Service Registry Core System

System Design Description

Abstract
This document provides system design description for the **Service Registry Core System**.

Contents

1 Overview

This document describes the Service Registry Core System, which exists to enable service discovery in a Eclipse Arrowhead Local Cloud (LC). 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.

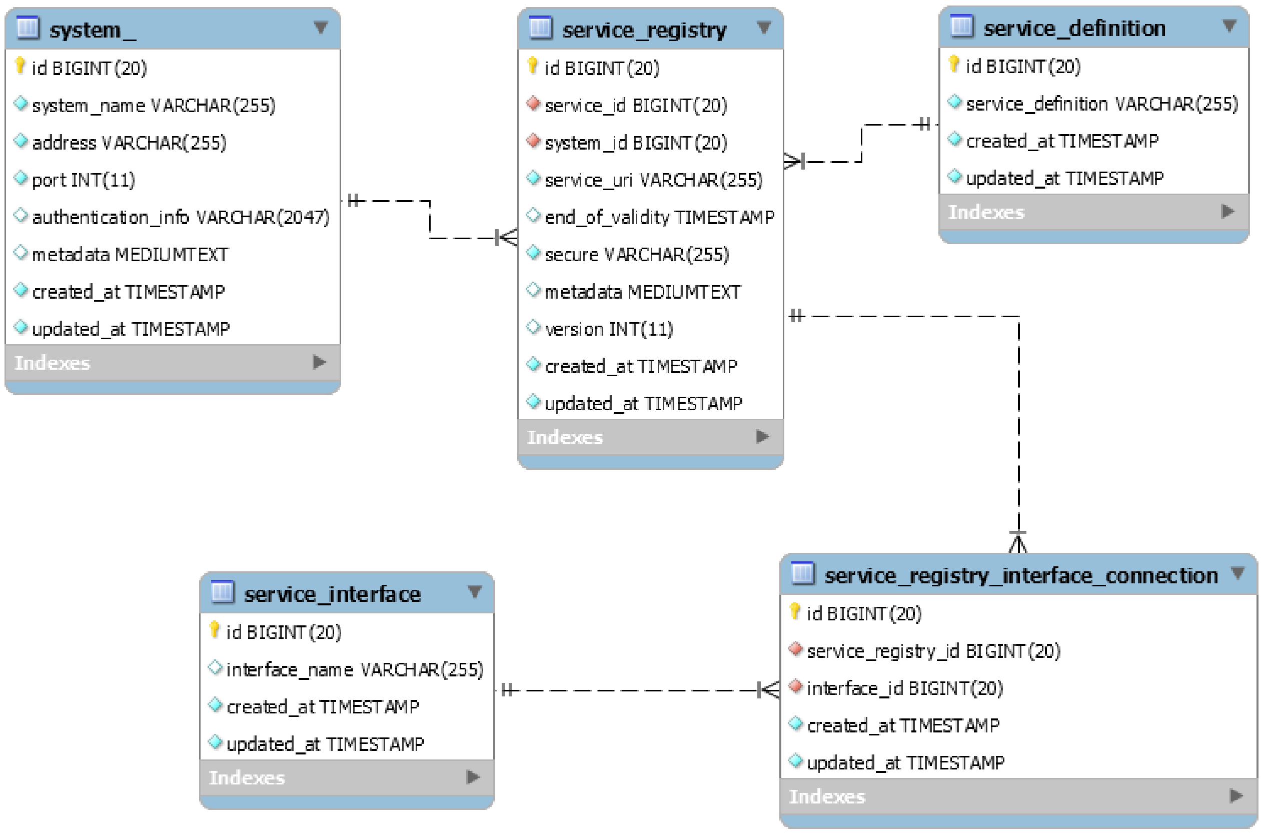

Figure 1: Database model of Service Registry Core System.

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.

- **ping_scheduled**
  If 'true' the service providers will be pinged in a fixed time interval and service offerings will be removed where the provider was not available.
- **ping_timeout**
  How much time the system should wait for the ping response (in milliseconds).
- **ping_interval**
  How frequently should the ping happen, in minutes.
- **ttl_scheduled**
  If 'true' the service offerings will be automatically removed, where the endOfValidity timestamp field is in the past, meaning the offering expired.
- **ttl_interval**
  How frequently the database should be checked for expired services, in minutes.
- **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.
- **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).
- **use_network_address_detector**
  If 'true', address detection from HttpServletRequest will be performed at the time of systems and service registration/unregistration.
- **filter_proxy_addresses**
  Comma separated list representing the possible proxy servers like load-balancer etc... before ServiceRegistry.
- **allow_self_addressing**
  If 'true', the registration of systems with self-addressing IPv4, IPv6 and no-type addresses are allowed. In case of self-addressing addresses the IP packets cannot be directed from one device to another.
- **allow_non_routable_addressing**
  If 'true', the registration of systems with non-routable IPv4 and IPv6 addresses are allowed. In case of non-routable addresses the IP packets cannot be directed from one network to another.

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 Service Registry 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.
- **core_system_name**
  Name of the system. Must be always 'SERVICEREGISTRY'.
- **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-serviceregistry-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 consol 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
service-register	Application + Core Systems	yes
service-unregister	Application + Core Systems	yes
register-system	Application + Core Systems	yes
unregister-system	Application + Core Systems	yes
query	Application + Core Systems	no
query-multi	Core Systems	no
query-all	Core Systems	no
query-by-system	Core Systems	no
query-by-system-id	Core Systems	no
pull-system	Core Systems	yes

Table 2: Services consumed.

Services consumed	Interface
-	-

4 References

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	YYYY-MM-DD	4.4.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	4.4.0	