Solace PubSub+ Connector for MQTT

Solace Corporation

Version 3.15.5

solace.

Table of Contents

Preface

Solace PubSub+ Connector for MQTT bridges data between the Solace PubSub+ Event Broker and MQTT providing a flexible and efficient way to integrate MQTT data with your Solace-backed, event-driven architecture and the Event Mesh. The connector is deployable standalone or in redundancy modes of “active-standby” or “active-active” to allow for high-availability and horizontal scaling of your data movement. Each connector instance supports up to 20 individual workflows (source-to-target pipeline), minimizing the number of connector instances deployed and managed. The use of various Spring Framework technologies allows for easy configuration of the connector, advanced logging capabilities, and export of live metrics data to external monitoring solutions.

Getting Started

Release Notes

Prerequisites

- Docker or Podman
- PubSub+ Event Broker
- MQTT

Usage

Connecting to Services on the Host

If services (for example a PubSub+ event broker) are exposed on the localhost, they can be referenced using the container platform's special DNS name with `SOLACE_JAVA_HOST`, which resolves to an internal IP address that's used by the host.

For example in Docker, use the following command:

```
docker run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  --env SOLACE_JAVA_HOST=host.docker.internal:55555 \
  solace/solace-pubsub-connector-mqtt:3.15.5
```

For example in Docker, use the following command:

```
podman run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/config/:/app/external/spring/config/:ro \
  --env SOLACE_JAVA_HOST=host.containers.internal:55555 \
  solace/solace-pubsub-connector-mqtt:3.15.5
```

Configuring a Healthcheck

You can configure the health to perform the following tasks:

- Create a regular read-only user called `healthcheck` with a password using `SOLACE_CONNECTOR_SECURITY_USERS_0_NAME` and `SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD`.
- Use the `healthcheck` user as the user to poll the management health endpoint in the container's `healthcheck` command and fails it if the connector is unhealthy.

Here's a basic example command of how to configure the health check for container:

For example in Docker, use the following command:

```
docker run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/application.yml:/app/external/spring/config/application.yml:ro \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_NAME=healthcheck \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD=healthcheck \
  --healthcheck-command="curl -X GET -u healthcheck:healthcheck --fail
localhost:8090/actuator/health" \
  solace/solace-pubsub-connector-mqtt:3.15.5
```

For example in Podman, use the following command:

```
podman run -d --name my-connector \
  -v `pwd`/libs/:/app/external/libs/:ro \
  -v `pwd`/application.yml:/app/external/spring/config/application.yml:ro \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_NAME=healthcheck \
  --env SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD=healthcheck \
  --healthcheck-command="curl -X GET -u healthcheck:healthcheck --fail
localhost:8090/actuator/health" \
  solace/solace-pubsub-connector-mqtt:3.15.5
```

Providing Configuration

You can provide Spring configuration properties to this container using one of the following ways:

1. Use environment variables.
2. Use volumes containing Spring configuration files (as well as other volumes).

Ports

The following ports are required for the container:

Port	Usage
`8090`	The connector's management endpoint.

Volumes

These are the supported directories for which volumes and bind mounts can be created:

Contents	Container Path	Optional	Recommended Permission
Spring configuration files	`/app/external/spring/config/`	Required unless all properties are defined using environment variables	Read-Only
Libraries	`/app/external/libs/`	Required	Read-Only
Classpath files	`/app/external/classpath/`	Optional	Read-Only
Output files	`/app/external/output/`	Optional	Read/Write

Volume: Spring Configuration Files

The Spring configuration files volume is used to add Spring configuration files (such as `application.yml`, etc.), add a read-only volume, or bind mount to `/app/external/spring/config/`.

This directory follows the same semantics as Spring's default `config/` directory. That fact means that this connector automatically finds and loads Spring configuration files from the following locations when the connector starts:

1. The root of `/app/external/spring/config/`.
2. Immediate child directories of `/app/external/spring/config/`.

If you want configuration files for multiple, different connectors within the same `config` directory for use in different environments (such as development, production, etc.), we recommend that you use Spring Boot Profiles instead of child directories. For example:

- Set up your configuration like this:
  - `/app/external/spring/config/application-prod.yml`
  - `/app/external/spring/config/application-dev.yml`
- Do not do this:
  - `/app/external/spring/config/prod/application.yml`
  - `/app/external/spring/config/dev/application.yml`

Child directories are intended to be used for merging configuration from multiple sources of configuration properties. For more information and an example of when you might want to use multiple child directories to compose your application's configuration, see the Spring Boot documentation.

Volume: Libraries

The Libraries volume adds additional libraries, adds a read-only volume, or binds a mount to `/app/external/libs/`.

This directory is provided as the location for the Java library dependencies (external JAR files) that are required only when using certain features of the Connector (such as Prometheus libraries when using the metrics export to Prometheus feature in your Connector configuration).

See the documentation provided in the `libs` directory of the connector in the ZIP file for more information.

Volume: Classpath Files

The Classpath Files volume adds a location for arbitrary files (not JAR libraries nor Spring Boot configuration files), adds a read-only volume, or binds a mount to `/app/external/classpath/`.

This directory must not contain JAR files for libraries or Spring Boot configuration files, otherwise there is a risk of libraries not getting picked up during the deployment of the connector and overwriting the connector's internal configuration.

Volume: Output Files

The Output Files volume is for some features that support writing output files, such as logging to a file. To capture these, add a read/write volume or bind the mount to `/app/external/output/` directory.

When using features that generates files, you must configure the features so that the files are generated to the `/app/external/output/` directory. Generating files to any other directory is not supported.

Configuring the JVM

You can set the `JDK_JAVA_OPTIONS` environment variable on the container to configure the Java Virtual Machine (JVM).

See the JDK documentation for more information.

This container is provided as an example and has been tested using:

- Two active processors (specified using `-XX:ActiveProcessorCount=2`).
- A maximum heap memory of 2 GB (specified using `-Xmx2048m`).

Support

Support is offered best effort via our Solace Developer Community.

Premium support options are available, please Contact Solace.

License

This project is licensed under the Solace Community License, Version 1.0. - See the `LICENSE` file under the container's `/licenses` for details.