor)	HTTP/TLS/JSON
qos-monitor-inter-relay-echo-measurement (QoS Monitor)	HTTP/TLS/JSON

4 References

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	