AIT Lablink Plotter

AIT Lablink Development Team

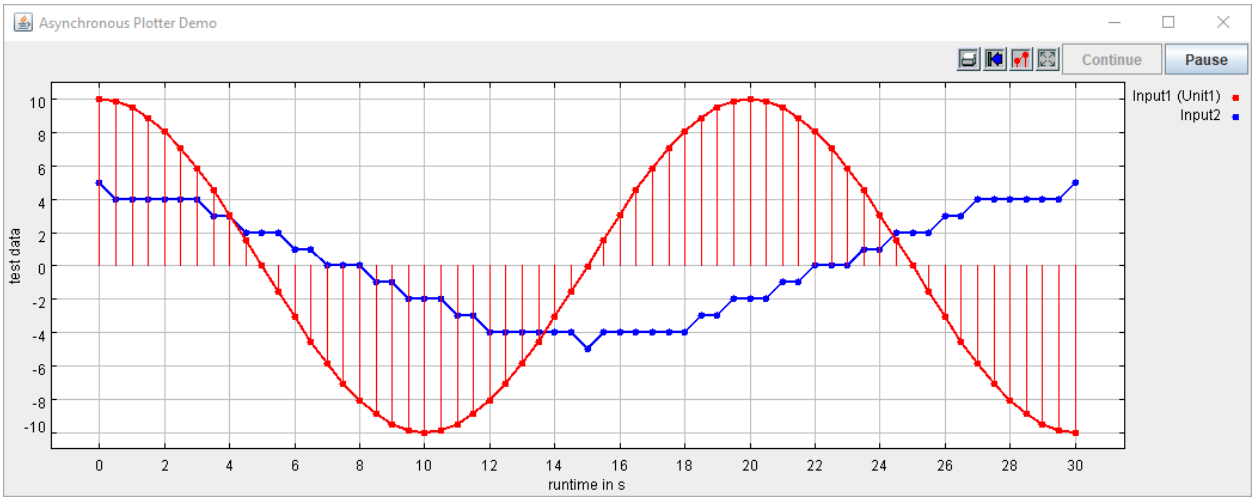
Feb 02, 2022

INSTALLATION

This package provides Lablink clients that visualize data as plotted graphs in a separate window. These plotted graphs are continuously updated while the client is running, showing the input data as a function of time.

The Lablink clients provided by this package are:

- *PlotterAsync*: this client runs asynchronously and plots new data points as a function of wall-clock time
- *PlotterSync*: this client runs synchronoulsy (i.e., synchronized by a sync host) and plots new data points as a function of synchronization time



CHAPTER

ONE

INSTALLATION

Find information about the installation of the Lablink plotter clients *here*.

1.1 Maven

The Lablink plotter'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>plotter</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 Building 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-plotter.git
cd lablink-plotter
mvnw clean package
```

This should create JAR file *plotter-<VERSION>-jar-with-dependencies.jar* in subdirectory *target/assembly*. Also, all additional Lablink resources needed for running the *examples* will be copied to directory *target/dependency*.

CHAPTER

TWO

RUNNING THE CLIENTS

Find basic instructions for running the clients *here*.

2.1 Invoking the clients from the command line

When running the clients, the use of the `-c` command line flag followed by the URI to the configuration (see *here*) is mandatory.

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

```
SET PLOT=at.ac.ait.lablink.clients.plotter.PlotterSync
SET LLCONFIG=http://localhost:10101/get?id=
SET CONFIG_FILE_URI=%LLCONFIG%ait.test.plotter.sync.config
SET MEMORY_FLAG=-Xmx1024M
java.exe %MEMORY_FLAG% -cp \path\to\plotter-<VERSION>-jar-with-dependencies.jar %PLOT% -
↪c %CONFIG_FILE_URI%
```

CHAPTER

THREE

CONFIGURATION

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

3.1 Overview

The configuration has to be JSON-formatted. It is divided into the following three categories:

Client basic configuration of the Lablink client (JSON object)

Input configuration of the client's inputs, each visualized as individual dataset (JSON array of JSON objects)

Plot configuration of the plot (JSON object)

In the following, the configuration parameters for these categories are listed.

See also:

See *below* for an example of a complete JSON configuration.

3.2 Client configuration

General client configuration

ClientName client name

GroupName group name

ScenarioName scenario name

labLinkPropertiesUrl URI to Lablink configuration

syncHostPropertiesUrl URI to sync host configuration

ClientDescription description of the client (optional)

ClientShell activate Lablink shell (optional, default: `false`).

Data plotted to the screen can also be stored as timeseries in CSV files. This can be configured individually for each input of the plotter (see parameter *WriteToFile* of the *input configuration*). The following parameters determine the general configuration of the CSV output, which is applied to all data written to CSV files.

CSV output configuration

WriteDataDirURI Specify in which directory to write the CSV output files as URI. If the specifed directory does not exist, the plotter will try to create it. If not specified, CSV files are written to the current working directory. Paths can be specified as `plotter://relative/path/to/dir`, where the path will be interpreted relative to the path specified via system property `dataDir`. For instance, using option `-DdataDir=/a/b/c` when starting the plotter client and specifiying `plotter://x/y/z` will result in CSV files being written to directory `` `/a/b/c/x/y/z ``. (optional)

WriteDataTimestamp If `true`, the timeseries written to the CSV files use timestamps. Otherwise, the elapsed time since starting the plotter is used. (optional, default: `false`)

3.3 Input configuration

Configuration for each input

InputID name of the input, used in plot legend

DataType data type of the input, allowed values are `double` and `long`

Unit unit associated to the input, used in plot legend (optional)

LineStyle string specifying the color for points, allowed values are `solid`, `dotted`, `dashed`, `dotdashed` and `dotdotdashed` (optional, default: `solid`)

MarksStyle set the marks style, allowed values are `none`, `points` and `dots` (optional, default: `dots`)

Connected if `true`, subsequent points in the plot are connected with a line (optional, default: `true`)

Impulses if `true`, then a line will be drawn from any plotted point down to the x axis (optional, default: `false`)

WriteToFile if `true`, then new values will not only be plotted to the screen but also written to a CSV output file called "<*InputID*>.*csv*" (optional, default: `false`)

3.4 Plot configuration

Note: Either **AutomaticRescale** has to be set to `true` or **XMin**, **XMax**, **YMin** and **YMax** have to be specified!

AutomaticRescale if `true`, axes are rescaled automatically at runtime to fit all data on the plot canvas

XMin left bound of x-axis

XMax right bound of x-axis

YMin lower bound of y-axis

YMax upper bound of y-axis

Other parameters:

Title title of the plot (optional, default: `Plotter`)

XLabel x-axis label (optional, default: `time`)

YLabel y-axis label (optional, default: `value`)

DisplayGrid control whether the grid is drawn (optional, default: `true`)

PersistencePoints a positive argument sets the persistence of the plot to the given number of points, calling with a zero argument turns off this feature, reverting to infinite memory (optional, default: `0`)

PersistenceX a positive argument sets the persistence of the plot to the given width in units of the horizontal axis, calling with a zero argument turns off this feature, reverting to infinite memory (optional, default: `0.0`)

3.5 Example configuration

```
{
  "Client": {
    "ClientDescription": "A simple plotter.",
    "ClientName": "TestPlotterSync",
    "ClientShell": false,
    "GroupName": "PlotterDemo",
    "ScenarioName": "PlotterSync",
    "WriteDataDirURI": "file:///C:/Development/lablink/plotter",
    "labLinkPropertiesUrl": "http://localhost:10101/get?id=ait.all.all.llproperties",
    "syncHostPropertiesUrl": "http://localhost:10101/get?id=ait.test.plotter.sync.sync-
↪host.properties"
  },
  "Input": [
    {
      "Connected": true,
      "DataType": "Double",
      "Impulses": true,
      "InputID": "Input1",
      "LineStyle": "dashed",
      "MarksStyle": "dots",
      "Unit": "Unit1"
    },
    {
      "DataType": "long",
      "InputID": "Input2",
      "WriteToFile": true
    }
  ],
  "Plot": {
    "AutomaticRescale": false,
    "DisplayGrid": true,
    "PersistencePoints": 0,
    "PersistenceX": 0,
```

(continues on next page)

(continued from previous page)

```
    "Title": "Asynchronous Plotter Demo",
    "XLabel": "runtime in s",
    "XMax": 60,
    "XMin": 0,
    "YLabel": "test data",
    "YMax": 10,
    "YMin": -10
  }
}
```

CHAPTER

FOUR

EXAMPLES

Find step-by-step instructions for running the examples *here*.

4.1 Prerequisites

4.1.1 Required Lablink resources

The following Lablink resources are required:

- Configuration Server: *config-0.0.1-jar-with-dependencies.jar*
- Datapoint Bridge: *dpbridge-0.0.1-jar-with-dependencies.jar*
- Simple Sync Host: *sync-0.0.1-jar-with-dependencies.jar*

When *building from source*, the corresponding JAR files will be copied to directory *target/dependency*.

4.1.2 Starting the configuration server

Start the configuration server by executing script run_config.cmd in subdirectory examples/0_config. This will make the content of database file *test-config.db* available via http://localhost:10101.

Note: Once the server is running, you can view the available configurations in a web browser via http://localhost:10101.

See also:

A convenient tool for viewing the content of the database file (and editing it for experimenting with the examples) is DB Browser for SQLite.

4.1.3 MQTT broker

An **MQTT broker** is required for running the example, for instance Eclipse Mosquitto or EMQ.

4.2 Example 1: Asynchronous plotter

All relevant scripts can be found in subdirectory examples/1_async. To run the example, execute all scripts either in separate command prompt windows or by double-clicking:

- dpb.cmd: runs the data point bridge service, connecting the data source and the plotter
- source.cmd: runs the data source, which will send data to the plotter
- plot.cmd: runs the plotter, which will plot incoming data to the screen (and write one of the inputs to a CSV output file)

Note: The order in which the scripts are started is arbitrary.

4.3 Example 2: Synchronous plotter

All relevant scripts can be found in subdirectory examples/2_sync. To run the example, execute all scripts either in separate command prompt windows or by double-clicking:

- dpb.cmd: runs the data point bridge service, connecting the data source and the plotter
- source.cmd: runs the data source, which will send data to the plotter
- plot.cmd: runs the plotter, which will plot incoming data to the screen (and write one of the inputs to a CSV output file)
- sync.cmd: runs the sync host

Note: Start the data point bridge and the clients first (in arbitrary order). **Before you start the sync host**, make sure that the **clients are already connected to the data point bridge** (check status messages of data point bridge).