The Aerie command line interface for planning allows manipulation of plans in batch using the commands listed below. In order to use the command line interface users have run Aerie on their localhost, where the end points are hardcoded in an Aerie config file. In the future, the CLI can be configured to point to any host machine to make requests against. At that point, aerie services can run anywhere, and the CLI will work from any other machine as long as it has the proper host configured. Example input/output JSON is given in the appendix for reference.

Installation

The CLI is contained in a JAR file that can be downloaded from Artifactory. To install it, download the tar file located here. Note that you may need to log in to Artifactory to reach the download link. Decompress the file to your desired location. This can be done manually by going to your desired location in a terminal and entering

```
tar -xzf ~/Downloads/aerie-develop.tar.gz
```

The JAR will now be located at `<installation-path>/merlin-cli/merlin-cli.jar`. The CLI can now be run by executing

```
java -jar <installation-path>/merlin-cli/merlin-cli.jar
```

but this is cumbersome, so we suggest setting up an alias to run the cli. For shells such as Bash and Zsh (the default shells for recent Macintosh computers), this can be done as such:

```
alias merlin-cli="java -jar <installation-path>/merlin-cli/merlin-cli.jar"
```

At this point the CLI can be run by entering `merlin-cli` at the command prompt, however if a new terminal is opened the alias will need to be set up again. To avoid this, add the alias to your rc file, so it will be initialized every time your terminal starts up.

Below is an example of how to install the merlin CLI to `~/opt` after downloading the .tar.gz file in zsh:

```
cd ~/opt
tar -xzf ~/Downloads/aerie-develop.tar.gz
alias merlin-cli="java -jar ~/opt/merlin-cli/merlin-cli.jar"
```

Known Issues

It is possible when running the JAR file you will see something like the following error:

```
Error: A JNI error has occurred, please check your installation and try again
```

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: gov/nasa/jpl/ammos/mpsa/a
has been compiled by a more recent version of the Java Runtime (class file version 55.0), th
Runtime only recognizes class file versions up to 52.0
```

If this error occurs, the cause is likely due to a mismatch in the version of Java being run and the version the JAR was compiled with. The CLI uses Java 11, so Java 11 should be used to run the CLI. Your java version can be checked by entering `java -version` at the command prompt.

Usage

The CLI is currently designed to automatically connect to a local deployment of the Aerie services (see here). In the future, the CLI will be configurable to connect to remote services, but at this time only use with a local deployment is possible.

Create a single adaptation

Create an adaptation from a JAR file and metadata, returns an adaptation ID.

```
merlin-cli --create-adaptation <path-to-adaptation-jar> name=<name> version=<version> missio
```

Query all adaptations

Returns a list of all adaptations indexed by adaptation ID.

```
merlin-cli --list-adaptations
```

Query adaptation metadata

Returns metadata of an adaptation specified with ID.

```
merlin-cli -a <adaptation-id> --view-adaptation
```

Query all activity types in an adaptation

Returns a list of activity types indexed by activity type name. For each activity type, parameter names and types as well as default values are given.

```
merlin-cli -a <adaptation-id> --activity-types
```

Query an activity type in an adaptation

Returns a list of parameters as well as their default values for an activity type specified by ID.

```
merlin-cli -a <adaptation-id> --activity-type <activity-type-id>
```

Query activity type parameters

Returns a list of activity type parameters for a given activity type within an adaptation, both specified by ID.

```
merlin-cli -a <adaptation-id> --activity-type-parameters <activity-type-id>
```

Delete an adaptation

Remove an adaptation specified by ID from the adaptation service.

```
merlin-cli -a <adaptation-id> --delete-adaptation
```

Create a plan

Create a plan by uploading a plan JSON file. The JSON should contain all required fields including adaptation ID, plan name, start timestamp and end timestamp (YYYY-DDDThh:mm:ss). Adaptation ID must be for a valid adaptation, and activity instances must be valid according to the adaptation.

```
merlin-cli --create-plan <path-to-plan-json>
```

Query plans

Returns a list of existing plans indexed by plan ID.

```
merlin-cli --list-plans
```

Query an activity instance

Given a plan and activity instance ID, return activity instance metadata and parameter list.

```
merlin-cli -p <plan-id> --display-activity <activity-instance-id>
```

Update plan metadata

Update plan metadata via key/value pairs. Valid fields are: `adaptationId`, `startTimestamp`, `endTimestamp`, `name`.

```
merlin-cli -p <plan-id> --update-plan <field>=<value>
```

Update an activity instance

Update any attribute of an activity instance specified by unique ID. Note that activity instance parameters cannot be updated this way. Valid fields are: `startTimestamp` and `name`.

```
merlin-cli -p <plan-id> --update-activity <field>=<value>
```

Delete an activity instance

Delete an activity instance from a plan by ID.

```
merlin-cli -p <plan-id> --delete-activity <activity-instance-id>
```

Delete a plan

Delete a plan specified by ID.

```
merlin-cli -p <plan-id> --delete-plan
```

Update plan from a file

Upload a JSON file containing fields of a plan to be updated. All fields except adaptation ID may be updated this way. Any fields that are left out will remain unchanged. Note that individual activity instances cannot be updated this way, and including any activity instances in the plan update JSON will replace the current activity instance set with them.

```
merlin-cli -p <plan-id> --update-plan-from-file <path-to-json>
```

Append activity instances to a plan by file upload

This action will retain activity instances in the plan, but add all new activity instances specified by a JSON file.

```
merlin-cli -p <plan-id> --append-activities <path-to-json>
```

Download a plan JSON

Download plan in a JSON file format. Note that a downloaded plan JSON is different from the format for creating a plan in that activity instances are given as a map indexed by activity ID instead of a list.

```
merlin-cli -p <plan-id> --download-plan <output-path>
```

Simulate a plan

Kick off a simulation of a plan by ID, with results written to a file. Takes a sampling period (in microseconds) to determine how often to sample states for simulation results. Note that the local version of this command currently ignores the output path, and instead prints results to the console.

```
merlin-cli -p <plan-id> --simulate <sampling-period> <output-path>
```

Convert an APGen APF file to a Merlin JSON Plan file

APF files can be converted to Merlin input JSON file format. The action requires specifying a path to the APGEN adaptation `.aaf` files.

```
merlin-cli --convert-apf <path-to-apf-file> <output-path> <path-to-apgen-adaptation-directo
```