Libraries Used

## JNLua (Terasology version)

* + <https://github.com/MovingBlocks/JNLua>
  + <https://github.com/airminer/jnlua/wiki>
  + Library for Java which implements compatibility between Java and Lua
  + Uses Java Natives Interface to interpret Lua with C-based Lua libraries
    - This ensures peak performance
  + Implements Java’s JSR 223
    - <https://github.com/airminer/jnlua/wiki/JSR223Provider>
    - Helps to abstract the manipulation of the provided Lua engine and Lua scripts in general
    - This is the interface that the project uses to interact with JNLua
  + Rather than interfacing with the standard C Lua libraries, this version of JNLua is built to use Sangar’s variant of them, called Eris, which is detailed below
  + This library is what allows us to expose Java objects to the Lua environment, and with proper sandboxing (via Lua’s load() function), allows for a tightly-controlled, feature-rich environment for agents to execute in

## Eris (Sangar)

* + <https://github.com/fnuecke/eris>
  + Modification of the standard C Lua libraries to support heavy-duty persistence
  + Supports persistence of Lua objects, no matter the datatype\*
    - Userdata and C entities cannot be persisted, since they are runtime pointers to memory
  + In the Lua environment, the features of this variant are available under the “eris” module, which provides two primary functions:
    - persist(perms, obj)
      * This function serializes an object into binary form, which can be reconstructed in its original state from another Lua instance, even on another machine
      * obj is the object to be serialized
      * perms is a table which maps all non-persist-able objects, like userdata and C entities, to a “key”
    - unpersist(uperms, persisted)
      * This function takes a binary serialization of a Lua object, usually the output of the previous function, and creates an exact replica of the object as it was when it was passed to persist(). The object will be in exactly the same state, and it will be as if nothing happened
      * persisted is the binary serialization
      * uperms is a table which is a reverse-mapping of perms from the persist() function

## Gson (Google)

* + <https://github.com/google/gson>
  + A very powerful JSON library which supports easy serialization/deserialization of Java objects to and from the JSON format
  + Used by the presentation layer of our MAS software to abstract data transmission
  + Most useful part of this library are these two functions:
    - String toJson<T>(T, Class<T>)
      * Pass the object you want to serialize as the first argument
      * For ambiguous JSON conversion cases, pass the class of your object
        + (The project’s source code always does this to be safe)
      * Outputs a JSON representation of the object, which can be converted back to the object you passed in at a later point in a different Java environment, even on another machine
      * Output can be in a human-readable format if so desired
    - T fromJson<T>(String, Class<T>)
      * Pass the json representation of the object you want to create
      * For ambiguous JSON conversion cases, pass the class of your object
        + (The project’s source code always does this to be safe)
      * Outputs a copy of the Java object which the JSON string was representing

## TinyLog

* + <https://tinylog.org/v2/>
  + A lightweight, easy-to-use, highly-configurable logging facility for Java (and other JVM languages like Kotlin and Scala)
  + Makes logging very simple and easy
  + Used by the agent platform and agent executor instances to log errors, warnings, exceptions, and other useful debugging/analysis data
  + Due to configuration limitations, it is NOT used by agents for logging. Instead, agents are provided with a special Lua-Java hybrid object (called an Agent Facility within the project architecture) to support their logging needs, although it is mostly just redirection and some nice syntax and semantics

## MtgSaberLib-HTTPServer (Andrew Arnold)

* + <https://github.com/MtgSaber/MtgSaberLib_HTTPServer>
  + Written by me, Andrew, as a generalization of the presentation layer of the server architecture that I designed and used in my senior project
  + Used by the agent platform to simplify server logic
  + Has a few design flaws, but it is far better than what could be developed within the short timespan of this project, does the job rather well, and is relatively easy to work with