Messaging Hub technical documentation

Messaging Hub

IGT Consulting s.r.o

Bratislava 20.11.2024

Table of Contents

Introduction

Messaging Hub is a tool designed to manage integrations between systems using Universal Messaging (UM) or Kafka. It is divided into three main components:

1. Connections: Configuring connections to UM or Kafka.
2. Topics: Creating topics with schemas and publishing messages.
3. Interfaces: Creating triggers to send messages to endpoints.

Key Benefits

- Simplified management of connections and topics.
- Automated message delivery based on triggers.
- Visual web interface for easy administration.

Installation and Setup

- Import the MessagingHub package to your Integration Server.
- Ensure all required dependencies are installed (list to be provided).
- After installation, the Messaging Hub UI will be available at: <hostname>/MessagingHub.

Internal Databases

Start up services automatically creates two internal Derby Databases

Connections Database Schema

```
create table connections(

   connection_name varchar(128) not null unique,

   connection_type varchar(128) not null,

   is_resource_name varchar(128) not null,

   prometheus_url varchar(128),

   global_prefix varchar(128) not null unique

);
```

Interfaces Database Schema

```
create table interfaces(
   id_interface int not null generated always as identity unique,
   interface_name varchar(128) not null,
   interface_type varchar(128) not null,
   environment varchar(128) not null,
   enabled boolean not null,
   source_topic varchar(128) not null,
   message_filter varchar(128),
   delivery_method varchar(128) not null,
   custom_service_name varchar(128),
   delivery_endpoint varchar(128),
   delivery_format varchar(128),
   exclude_fields varchar(128),
   auth_type varchar(128),
   auth_user_name varchar(128),
   auth_password varchar(128),
   auth_token_service varchar(128),
   package_name varchar(128),
   um_connection varchar(128),
   global_prefix varchar(128),
   messaging_hub_forwarding varchar(128),
   trigger_execution_user varchar(128)
);
```

API Endpoints

Messaging Hub exposes these APIs

Connections

- GET /connections: Retrieves all connections.
  Responses: 200, 400, 401, 403, 404, 500.
- POST /connections: Creates a new connection.
  Responses: 201, 400, 401, 403, 404, 500.
- GET /connections/{connectionName}: Retrieves a connection by name.
  Responses: 200, 400, 401, 403, 404, 500.
- PUT /connections/{connectionName}: Updates a connection.
  Responses: 202, 400, 401, 403, 404, 500.
- DELETE /connections/{connectionName}: Deletes a connection.
  Responses: 204, 400, 401, 403, 404, 500.

Topics

- GET /{connectionName}/topics: Retrieves all topics for a connection.
  Responses: 200, 400, 401, 403, 404, 500.
- POST /{connectionName}/topics: Creates a topic.
  Responses: 201, 400, 401, 403, 404, 500.
- GET /{connectionName}/topics/{topicName}: Retrieves topic details.
  Responses: 200, 400, 401, 403, 404, 500.
- PUT /{connectionName}/topics/{topicName}: Updates a topic.
  Responses: 202, 400, 401, 403, 404, 500.
- POST /{connectionName}/topics/{topicName}: Publishes a message to a topic.
  Responses: 200, 400, 401, 403, 404, 500.
- DELETE /{connectionName}/topics/{topicName}: Deletes a topic.
  Responses: 204, 400, 401, 403, 404, 500.

Interfaces

- GET /interfaces: Retrieves all interfaces.
  Responses: 200, 400, 401, 403, 404, 500.
- POST /interfaces: Creates a new interface.
  Responses: 201, 400, 401, 403, 404, 500.
- GET /interfaces/{env}/{interfaceName}: Retrieves interface details.
  Responses: 200, 400, 401, 403, 404, 500.
- PUT /interfaces/{env}/{interfaceName}: Updates an interface.
  Responses: 202, 400, 401, 403, 404, 500.
- DELETE /interfaces/{env}/{interfaceName}: Deletes an interface.
  Responses: 204, 400, 401, 403, 404, 500.

Frontend

The frontend of the application is built using:

- React: A JavaScript library for building user interfaces.
- Material-UI (MUI): A component library for modern, responsive designs.

Key Features

Component-Oriented Architecture: Each section (Connections, Topics, Interfaces) is implemented as a React component.

Dynamic Data Loading: Utilizes API calls to fetch connections, topics, and interfaces.

User Experience: Intuitive controls using MUI components such as tables, forms, and modals.

Responsiveness: Optimized interface for all devices.

Technical Details

Frontend code location: MessagingHub/code/pub.

Core libraries:

- React Router: For navigation between sections.
- Axios: For handling API calls.
- Formik + Yup: For form validation.
- MUI: For UI components such as cards, buttons, and tables.