Subject File Documentation

To avoid overlap and repetition when creating results file templates, it should be outlined what the subject files should contain.

Required keys

- Subject identifier
- Associated config (use MD5 hash calculated from the config file)
- Model
- Task
- Results data

The *Model* and *Task* keys are clone from the associated config for the sake of readability when the experimenter is analysing their results.

Results data

All result values in HULTI-GEN are sorted in the following way:

```
session > group > trial > response value(s)
```

As implied by the plural, a given trial can contain multiple response values, for example in MUSHRA test.

Sessions and groups are object arrays, whilst trials are arrays of keys with an associated response value array.

The number of trials in each group is determined by the test type and the number of repetitions for each stimulus.

Update 2020-07-02

The above method is incompatible with multiple comparison tests, e.g. MUSHRA and ITU-R.1116-3.

This is due to the fact that, with the current model, one group of stimuli in a multiple comparison test will be represented only as one trial.
This can overcomplicate the design and limit expandability if there were different results structures for each type of test.]

The solution is to structure the results data as not to rely on responses per trial, but associate response values to the stimuli themselves using the following structure:

```
session > group > stimulus > response(s)
```

Responses in the stimulus keys can be arrays where each element is a repetition of that particular stimulus. In a single test stimulus per trial scenario such as 2AFC,
the following snippet shows how the response data is stored

```
set session[0]::group[0]::stimuli[0]::response[0] "y" // Y means they correctly identified the stimulus.
```

For a multiple comparison grading test between 6 stimuli, the the data is stored in multiple stimuli at once in the following way:

```
set session[0]::group[0]::stimuli[0]::response[0] 5
set session[0]::group[0]::stimuli[1]::response[0] 90
set session[0]::group[0]::stimuli[2]::response[0] 45
set session[0]::group[0]::stimuli[3]::response[0] 100
set session[0]::group[0]::stimuli[4]::response[0] 0
set session[0]::group[0]::stimuli[5]::response[0] 10
```

The target session and group can still be randomised and pre-baked into the subject file as before, but the access method is test dependent and deduced at runtime. Presentation can either be pre-baked or calculated at run-time, though it doesn't matter, just as long as the data ends up in the correct place.

The advantage of this method is that it reduces setup and run-time complexity, and improves readability as you can see the results on a per stimulus basis and still compare the values between stimuli vertically with your eyes. Finally, it resembles the output CSV file much more closely than the previous method.