Documentation of the JSON input format for the SAPHIRE Risk Assessment Tool

In this document, a detailed description of the SAPHIRE JavaScript Object Notation (JSON) input format is presented. Figure 1 shows the general structure of the SAPHIRE JSON input, and each section is explained in detail. A fault-tree example from the SPAR model is used to describe each section of the input, Figure 2 shows the fault-tree example.

Figure 1: General structure of the SAPHIRE JSON input.

**Table

Description automatically generated with low confidence**

Figure 2: Fault tree example from the SPAR model.

# SAPHIRE Solver Input

The input starts with the version and a whole block of the SAHPIRE solver input as can be seen in Figure 3. The solver input block includes other sub-blocks that contains all the information about the event trees and fault trees and those blocks are named: Header, system gate list, fault tree list, and the event list, as can be seen in Figure 1. In the next sections, each sub-block will be explained in detail.

A picture containing text

Description automatically generated

Figure 3: Beginning of the input.

## Header

The header sub-block contains: the project path, the event tree description, additional information, truncation parameters information and workspace pair information. Table 1 describes each in line in the header sub-block.

Table 1: Header sub-block description.

|  |  |
| --- | --- |
| **Project Path** | |
|  | |
| "projectpath": | The path where the model is located, has nothing to do with the solve but it’s useful to identify where is it coming from. |
| **Event Tree** | |
|  | |
| "name": | Event tree name |
| "number" | Internal number and it should be unique to every event tree |
| "initevent": | Internal number of the initiating event and it should be unique |
| "seqphase": | The sequence phase of the event tree |
| **Additional Information** | |
|  | |
| "flagnum": | Default is 0, it doesn’t matter for the solve but it’s coming back for the solve, it’s a definition of a state, i.e., |
| "ftcount": | Default is 1, meaning that we 1 fault tree per JSON input file. But it will be different if we’re solving sequence because we will have several event trees. |
| "fthigh": | Fault tree unique ID of the highest number fault trees |
| "sqcount": | Always Zero, the total number of sequences that I am solving in this event tree |
| "sqhigh": | Always Zero, the unique internal number in an event tree. |
| "becount": | 4 house events + # of basic events in the fault tree. |
| "behigh": | The highest ID number of the basic events |
| "mthigh": | Model type is always 1, unless there are other models, e.x. seismic model |
| "phhigh": | Phase number is always 1 |
| **Truncation parameters** | |
|  | |
| "ettruncopt": | "NormalProbCutOff", “NoProbCutOffET”, “CondProbCutOff” |
| "fttruncopt": | "GlobalProbCutOff”, “NoProbCutOff”, “SystemProbCutOff” |
| "sizeopt": | "ENoTrunc",”ESizeTrunc”, “EZoneTrunc” |
| "ettruncval": | Event tree truncation value |
| "fttruncval": | Fault tree truncation value |
| "sizeval": | Always 99, it’s related to the size option. |
| "transrepl": | Always false in this case, because we are not transforming any information i.e. cable trays failure in a fire event. |
| "transzones": | Always false, same explaination as above. |
| "translevel": | Always zero, same explaination as above. |
| "usedual": | Always false, it is a way to keep certain cut sets to keep certain high probability events even if they exceed or below the truncation limit. |
| "dualcutoff": | Always zero, same explaination as above. |
| **Work space pair** | |
| The “I” is for the iniatin event and the other one is for the whole process | |
| "ph": | Phase is always 1 |
| "mt": | Model type is always 1, unless there are other models, e.x. seismic model |

## System gate list

The system gate sub-clock contains general information about the fault tree. Table 2 describes each line in the system gate list sub-block.

Table 2: System gate list sub-block description.

|  |  |
| --- | --- |
| **System gate list** | |
|  | |
| "name": | Fault tree name. |
| "id": | Fault tree unique ID, same as in “fthigh” only it’s a fault tree because we only have one in this case |
| "gateid": | Top gate ID in this fault tree. |
| "gateorig": | Same as “gateid” |
| "gatepos": | Gate position: a way to remember where this gate shows up in the gate table: Always zero in our case. |
| "eventid": | The fault tree itself is considered to be an event and has another unique ID. |
| "gatecomp": | Gate complement: Same as “gateid”, rarely it is different, if I wanted to treat the complemented gate differently. |
| "comppos": | Complementary position: Always zero, same explanation as above. |
| "compflag": | Complementary flag: Always empty, same explanation as above. |
| "gateflag": | How we’re going to treat the gate: Always empty for the synthetic. |
| "gatet": | Always empty for the synthetic. |
| "bddsuccess": | Always false for the synthetic. |
| "done": | Always false for the synthetic. |

## Fault tree list

The fault tree list sub-block contains the fault tree and grate information in two sub-blocks named: Fault tree header and gate list. Table 3 describes each line in the fault tree list sub-block.

Table 3: Fault tree list sub-block description.

|  |  |
| --- | --- |
| **Fault tree header** | |
|  | |
| "ftid": | Fault tree unique ID, same as “fthigh” only if we have one fault tree. |
| "gtid": | Top gate ID, same as “gatecomp”, and “gateid” |
| "evid": | Fault tree event ID, same as "eventid" |
| "defflag": | Default is zero., it helps define the state, similar as above. |
| "numgates": | Number of gates in this fault tree. |
| **Gate list** | |
|  | |
| "gateid": | The gate unique ID |
| "gatetype": | The logical gate type (OR, AND,..) |
| "numinputs": | Number of inputs(gates or/and basic events) in the fault tree, corresponding to the specified gate |
| "eventinput": | Basic events unique ID numbers. |
| “gateinput” | Gates unique ID numbers. |
| **"compeventinput”** | Complemented Basic events unique ID numbers. |
| **“compgateinput”** | Complemented Gates unique ID numbers. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Event list

The event list sub-block contains many sub-blocks for each basic event information. Table 4 describes each line in the event list sub-block. In addition, every event tree list should contain all the basic events plus 4 additional house events that should be there no matter how many basic events the fault trees contain.

Table 4: Event tree list sub-block description.

|  |  |
| --- | --- |
| **Event list** | |
|  | |
| "id": | Event unique ID |
| "corrgate": | Always zero, except for the fault tree event, it should be the top gate number. |
| "name": | Name of the basic event/ name of the 4 events (<FALSE>,<TRUE>, <PASS>,<”fault tree name”> |
| "evworkspacepair": | Phase is always 1.  Model type is always 1, unless there are other models, i.e.. seismic model. |
| "value": | Failure probability value for the basic event.  Value for <TRUE> always 1  Value for <FALSE> always 0  Value for <FALSE> always 1  Value for <”fault tree name”>, however it’s been defined in the process flag but most of the time it is going to be 1. |
| "initf": | Always empty |
| "processf": | Always empty (related to the event tree) |
| "calctype": | Always 1: failure probability |