Document title
Authorization Core System
Date
2023-02-27
Author
Tamás Bordi
Contact
tbordi@aitia.ai

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

Authorization Core System

System Design Description

Abstract

This document provides system design description for the **Authorization Core System**.

Contents

1 Overview

This document describes the Authorization Core System, which exists to enable authorization control and session control in an 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.

Please note that the Authorization 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.

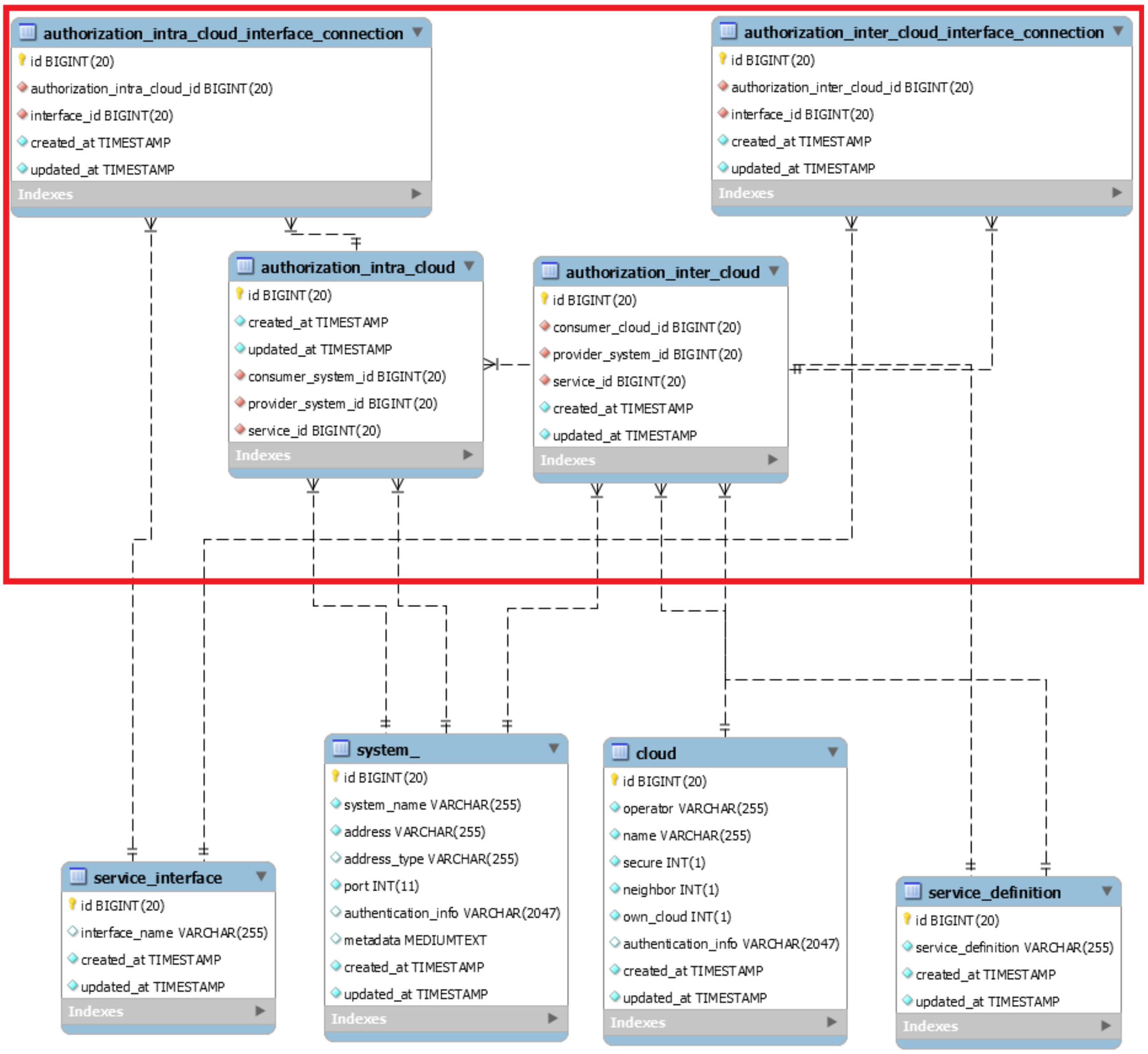

Figure 1: Database model of Authorization Core System.

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 Sercive Registry Core System in the Local Cloud.

- **sr_port**

  The port of Sercive Registry Core System in the Local Cloud.

- **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.

- **eventhandler_is_present**

  Set to 'true' if Event Handler Core System is deployed within the Local Cloud.

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 Authorization 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 'AUTHORIZATION'.
- **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
authorization-control-intra	Core Systems	yes
authorization-control-inter	Core Systems	yes
authorization-control-subscription	Core Systems	yes
token-generation	Core Systems	yes
token-generation-multi-service	Core Systems	yes
auth-public-key	Application + 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	4.6.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	4.6.0	