

# General overview of L0 API

## Use cases

Layer0 API is a relatively simple way to:

- 1) Replay your own file format with Bookmap.
- 2) Connect Bookmap to your own datasource in real time (both for receiving data and trading).

Doing that requires basic Java knowledge, however (1) is possible even without any Java knowledge at all using any programming language if you don't mind converting files before playing instead of integrating format support into Bookmap platform.

## Code examples

**Layer0ApiDemo** project provides 4 demo classes with detailed comments that illustrate most typical scenarios:

- 1) **DemoTextDataReplayProvider** - load simple text format into Bookmap. As an option you can just generate a file in this format and feed it to bookmap using this module. This enables using any language for conversion to Bookmap format
- 2) **DemoGeneratorReplayProvider** - mimics a replay provider, but instead of loading data this provider generates it. Example of setting order queue position and displaying simple indicators using legacy API. Intended for simple transition from Recorder API, at some point in the future will be replaced by Layer2-based solution.
- 3) **DemoExternalRealtimeProvider** - example of custom realtime data provider. This one only generates random data for illustrational purposes, but you can use it to build your own connectivity.
- 4) **DemoExternalRealtimeTradingProvider** - extends **DemoExternalRealtimeProvider** providing trading capability. Simulates limit orders over the generated data. Shows how to build your own provider for connecting to platform with trading support.

Project also contains some more complicated examples.

## Javadoc

Javadoc is available in

- Maven repository, you can see exact artifact version in build.gradle file located in the root of the repository.
- bm-l1api-javadoc.jar typically located in "C:\Program Files\Bookmap\lib" (location might differ depending on path selected during installation). It contains documentation for all

levels of API, but for L0 API it's mostly sufficient to only look inside `velox.api.layer0` package.

It's highly advised to read javadoc for **ExternalLiveBaseProvider** and **ExternalReaderBaseProvider** - extending those classes will provide some parts of the logic already implemented for you.

## Loading modules

There are 2 ways to load modules.

### Loading via configuration files

This approach is recommended for development, as it allows loading class files directly, without building a jar file.

In order for Bookmap to load your module it should be added to a configuration file inside `C:\Program Files\Bookmap\lib\UserModules\L0`

There are 2 files, **external-live-modules.txt** and **external-reader-modules.txt**

First one contains list of live connectivity modules, and second one contains a list of replay modules.

Files can contain comment lines (start with #) and lines in following format:

`<full-class-name> <path-without-spaces>`

Path can point to either folder or .jar file with classes, whatever is more convenient.

For example, line in **external-live-modules.txt** can look like this:

`velox.api.layer0.live.DemoExternalRealtimeTradingProvider D:\Layer0ApiDemo\bin`

### Loading from a jar file

As long as module is annotated with `@Layer0LiveModule` or `@Layer0ReplayModule` it can be packed into JAR file and placed in `C:\Bookmap\API\Layer0ApiModules` (or similar location if path was changed during installation process). Bookmap will load it automatically on startup, as long as it's permitted by license.

## Using modules

After modules are configured those are used similarly to Bookmap internal functionality

### Replay

Select a file the same way you would open Bookmap file. Note, that since extension of your file will likely be different from .bmf, you should first type "\*" into the file name field and press enter - this will reset file type filter and will allow you to load any type of file.

## Live

Configure a connection in “Connections” menu selecting “External” as platform. Fill username, password fields, connect the same way you would do with any other connection.

If you don’t need username or password just leave those empty - actual use of that data will be defined by your code.

If you need to place some additional data like server address, you can place it into one of those fields. E.g. user “u1” at server “127.0.0.1” could be set as u1@127.0.0.1 inside “username” field.

## Environment setup

### Example project

Repository contains a gradle project, which you can import into IDE of your choice.

It was tested with Eclipse and it will be used for the rest of this guide (though, technically, you don’t even need an IDE - project can be packed by running “gradle jar” in the root of the repository).

Make sure you have Bookmap installed (using default path will make things a bit more simple), then import example **Layer0ApiDemo** project as gradle project (via Builldship plugin; depending on your Eclipse version you might have to install it first).

Now you can edit the code and compile it. You can either include “bin” folder using config file or pack it to jar and load it into bookmap.

### Creating your own project

For your own project you can copy the build.gradle file to a new folder and create src\main\java and src\main\resources folders near it.

If you don’t want to use gradle, you can just use API (either from "C:\Program Files\Bookmap\lib\bm-l1api.jar" or downloaded from the repository) as a compile-time dependency and build classes or jar file in any other way.

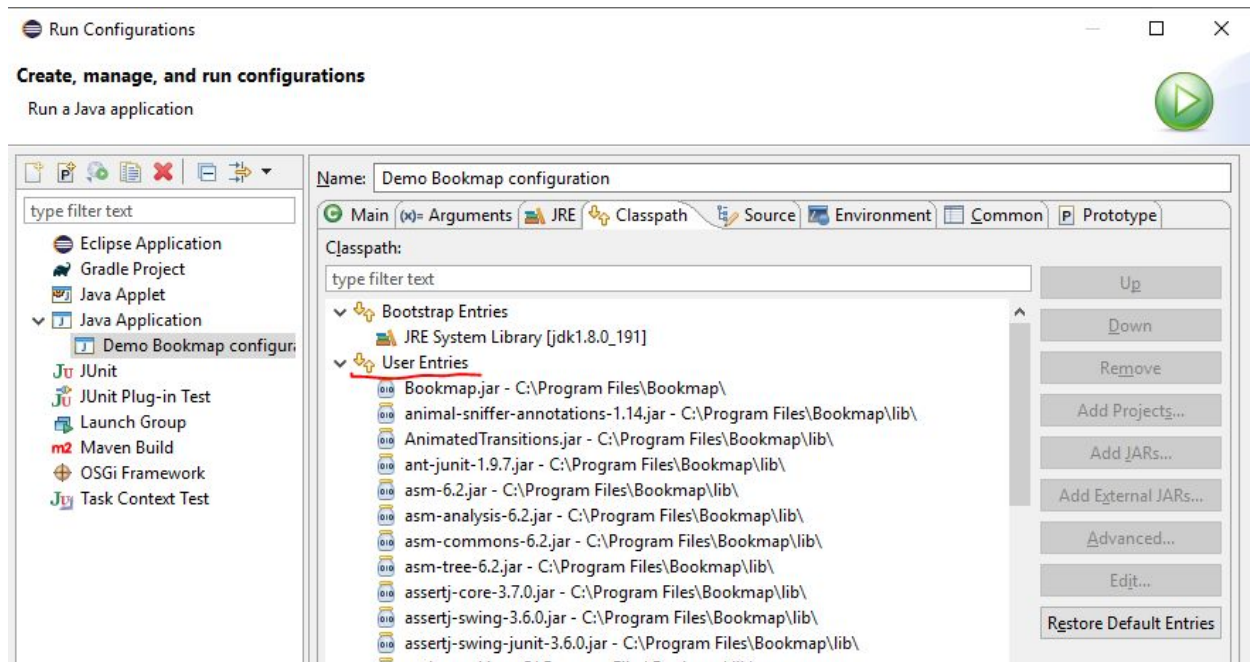
## Running from IDE

While you can run Bookmap using the shortcut, it will be easier to run it from IDE and this will also provide additional debugging capabilities.

- 1) Configure your IDE to run bookmap. For this configure ide to
  - use Java 8 x64 to run Bookmap (while it will run with 32 bit version, it's not recommended).
  - add C:\Program Files\Bookmap\Bookmap.jar to classpath
  - add all jar files in C:\Program Files\Bookmap\lib to classpath
  - start velox.ib.Main

- Use C:\Bookmap\Config as working directory
- 2) Adjust configuration files in C:\Program Files\Bookmap\lib\UserModules\L0 to load the class you are developing from the folder where it's compiled to (in eclipse it's "bin" folder)
- 3) Now you should be able to use this configuration to start bookmap with your strategy being loaded after it starts.

Note that jar files should be added as User entries (in Eclipse run configuration):



When you start Bookmap this way you will be able to debug your code from IDE, including capability to edit code in debug mode without restarting the application.