ELEMENTS SPECIFICATION

# DEFINITIONS

The process of progressing a matrix from the introductory state to solved state is called a **solution**.

A **solution** is composed of several **waves.**

A **wave** is composed of **tasks** which reference **packages**.

**Packages** are recorded separately, identified by **solution** and **wave**.

# COMPONENTS BEHAVIOR

# SERVER

The server is a persistent entity, one per processing cloud. The server can fulfill several tasks and has triggers. The tasks and signals the server performs and recognized are as follows.

1. **integer CREATE\_SOLUTION**(solution\_name, authentication\_key)

Server will check credentials and if they match, create a solution with the name of solution\_name in default state of SOLUTION\_UNPUBLISHED. It will return:

* 1. If creation succeeds: ID of solution
  2. If creation fails for database-related reasons: 0
  3. If creation fails due to lack of authentication: -1

1. **integer CREATE\_WAVE**(solution\_id, authentication\_key)

Server will check credentials and if they match, create a wave in the provided solution in the default state of WAVE\_UNPUBLISHED. It will return:

* 1. If creation succeeds: ID of wave
  2. If creation fails for database-related reasons: 0
  3. If creation fails due to lack of authentication: -1

# CLIENT

# DATA MODELS

# SOLUTION

Solution will contain fields:

1. ID
2. state (see states below)
3. current\_wave (numeric ID)
4. name (text)
5. solution (text, file reference)
6. created (timestamp)
7. updated (timestamp)
8. completed (timestamp)

Solution has states:

**0** SOLUTION\_UNPUBLISHED

**1** SOLUTION\_AVAILABLE

**2** SOLUTION\_COMPLETE

**9** SOLUTION\_BROKEN

# WAVE

Wave will contain fields:

1. ID
2. solution\_id (numeric id)
3. seq (numeric)
4. tasks (numeric)
5. state (see states below)
6. created (timestamp)
7. updated (timestamp)
8. completed (timestamp)

Wave has states:

**0** WAVE\_UNPUBLISHED

**1** WAVE\_AVAILABLE

**2** WAVE\_COMPLETE

**3** WAVE\_BROKEN

# SPLITTER

The working assumption for now is that a precomposed matrix in MTX format will be preloaded into a splitting system, which will reside on a server. This is reasonable since matrix splitting is considered to be an unparallelizable task.

The sequence of tasks will be as follows:

1. MTX file handle is opened
2. Matrix dimensions are read
   1. If matrix is not square, file is rejected – appropriate error message is produced
3. Matrix is conceptually divided into parts of configurable heights
4. File handles for the parts are opened
5. MTX is read and entries are passed on to appropriate files
6. MTX file handle is closed
7. Part file handles are closed
8. **FUTURE** Files are encrypted
9. Files are compressed
10. Files are recorded as parts in the database
11. 0-Wave is recorded composed of the parts
12. Splitter exits