AIT Lablink Redis Client

AIT Lablink Development Team

Jul 04, 2022

INSTALLATION

Lablink client for interfacing to a Redis database. Its main purpose is to interface to AIT's roadb (Redis-Opal Asynchronous Data Buffer) in order to access measurement data from a real-time simulation running on an OPAL-RT system. Similarly, the client allows sending commands and set points to a running simulation. The client can also be used in a basic mode, to map a Lablink datapoint to a Redis key.

See the examples section for more infos.

Note: in order to work with OPAL-RT, the roadb service should be running on the OPAL target computer.

CHAPTER

ONE

INSTALLATION

Find information about the installation of the Lablink Redis client *here*.

1.1 Maven project dependency

The Lablink Redis client's compiled Java package is available on the Maven Central Repository. Use it in your local Maven setup by including the following dependency into your *pom.xml*:

```
<dependency>
  <groupId>at.ac.ait.lablink.clients</groupId>
  <artifactId>redisclient</artifactId>
  <version>0.0.2</version>
</dependency>
```

Note: You may have to adapt this snippet to use the latest version, please check the Maven Central Repository.

1.2 Installation from source

Installation from source requires a local **Java Development Kit** installation, for instance the Oracle Java SE Development Kit 13 or the OpenJDK.

Check out the project and compile it with Maven:

```
git clone https://github.com/ait-lablink/lablink-redis-client
cd lablink-redis-client
mvnw clean package
```

This will create JAR file *redisclient-<VERSION>-jar-with-dependencies.jar* in subdirectory *target/assembly*.

CHAPTER

TWO

RUNNING THE CLIENTS

Find basic instructions for running the client *here*.

2.1 Invoking the clients from the command line

When running the clients, the use of the `-c` command line flag followed by a **local configuration file** or the **URI to the configuration** is mandatory (see *here* for more on configuring the Lablink Redis client).

For example, on Windows this could look something like this:

```
SET LLCONFIG=http://localhost:10101/get?id=
SET CONFIG_FILE_URI=%LLCONFIG%ait.test.redis.config

SET REDIS_OPAL=at.ac.ait.lablink.clients.redisclient.RedisOpalClient
SET REDIS_JAR_FILE=\path\to\lablink-redis-client\target\assembly\redisclient-<VERSION>-
↪jar-with-dependencies.jar

java.exe -cp "%REDIS_JAR_FILE%" %REDIS_OPAL% -c %CONFIG_FILE_URI%
```

CHAPTER

THREE

CONFIGURATION

Find the reference for writing a configuration for a Lablink Redis client *here*.

3.1 Overview

The configuration has to be JSON-formatted.

3.2 Basic Lablink Client Configuration

Required parameters:

clientName
client name

groupName
group name

scenarioName
scenario name

labLinkPropertiesUrl
URI to Lablink configuration

syncHostPropertiesUrl
URI to sync host configuration

3.3 Redis Database Connection Configuration

Required parameters:

redisIP
IP address of Redis database

redisPort
connection port of Redis database

Optional parameters:

msTimeInterval
: time interval in milliseconds for retrieving data values (measurements) from the Redis database (default: `5000`)

3.4 Commands / Measurements Configuration

For interacting with the Redis database, the client defines **measurements** (i.e., data values to be read from the database) and **commands** (i.e., data values to be written to the database). Both can be either defined in a local configuration file (with Redis keys in separate lines) or as JSON arrays.

Configuration via local file:

cmdsFile
: path to local configuration file defining commands via Redis keys in separate lines

measFile
: path to local configuration file defining measurements via Redis keys in separate lines

Configuration via JSON array:

commands
: JSON array defining commands via Redis keys in separate entries

measurements
: JSON array defining measurements via Redis keys in separate entries

Note: Redis keys may not contain slashes (/)!

3.5 Example Configuration

The following is an example configuration using local configuration files to define commands / measurements:

```
{
  "clientName" : "RedisOpalClient",
  "groupName" : "RedisOpalDemo",
  "scenarioName" : "RedisOpalScenario",
  "syncHostPropertiesUrl" : "http://localhost:10101/get?id=ait.example.all.sync-host.
↪properties",
  "labLinkPropertiesUrl" : "http://localhost:10101/get?id=ait.example.all.llproperties",
  "redisIP" : "192.168.100.200",
  "redisPort" : "6379",
  "cmdsFile" : "commands.sgnl",
```

(continues on next page)

(continued from previous page)

```
  "measFile" : "measurements.sgnl"
}
```

The following is an example configuration using JSON arrays to define commands / measurements:

```
{
  "clientName" : "RedisOpalClient",
  "groupName" : "RedisOpalDemo",
  "scenarioName" : "RedisOpalScenario",
  "syncHostPropertiesUrl" : "http://localhost:10101/get?id=ait.example.all.sync-host.
↪properties",
  "labLinkPropertiesUrl" : "http://localhost:10101/get?id=ait.example.all.llproperties",
  "redisIP" : "192.168.100.200",
  "redisPort" : "6379",
  "commands": [
    "cmd_Test.line_1_Line.Data.Points.P1",
    "cmd_Test.line_1_Line.Data.Points.Q1"
  ],
  "measurements": [
    "meas_Test.ext_el_grid_Generator.Data.Points.Vmag",
    "meas_Test.ext_el_grid_Generator.Data.Points.Vang"
  ]
}
```