Specification of the JSON file format used to save escape games
Draft 1

Arnaud Schmetz

December 2021

Each escape game, when saved, is saved as a JSON file which can be re-opened with the application to load the saved escape game. Only the escape game itself is saved: the results of the different checking or simulation operations are not saved in any way from one run to another.
Each JSON file will be saved in accordance with the following specification and must follow that same specification in order to be openable by the application.

This specification will break the JSON file format into blocks and define them separately for more clarity.

1 Overall structure

```
{
"Name" : "string",
// The name of the escape game.
"Description": "string",    Facultative
// The description of the escape game.
"Difficulty": "string",    Facultative
// The difficulty of the escape game.
"Players minimum": integer,    Facultative
// The inclusive minimal number of players, has to be superior or equal to 1.
"Players maxmum": integer,    Facultative
// The inclusive maximal number of players, has to be superior or equal to "Players minimum".
"Estimated duration": integer,    Facultative
// The estimated duration of the escape game in minutes.

"Physical elements": { Physical element 1, Physical element 2, ... },
   With at least 1 Physical element
   (see the related section for the specification of a Physical element)

// Lists all the physical elements of the escape game and their characteristics.
// Is defined as a physical element any person or physical object in the escape game.
// It may be an element whose type is predefined, like a door, a room, a player, etc.
// or any other physical object, which would then have the "CUSTOM" type.
// (e.g. : a setting asset, a computer, a curtain, etc.)

"Events": { Event 1, Event 2, ... },
   With at least 1 Event
   (see the related section for the specification of an Event)
// Lists all the events which can occur in the escape game and their characteristics.
// An event is defined as one or multiple actions occuring if one or
// multiple conditions, triggers are met.

"Hints": { Hint 1, Hint 2, ... },     Facultative
   If "Hints" is included : With at least 1 Hint
```

```
   (see the related section for the specification of a Hint)
// Lists all the hints available in the escape game and their characteristics.

"Problems": { Problem 1, Problem 2, ... },     Facultative
   If "Problems" is included : With at least 1 Problem
   (see the related section for the specification of a Problem)
// Lists all the problems or tasks the players may be confronted with and
// their characteristics.

"Countdowns": { Countdown 1, Countdown 2, ... }     Facultative
   If "Countdowns" is included : With at least 1 Countdown
   (see the related section for the specification of a Countdown)
// Lists all the countdowns of the escape game and their characteristics.
}
```

2 Physical Elements

Each physical element of the escape game is described in the JSON file as follows :

```
"Id": {
    // With Id being a string corresponding to the id of the physical element.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Description": "string",    Facultative
    // The description of the element.
    "Shape": "string",
    // The shape of the element as a svgpath, must be as described in the SVG Version 2
    // specification, by the EBNF at https://www.w3.org/TR/SVG/paths.html#PathDataBNF
    "Accessible": boolean,
    // Expresses if the element is initially accessible to the player(s),
    // at the beginning of the game.
    // An element is accessible if one or multiple player(s) can interract with it,
    // regardless of any spacial constrain.
    "Type": "string",
    // The type of the physical element is, that is.
    // The string must be one of the strings of this list :
    // KEY, DOOR, ROOM, PLAYER, GAME MASTER, etc
%(à compléter)
    "Position": [integer, integer],
    // The coordinates of the center of the element,
    // that is, the center of the shape formed by its svg path.
    // with the first integer being the x and the second the y, on the Javafx coordinates system.
    "States": {"Initial state", State 1, State 2, ... },    Facultative
    // Lists all the states the physical element may have and their repsective characteristics.
    // For example : a door can have the states "opened" and "closed".
    // "Initial state" is the Id of the initial state of the object.
```

With a state being saved as follow :

```
"Id": {
    // With Id being a string corresponding to the id of the state.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Name": "string",
    // The name of the state.
    "Description": "string",    Facultative
    // The description of the state.
    "Accessible": boolean,
```

```
    // Expresses if the state is initially accessible.
    // The state of an element is currently accessible if it can currently become the current state
    // of this object, regardless of any possible spatial limitation.
```

3 Events

Each event of the escape game is described in the JSON file as follows :

```
"Id": {
    // With Id being a string corresponding to the id of the event.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Name": "string",
    // The name of the event.
    "Description": "string",    Facultative
    // The description of the event.
    "Triggers": "JSONLogic",
    // A predicate. If it evaluates true, the action(s) of the event will be executed.
    // The predicate is saved following the JSONLogic format. https://jsonlogic.com/
%(définir plus ?)
    "Actions": [Action,Action,...]
    // A list of actions that will be executed if the triggers evaluate as true.
```

4 Hints

Each hint of the escape game is described in the JSON file as follows :

```
"Id": {
    // With Id being a string corresponding to the id of the hint.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Name": "string",
    // The name of the hint.
    "Description": "string",    Facultative
    // The description of the hint.
    "Position": [integer, integer]}     Facultative
    // The coordinates of the hint or the coordinates the hint is gonna be shown at.
    "Physical elements": [Id, Id, ...],     Facultative
    // The list of the Ids of all physical elements which form the hint.
```

5 Problems

Each problem/task of the escape game is described in the JSON file as follows :

```
"Id": {
    // With Id being a string corresponding to the id of theproblem.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Name": "string",
    // The name of the problem.
    "Description": "string",
    // The description of the problem.
    "Physical elements": [Id, Id, ...],     Facultative
    // The list of the Ids of all physical elements which are related to the problem.
```

6 countdowns

Each problem/task of the escape game is described in the JSON file as follows :

```
"Id": {
    // With Id being a string corresponding to the id of the countdown.
    // Each Id accros the entire file must be unique and usable as a variable name in Java 17.0.1.
    "Name": "string",
    // The name of the countdown.
    "Description": "string",    Facultative
    // The description of the countdown.
    "Duration": integer}
    // The duration of the countdown, in minutes.
%(seconds more flexible or useless?)
```