INL/RPT-23-75066-Revision-0

Refining Processing Engines from SAPHIRE

October 2023

Egemen Mutlu Aras, Asmaa Salem Amin Aly Farag, Stephen Ted Wood, Jordan Thomas Boyce

INL/RPT-23-75066-Revision-0

Refining Processing Engines from SAPHIRE

Egemen Mutlu Aras, Asmaa Salem Amin Aly Farag, Stephen Ted Wood, Jordan Thomas Boyce

October 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

INL/XXX-XX-XXXXX

Refining Processing Engines from SAPHIRE: Initialization of Fault Tree/Event Tree Solver

September 2023

Input and Output File Format of SAPHSOLVE

Egemen M. Aras, Asmaa S. Farag, S. Ted Wood, and Jordan T. Boyce
Idaho National Laboratory

INL/XXX-XX-XXXXX

Refining Processing Engines from SAPHIRE: Initialization of Fault Tree/Event Tree Solver

Input and Output File Format of SAPHSOLVE

Egemen M. Aras, Asmaa S. Farag, S. Ted Wood, and Jordan T. Boyce
Idaho National Laboratory

September 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Under the U.S. Department of Energy, Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

ABSTRACT

System Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) has been extensively employed for over 35 years to model risk-important systems and quantify risk models. As a well-established probabilistic risk assessment (PRA) tool, SAPHIRE has continuously tracked computational trends and received regular updates. Despite its ongoing evolution, there remains a need for further enhancements, particularly in dealing with the quantification of exceptionally large models. These improvements could include algorithmic advancements, harnessing the power of parallel computing, and exploring the potential benefits of cloud computing solutions.

From these aspirations came the notion of a remote solve option, which evolved into a dedicated project within the SAPHIRE development team. A significant outcome of this initiative is SAPHSOLVE, an engine extracted from the SageRisk API, and is designed specifically for remote-solving capabilities. The project has been marked by a series of discoveries that have brought undocumented aspects to light. Among these revelations is the intricacy of the input and output format for the SAPHSOLVE engine. This document serves the purpose of delineating the precise formats for both input and output, as they form an indispensable foundation for using this engine.

Documenting these formats is a pivotal step in providing the capability to construct models or transform existing ones into a compatible SAPHSOLVE format, which is imperative for facilitating rigorous testing and comparison of results against those of the internal integrated solver or other external solvers.

Section 1 offers a succinct introduction to both SAPHIRE and SAPHSOLVE and is followed by Section 2, which outlines the roadmap for enhancing SAPHSOLVE. The core of this report is Section 3, which elucidates the details of the input and output file formats. To provide a tangible illustration of these formats, a rudimentary example has been compiled. Appendix A contains an input file example, and Appendix B contains an example output (cut set result) file.

SAPHSOLVE represents a novel external solving mechanism developed by the SAPHIRE team, although it has not yet reached the full spectrum of capabilities possessed by SAPHIRE’s internal solver. A comprehensive outlook on the future of SAPHSOLVE is discussed in Section 4.

Page intentionally left blank

CONTENTS

FIGURES

TABLES

ACRONYMS

API	Application Programing Interface
BDD	Binary Decision Diagram
CCS	Containment Cooling System
ECS	Emergency Cooling System
GUI	Graphical User Interface
INL	Idaho National Laboratory
IRRAS	Integrated Reliability and Risk Analysis System
JSCUT	JavaScript Object Notation Output File
JSINP	JavaScript Object Notation Input File
JSON	JavaScript Object Notation
LOOP	Loss of Offsite Power
MCUB	minimum cut set upper bound
NRC	(U.S.) Nuclear Regulatory Commission
PRA	Probabilistic Risk Assessment
SAPHIRE	System Analysis Programs for Hands-on Integrated Reliability Evaluations
SPAR	Standardized Plant Analysis Risk
ZBDD	Zero-suppressed Binary Decision Diagram

Page intentionally left blank

Refining Processing Engines from SAPHIRE: Initialization of Fault Tree/Event Tree Solver

1. INTRODUCTION

System Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) [1] serves as the successor to the Integrated Reliability and Risk Analysis System (IRRAS) [1]. SAPHIRE is a critical and distinctive software used by the U.S. Nuclear Regulatory Commission (NRC) to validate PRA models and make risk-informed decisions.

Having successfully served various industries, including nuclear, space, and other risk-significant sectors, such as chemical facilities, for over 35 years, SAPHIRE has proven its reliability. While it has adapted to new computational trends, the growing complexity of models and evolving expectations has presented new challenges.

As these models increase in complexity, their quantification becomes increasingly demanding, sometimes surpassing the available resources. The challenges the legacy probabilistic risk assessment (PRA) tools encounter are addressed in a pair of works [3,4] conducted at the Idaho National Laboratory (INL). Additionally, the computational landscape and expectations are evolving rapidly, prompting the SAPHIRE team to devise an enhancement roadmap. Since the initiation of this roadmap, significant progress has been made to enhance the software.

This report primarily delves into the comprehensive explanation of the input and output file formats of one of SAPHIRE's extracted processing engines, namely the fault tree/event tree solver (hereafter referred to as SAPHSOLVE). Furthermore, this report provides instructions for running SAPHSOLVE as a standalone application without the graphical user interface (GUI). These efforts form the foundation for the advancement of SAPHSOLVE, paving the way to benchmark and profile the software. These benchmarks aim to identify the limitations of the engine and pinpoint any performance bottlenecks. Ultimately, this initiative is expected to result in a dedicated enhancement roadmap for SAPHSOLVE.

Section 2 outlines the roadmap to enhance SAPHSOLVE, while Section 3 provides an in-depth exploration of SAPHSOLVE's input and output file structures. The report concludes in Section 4 with a discussion on future endeavors and directions.

2. ROADMAP FOR ENHANCEMENT

Previous annual reports detailing SAPHIRE enhancements comprehensively present a roadmap for the system's improvement and associated findings. The proposed roadmap depicted in Figure 1 focuses on augmenting quantification speed and cut set generation. The PRA Model Solver within the figure represents SAPHIRE itself. SAPHIRE's operation hinges on a PRA Model input file, which can be generated synthetically or in a more general manner. The initial phase of enhancement prioritized synthetic models to work around SAPHIRE's limitations, with the transition to generic models occurring during validation.

While SAPHIRE offers a dedicated model creation editor, the potential need for a Model Converter is apparent. This tool would facilitate the adaptation of models originally designed for other PRA tools, thus expanding SAPHIRE's versatility.

The research roadmap for SAPHIRE follows a cyclic structure. It commences with diagnostic assessments, comprising two distinct steps. The first step involves a performance evaluation through benchmarking, followed by an extensive analysis of the source code. This analysis aims to identify performance bottlenecks and isolate segments contributing significantly to processing time.

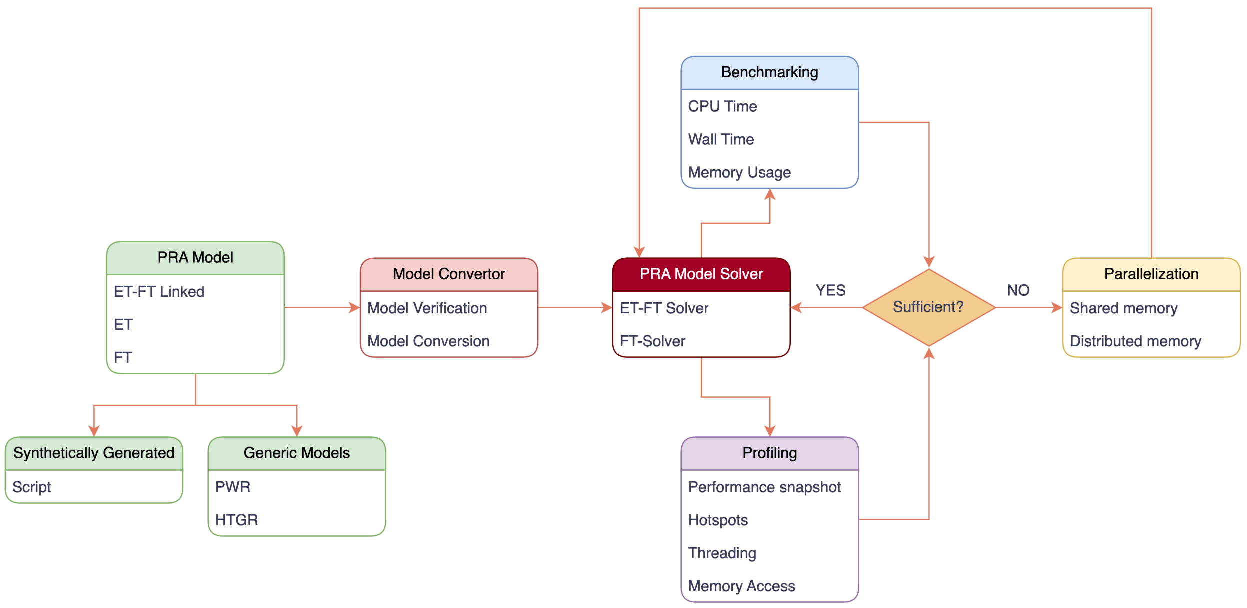


Figure 1. Research roadmap for increasing quantification and cut set generation speed for SAPHSOLVE.

These findings lay the foundation for optimizing the quantification engine. This objective can be achieved through two approaches: incorporating algorithmic updates and integrating parallel processing capabilities. These measures collectively enhance the efficiency and functionality of the quantification engine.

While this work elaborates on the enhancement roadmap for SAPHSOLVE in terms of quantification and cut set generation speed, it also underpins the broader PRA Model. The extraction of the fault tree/event tree solver marks the initial step and is followed by the generation of the PRA Model, which represents the subsequent phase of development.

3. EMERGENCE OF SAPHSOLVE

SAPHSOLVE is a remarkable outcome resulting from the concerted efforts to establish a remote-solving capability. This achievement, prominently featured among the fiscal year 2021 accomplishments, offers SAPHIRE users a practical avenue for remote-solving operations. This breakthrough was realized through the extraction of the solving engine from the SageRisk API, which paved the way to creating SAPHSOLVE. Subjecting the SAPHSOLVE engine to thorough testing input and output files is an indispensable requisite, warranting meticulous alignment with the internal integrated solver.

Upon a comprehensive assessment of the available options, it was judiciously determined that the use of JavaScript Object Notation (JSON) would optimally satisfy the prerequisites of the remote solver. This selection supported the format’s inherent attributes to be a lightweight and versatile data-interchange framework. JSON’s adeptness at encapsulating the necessary data and facilitating seamless communication between the remote solver and its components rendered it a fitting choice for the creation of input and output files. This judicious selection ensured that SAPHSOLVE could seamlessly perform its designated functions and yield outcomes that can be scrutinized and compared against the internal integrated solver.

The adoption of JSON for structuring the input and output files underscores the commitment to precision and efficacy within the realm of remote solving, offering the proactive approach to accommodate technological advancements while maintaining compatibility and reliability. This strategic decision forms a pivotal cornerstone in the continuous refinement and evolution of SAPHIRE's capabilities, affirming its status as a tool that remains at the forefront of PRA.

Users are expected to adhere to the following naming convention for the input and output files:

- Input files should be named in the format `[input-filename].JSInp.` This nomenclature is chosen to distinctly identify these files from their output counterparts.
- Output files should adopt the nomenclature `[output-filename].JSCut`.

By maintaining these naming conventions, users ensure clear differentiation between input and output files while adhering to the JSON format for content representation.

This report segment serves to elaborate on the specifications for the input file, as delineated in Section 3.1, as well as the specifications pertaining to the output file, as described in Section 3.2.

3.1. Input File Specification

The input file uses the `JSInp` extension, encompassing essential logic and foundational data related to event failures that is aimed at generating cut sets. All the pertinent details required for the resolution of fault tree or event tree sequences are encapsulated within this file format.

Figure 2. General structure of the SAPHSOLVE input file.

In cases of fault trees, each `JSInp` file corresponds to a single fault tree, housing the relevant information. In scenarios involving event trees, a `JSInp` file for an event tree comprises an amalgamation of fault tree logic, sequences, and logical constructs derived from combinations within the fault tree. This amalgamation serves to define event tree accident sequences, which are supplemented by fundamental event particulars essential for the derivation of cut sets.

Furthermore, these `JSInp` files encompass the truncation levels, a pivotal component in the analysis process.

Figure 2 depicts the overarching framework of an input file used by SAPHSOLVE. Each component within this structure is comprehensively elucidated. For users to construct their own input files to employ SAPHSOLVE, adherence to these structure guidelines is imperative. An interconnected event tree instance derived from a SAPHIRE basics workbook [5] report model illustrates the various input sections.

For a detailed breakdown of each section within the input, an event tree example is presented in Figure 3. Additionally, the fault trees associated with and linked to this event tree are visually represented in Figure 4 and Figure 5, respectively. These visual aids offer a comprehensive understanding of the input file's structure and its interplay with the underlying event and fault tree models.

The attached input file is provided in Appendix A, "Input Demo Model" for reference. It is important to note that SAPHSOLVE exclusively considers failure sequences, disregarding any OK or SUCCESS sequences. Consequently, the input model solely incorporates sequences two and three pertaining to SMALL-RELEASE and LARGE-RELEASE. The initiating event involves a loss of offsite power (LOOP), and the safety framework comprises two systems: the emergency cooling system (ECS) and the containment cooling system (CCS).

The breakdown of sequences is as follows:

- Sequence 1 postulates the successful operation of both safety systems, resulting in the absence of any release.

- Sequence 2 hypothesizes the failure of the ECS while the CCS remains functional, yielding a scenario of small release.
- Sequence 3 envisions the simultaneous failure of both safety systems, leading to a scenario of large release.

Figure 4 and Figure 5 illustrate the fault trees corresponding to the ECS and CCS, along with their associated failure combinations. These graphical representations provide a visual grasp of the intricate relationships within the system and the potential failure pathways.

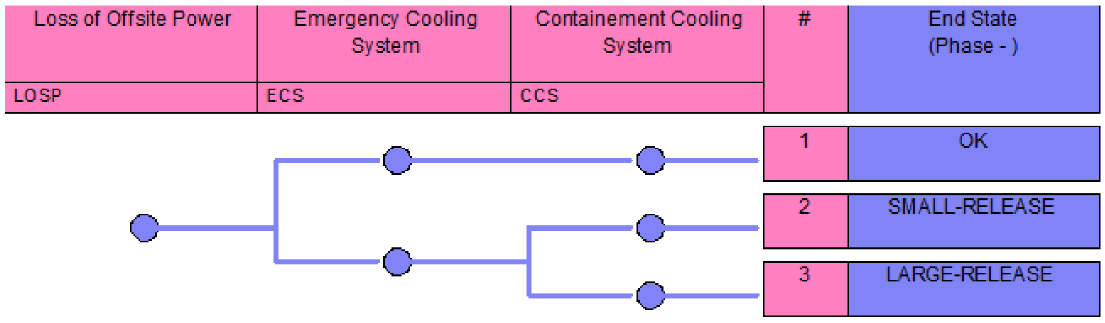


Figure 3. LOOP event tree.

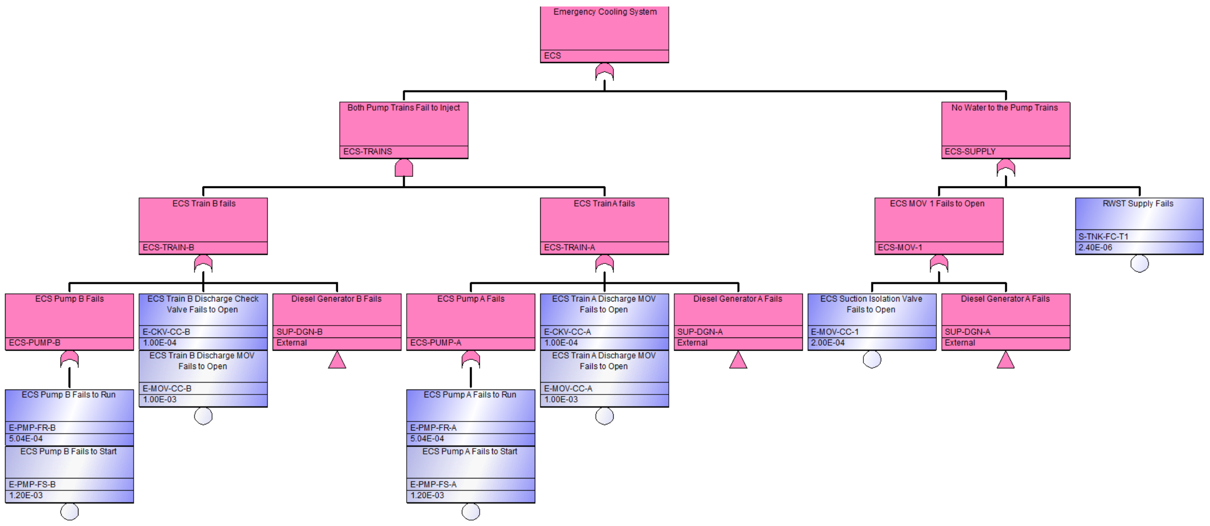


Figure 4. ECS fault tree.

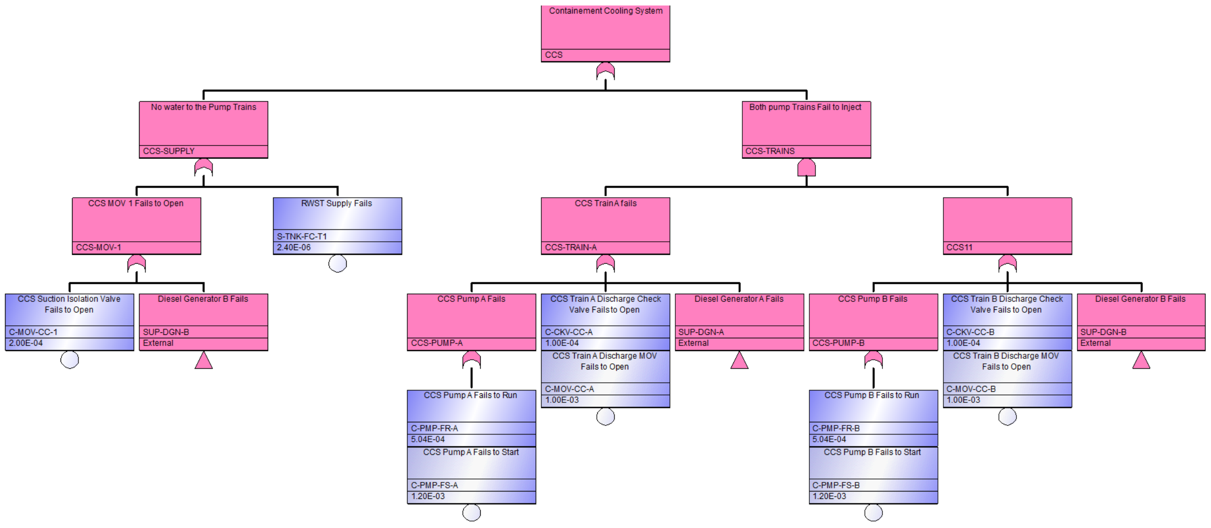


Figure 5. CCS fault tree.

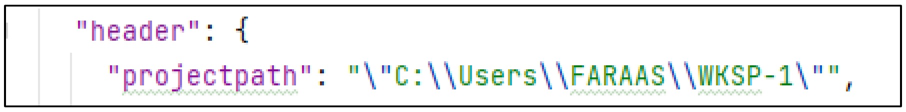
3.1.1. Header Information

The header block serves as a concise overview of a given `JSInp` file, encapsulating key details that users need to define. Within the header, specific properties must be specified. This assortment of information commonly includes the following:

- **Project Path**: This denotes the location of the project.
- **Event Tree Description**: A brief description of the event tree considered.
- **Model Specific Information**: Pertinent information unique to the model being analyzed.
- **Truncation Parameters**: Parameters relevant to the truncation process.
- **Workspace Pair Information**: Information about workspace pairs used in the analysis.

For an exhaustive breakdown of each line in the header block of a `JSInp` file, Table 1 through Table 5 and Figure 6 through Figure 10 provide comprehensive descriptions. These tables serve as a valuable reference, detailing the significance of each line and the associated information it encompasses. It is important to emphasize that all parameters are case-sensitive.

3.1.1.1. Project Path



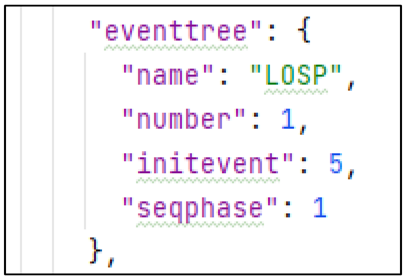
```
  "header": {
    "projectpath": "\"C:\\Users\\FARAAS\\WKSP-1\"",
```

Figure 6. Project path example.

Table 1. Project path explanations and instructions.

Argument	Explanation	Instructions
"projectpath"	Including the model's location path in the header serves as a useful reference for its origin and futureproofing, even though it does not directly impact the solution. This practice aids in maintaining traceability, documentation, and adaptability over time.	Note for the user to identify the results of the solve.

3.1.1.2. *Event Tree Description*



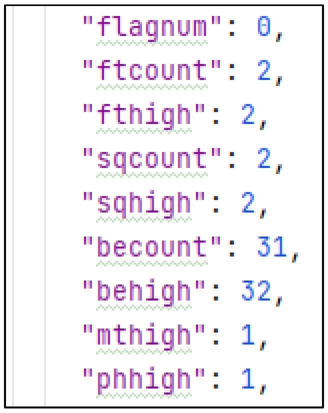
```
"eventtree": {
  "name": "LOSP",
  "number": 1,
  "initevent": 5,
  "seqphase": 1
},
```

Figure 7. Event tree example.

Table 2. Event tree block explanations and instructions.

Argument	Explanation	Instructions
"name"	Event tree name.	The user should choose a name for their event tree. It can contain 48 uppercase, alphanumeric characters. Forbidden characters include `, @, #, $, ^, *, +, =, {, [,], }, |, \, /, :, ', :, ", ',.,?,,.
"number"	Internal number. It should be unique to every event tree.	The user should choose a unique and different integer number for each event tree.
“initevent”	Internal number of the initiating event, and it should be unique for each initiating event.	The user should choose a unique and different integer number for each initiating event that is related to each event tree.
"seqphase"	The sequence phase of the event tree, the default is 1.	The user needs to identify which phase level they are applying.

3.1.1.3. *Additional Information*



```
"flagnum": 0,
"ftcount": 2,
"fthigh": 2,
"sqcount": 2,
"sqhigh": 2,
"becount": 31,
"behigh": 32,
"mthigh": 1,
"phhigh": 1,
```

Figure 8. Example of the additional information section.

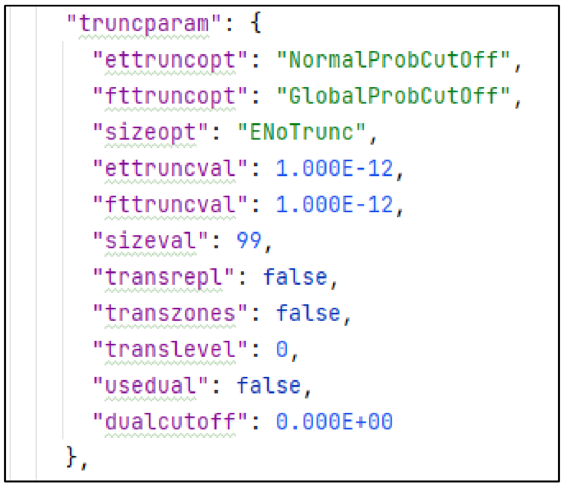
Table 3. Additional information section explanations and instructions.

Argument	Explanation	Instructions
"flagnum"	The flag sets are used to indicate modifications to particular events on a sequence-by-sequence	The user can learn more about flag sets by visiting the advanced manual

Argument	Explanation	Instructions
	basis (or fault tree logic). Default is 0, which means it does not matter for the solution. It is a definition of a state.	[6].
"ftcount"	Default is 1, which means there is one fault tree per JSON input file. It will be different, however, if the user is solving sequences because they includemultiple fault trees. In this example, there are ECS and CCS fault trees.	The user should identify the number of fault trees in their input.
"fthigh"	Unique fault tree ID of the highest number fault trees.	The user should identify the highest fault tree integer ID number among all the fault trees in their input.
"sqcount"	The total number of sequences that the user is solving in this event tree. It will be zero if the user is solving only one fault tree.	The user should identify the total number of sequences in this subset of the event tree.
"sqhigh"	The highest sequence with a unique internal number in the list of sequences. It will be zero if the user is solving only one fault tree.	The user should identify the highest number of the sequences included in their input, and they should put zero if solving only a fault tree.
"becount"	Total number of basic events in all listed fault trees plus the associated basic event for all listed fault trees plus the initiating event (if needed) plus three SAPHIRE generated basic events (i.e., "<TRUE>," "<FALSE>," "<PASS>").	The user should add the total number of their basic events plus three essential basic events plus the number of initiating events plus the number of fault trees. The three basic events are events that need to be passed through the input. It is explained in more details in the event list block.
"behigh"	The highest integer ID number of the basic events.	The user should identify the highest basic event integer ID number among all the fault trees in their input.
"mthigh"	Model types can range between 1–32 and are designed to have different failure potentials for components and different modeling options for user-defined assessments. The default is 1, which indicates that the user is only solving internal events. If the model contains external events, such as seismic, fire, flood, then it has to be 2, 3, 4, repectively.	The user should use the model type that describes their input—whether they are using internal or external events in their input. `mthigh` = 1, in case of internal events and it is the default. `mthigh` = 2, in case of external seismic events. `mthigh` = 3, in case of external fire events. `mthigh` = 4, in case of external flood events. For more details, please visit the advanced SAPHIRE manual [6].
"phhigh"	Phase numbers can range between 1–100 and are designed to interrupt the solving process at logical points, such as core damage, plant	The user can identify which level of PRA they need to perform. Reference that advanced SAPHIRE

Argument	Explanation	Instructions
	damage state, and release, in a nuclear power plant PRA. The default is 1.	manual for more information [6].

3.1.1.4. *Truncation Parameters*



```
"truncparam": {
  "ettruncopt": "NormalProbCutOff",
  "fttruncopt": "GlobalProbCutOff",
  "sizeopt": "ENoTrunc",
  "ettruncval": 1.000E-12,
  "fttruncval": 1.000E-12,
  "sizeval": 99,
  "transrepl": false,
  "transzones": false,
  "translevel": 0,
  "usedual": false,
  "dualcutoff": 0.000E+00
},
```

Figure 9. Example of truncation parameters block.

Table 4. Truncation parameters block explanations and instructions.

Argument	Explanation	Instructions
"ettruncopt"	Event tree cut set truncation parameters and options.	The user must choose the cut set truncation parameters. There are three options: `NoProbCutOffET`, where no truncation will be performed. `NormalProbCutOff`, where it uses the cut-off value in the adjacent cut-off field. `CondProbCutOff`, where it uses the cut-off value stored with the fault tree record. For more details, please visit the SAPHIRE basics manual [5]
"fttruncopt"	Fault tree truncation parameters and options.	The user must choose the cut set truncation parameters. There are three options: `NoProbCutOff`, where no truncation will be performed. `GlobalProbCutOff`, where it uses the cut-off value in the adjacent cut-off field. `SystemProbCutOff`, where it uses the cut-off value stored with the fault tree record. For more details, please visit the SAPHIRE basics manual [5]

Argument	Explanation	Instructions
"sizeopt"		The user must choose the size of the cut sets. There are three options: ENoTrunc, which means that the number of events in a cut set will not affect whether the cut set is retained or discarded. ESizeTrunc, which means that the cut sets having more than specified number will be discarded. EZoneTrunc, which means that the cut sets with more Zone Flagged Events than specified in the adjacent text box will not be retained. For more details, please visit the SAPHIRE basics manual [5]
"ettruncval"	Event tree truncation value.	The user should identify the truncation value for solving the event tree.
"fttruncval"	Fault tree truncation value.	The user should identify the truncation value for solving the fault tree.
"sizeval"	Size of cut sets, default is 99.	The user should enter in the maximum size of cut sets they want to retain. If the `ENoTrunc` option was chosen, the solver ignores it.
"transrepl"	Always false in this case because the user is not transforming any information (i.e., cable trays failure in a fire event).	—
"transzones"	Always false, same explanation as `transrepl`.	—
"translevel"	Always zero, same explanation as `transrepl`.	—
"usedual"	Always false, it is a way to keep certain cut sets at certain high probability events even if they exceed or are below the truncation limit.	—
"dualcutoff"	Always zero, same explanation as `usedual`.	—

3.1.1.5. Workspace Pair Information

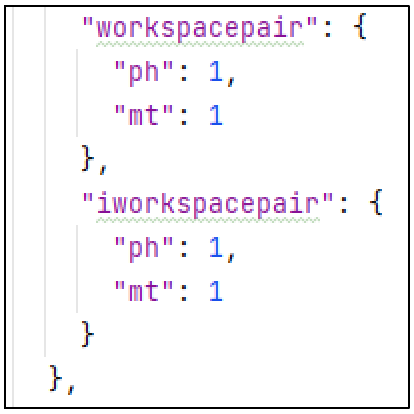

```
    "workspacepair": {
      "ph": 1,
      "mt": 1
    },
    "iworkspacepair": {
      "ph": 1,
      "mt": 1
    }
  },
```

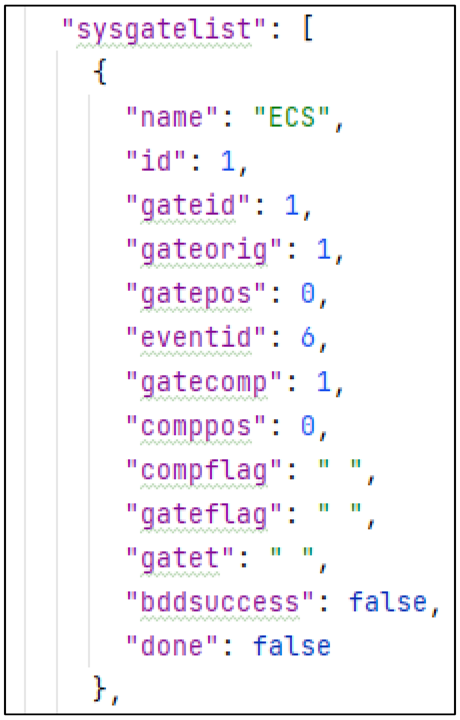
Figure 10. Example of workspace pair block. The “i” is for the initiating event**.**

Table 5. Workspace pair block explanations and instructions.

Argument	Explanation	Instructions
"ph"	Phase numbers can range between 1–100 and are designed to interrupt the solving process at logical points, such as core damage, plant damage state, and release, in a nuclear power plant PRA. The default is 1.	The user can identify which level of PRA they need to perform. Refer to the advanced SAPHIRE manual to learn more information [6].
"mt"	Model types can range between 1–32 and are designed with different failure potentials for components and different modeling options for user-defined assessments. The default is 1, which indicates the user is only solving internal events. If the model contains external events, such as seismic, fire, flood, then it has to be 2, 3, 4, repectively.	The user should use the model type that describes their input—whether they are using internal or external events in their input. `mthigh` = 1, in case of internal events. This is the default. `mthigh` = 2, in case of external seismic events. `mthigh` = 3, in case of external fire events. `mthigh` = 4, in case of external flood events. For more details, please visit the advanced SAPHIRE manual [6].

3.1.2. System Gate List

The system gate sub-block contains general information about the fault tree. Table 6 and Figure 11 describe each line in the system gate list sub-block.



```
"sysgatelist": [
  {
    "name": "ECS",
    "id": 1,
    "gateid": 1,
    "gateorig": 1,
    "gatepos": 0,
    "eventid": 6,
    "gatecomp": 1,
    "comppos": 0,
    "compflag": " ",
    "gateflag": " ",
    "gatet": " ",
    "bddsuccess": false,
    "done": false
  },
```

Figure 11. Example of system gate list block.

Table 6. System gate list block explanations and instructions.

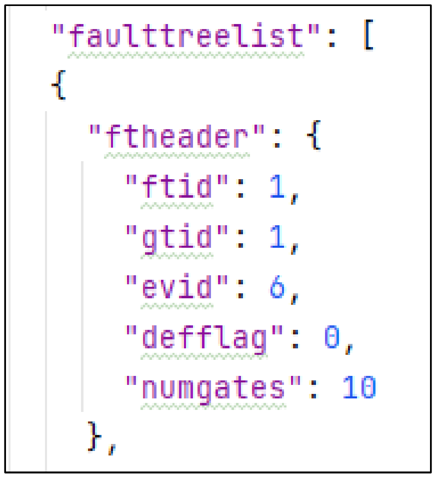
Argument	Explanation	Instructions
"name"	Fault tree name.	The user should choose a name for each fault tree in their input. The name can contain alphanumeric characters and has a 48-character maximum. The name cannot contain spaces, except the following {}.
"id"	Fault tree unique ID, same as in `fthigh`, only it is a fault tree because the user only has one in this case.	The user should choose a unique and different integer number for each fault tree.
"gateid"	Top gate integer ID in this fault tree.	The user should choose a unique and different integer number for the top gate integer ID for each fault tree.
"gateorig"	Same as `gateid`.	Same as `gateid`.
"gatepos"	0 – Special Use is to be determined.	—
"eventid"	The fault tree is an event and has another unique ID.	The user should consider each fault tree as an event with its own unique integer ID. The user should add it to the event list.
"gatecomp"	Gate complement: Same as	The user should make `gatecomp` equal to the

Argument	Explanation	Instructions
	`gateid`. It is rarely different if the user wanted to treat the complemented gate differently.	`gateid`, but it may be different if it is used as a complemented gate.
"comppos"	0 – Special Use is to be determined.	—
"compflag"	Complementary flag: Always empty, same explanation as `comppos`.	—
"gateflag"	How the user will treat the gate.	Same as basic event process flag.
"gatet"	Always empty except for the synthetic.	—
"bddsuccess"	Always false except for the synthetic.	—
"done"	Always false except for the synthetic.	—

3.1.3. Fault Tree List

The fault tree list sub-block contains the fault tree and gate information in two sub-blocks named: fault tree header and gate list. Table 7 and Figure 12 describe each line in the fault tree list sub-block.

3.1.3.1. Fault Tree Header



```
"faulttreelist": [
{
  "ftheader": {
    "ftid": 1,
    "gtid": 1,
    "evid": 6,
    "defflag": 0,
    "numgates": 10
  },
```

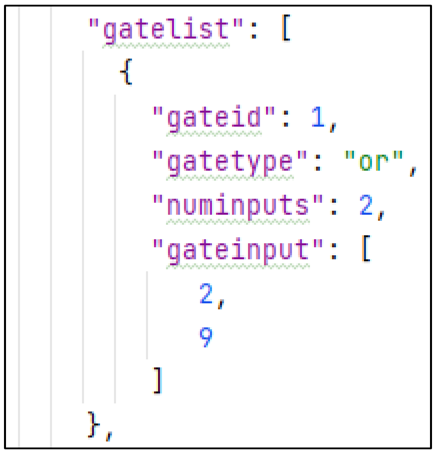
Figure 12. Example of fault tree block. For each fault tree in the input, the user needs to have a fault tree header related to each one.

Table 7. Fault tree block explanations and instructions.

Argument	Explanation	Instructions
"ftid"	Fault tree unique ID. This is the same as `fthigh` if the user has one fault tree.	The user should write the unique integer ID for the specific fault tree.
"gtid"	Top gate ID, same as `gatecomp` and `gateid`.	The user should write the unique integer number of the top gate integer ID for this specific fault tree.
"evid"	Fault tree event ID, same as `eventid` in the system gate list information.	The user should consider each fault tree as an event. Each fault tree has its own unique integer ID, which should be added to the event list. The user should write the ID here.
"defflag"	The flag sets are used to indicate modifications to particular events on a sequence-by-sequence basis (or fault tree logic). Default is 0, it does not matter for the solution but it does matter when coming back for the solution. This is a definition of a state, i.e.,	The user can learn more about flag sets by visiting the advanced manual [6].
"numgates"	Number of gates in this fault tree.	The user should define the number of gates in the related fault trees.

3.1.3.2. Gate List

Gate list sub-block is detailed in Figure 13 and Table 8.



```
"gatelist": [
  {
    "gateid": 1,
    "gatetype": "or",
    "numinputs": 2,
    "gateinput": [
      2,
      9
    ]
  },
```

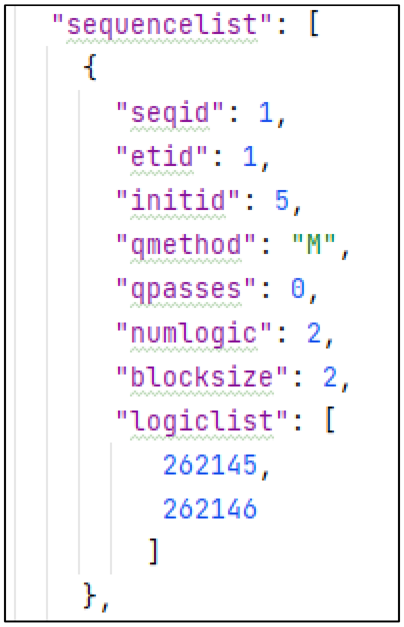
Figure 13. Example of gate list block. Each gate will have a sub-block under the gate list block. It should have at least one of the four input lists listed in the table.

Table 8. Gate list block explanations and instructions.

Argument	Explanation	Instructions
"gateid"	The gate unique integer ID.	The user should write the unique integer ID for the specified gate.
"gatetype"	The logical gate type (e.g., OR, AND, N/M)	The user should identify here the gate logical type (e.g., OR, AND, N/M).
"numinputs"	Number of inputs (basic events or gates) in the fault tree that correspond to the specified gate.	The user should write the number of inputs, which includes the number of gates and the number of basic events that corresponds to the specific gate.
"eventinput"	Basic events integer ID that corresponds to that gate.	The user should write all the basic events integer ID that corresponds to this gate.
“gateinput”	Gates integer ID that corresponds to that gate.	The user should write all the gate integer ID that corresponds to this gate.
"compeventinput”	Complemented basic events unique integer ID numbers.	The user should write all the complemented basic events integer ID that corresponds to this gate.
“compgateinput”	Complemented gates unique integer ID numbers.	The user should write all the complemented gates integer ID that corresponds to this gate.

3.1.4. Sequence List

Sequence list sub-block is detailed in Figure 14 and Table 9.



```
"sequencelist": [
  {
    "seqid": 1,
    "etid": 1,
    "initid": 5,
    "qmethod": "M",
    "qpasses": 0,
    "numlogic": 2,
    "blocksize": 2,
    "logiclist": [
       262145,
       262146
      ]
  },
```

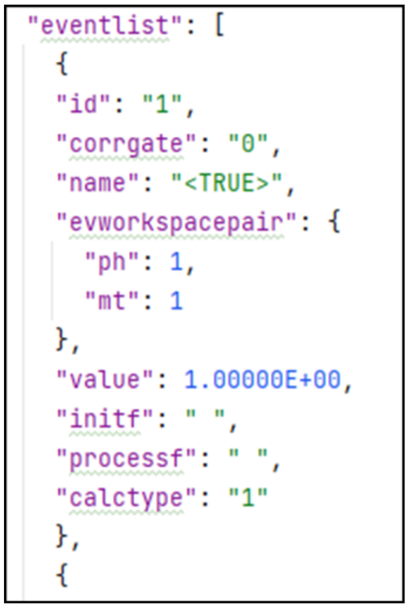
Figure 14. Example of sequence list block. Each sequence will have a sub-block under the sequence list block.

Table 9. Sequence list block explanations and instructions.

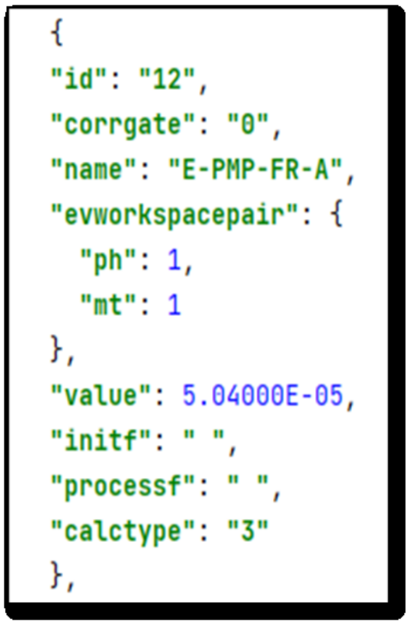
Argument	Explanation	Instructions
"seqid"	The sequence unique ID.	The user should write the unique integer ID for the specified sequence.
"etid"	Event tree number.	The user should write the unique number of the event tree that this sequence relates to.
"initid"	Initiating event number.	The user should write the unique number of the initiating event that this sequence relates to.
"qmethod"	Quantification method for quantifying the cut sets.	The user should identify the quantifications methods. There are five options: R is for rare event approximation. M is for minimum cut set upper bound (MCUB). B is for binary decision diagram (BDD). X is for min./max. S is for split fractions.
"qpasses"	If it is X (min./max.), the user must give it how many passes. If it is not used, then it is ignored. The user must use zero.	The user should identify how many passes there are if the quantification method used in min./max.
"numlogic"	How many fault trees in the logic list `logiclist`.	The user should identify how many fault trees exist in this sequence path.
"blocksize"	It should be equivalent to the `numlogic`.	The user should match the number in the `blocksize` with the number in `numlogic`.
"logiclist"	The list of successful and failed fault trees for this sequence.	The user should identify the failed and successful fault tree numbers.

3.1.5. Event List

The event list sub-block contains many sub-blocks for each basic event information. Table 10 and Figure 15 describe each line in the event list sub-block. In addition, every event tree list should contain all the basic events plus three additional house events that should be there no matter how many basic events the fault trees contain. These house events are <TRUE>, <FALSE>, <PASS>. In addition to those house events, we do have a basic event that represents each fault tree.



```
"eventlist": [
  {
  "id": "1",
  "corrgate": "0",
  "name": "<TRUE>",
  "evworkspacepair": {
    "ph": 1,
    "mt": 1
  },
  "value": 1.00000E+00,
  "initf": " ",
  "processf": " ",
  "calctype": "1"
  },
  {
```



```
{
"id": "12",
"corrgate": "0",
"name": "E-PMP-FR-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 5.04000E-05,
"initf": " ",
"processf": " ",
"calctype": "3"
},
```

Figure 15. Example of tree list block. Each event will have a sub-block under the event list block.

Table 10. Event tree list block explanations and instructions.

Argument	Explanation	Instructions
“id”	Event unique integer ID.	The user should write the unique integer ID for the specified event.
"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 events (i.e., <TRUE>, <FALSE>, <PASS>).	The user should write the name of the basic events and must include these events: <FALSE> <TRUE> <PASS> A basic event that represents each top-level fault tree. Initiating event(s).
"evworkspacepair"	Explained in Table 5.	Explained in Table 5.
"value"	Failure probability/frequency value for the basic event.	The user should write the probability/frequency of each event: <FALSE> = 0.0 <TRUE> = 1.0

Argument	Explanation	Instructions
		`<PASS>` = 1.0 All top-level fault trees = 1.0. It can be different based on the process flag. All initiating events = frequency of the initiating events.
"initf"	"I" stands for the initiating event. If it is not an initiating event, the field is blank.	The user should write an "I" if it's an initiating event. If it is not an initiating event, write "blank."
"processf"	Process flag for each basic events.	Most are blanks. Refer to the basics SAPHIRE manual for more explanation [5].
"calctype"	The failure model for each basic events.	The user should identify the failure model used for each basic event. There are many options, including these:

Failure Models	Parameters*
1	Mean Failure Probability
3	Lambda** (per hour), Mission Time*** (hours)
5	Lambda* *(per hour), Tau (hours), Mission Time*** (hours)
7	Lambda** (per hour), Tau (hours)
C	Library and Procedures
V	Value
N	Frequency, Frequency Units (initiating event)
J	Median Failure Acceleration, Fragilities, Screening G-Line
G	Median Failure Acceleration, Fragilities, Screening G-Line
H	Median Failure Acceleration, Fragilities
X	SPAR-H Diagnosis, Action, and Dependency Parameters
A,Q,R	CCF Properties Edited in model data****
E,S,I,T,F,U,Y,D,A,O	None

3.2. Output File Specification

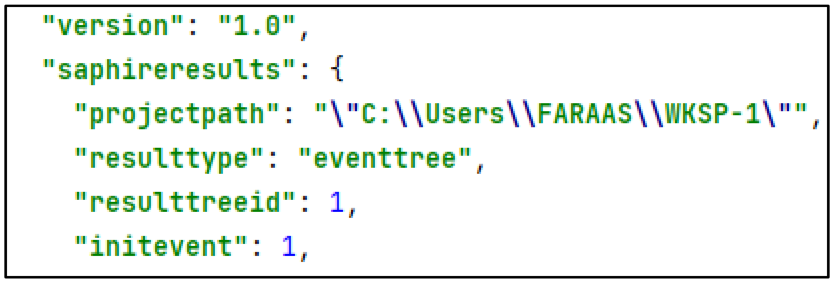
The remote solver produces cut sets based on the provided input file. These cut sets are returned in a JSON-based `JSCut` file. This file not only includes the cut sets, but it also encompasses details about their origin and the truncation parameters employed during their generation. Specifically, it offers information about the total count of cut sets and their collective failure value for each fault tree or sequence within an event tree.

Figure 16 illustrates the general composition of the SAPHSOLVE output file. The following sections explain the output file's structure. The subsequent output file example is derived from the model expounded upon in the preceding section's basics workshop [5]. For further reference, the output file is included in Appendix B. The remote solver generates cut sets for the input file passed into it. It sends those cut sets back in a JSON-based `JSCut` file. This file contains information about the source of the cut sets and the various truncation parameters used to generate the cut sets. It has the overall number of cut sets and the overall failure value of the cut sets for each fault tree or each sequence of an event tree.

Figure 16. General structure of the SAPHSOLVE output file.

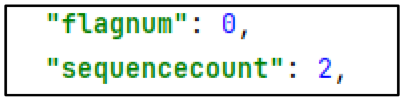
3.2.1. General Information

The SAPHSOLVE output file begins with a summary of solution details such as version number, project path, results, and additional information. Table 11, Figure 17, and Figure 18 describe each line of this information.



```
"version": "1.0",
"saphireresults": {
  "projectpath": "\"C:\\Users\\FARAAS\\WKSP-1\"",
  "resulttype": "eventtree",
  "resulttreeid": 1,
  "initevent": 1,
```

Figure 17. Example of general information.



```
"flagnum": 0,
"sequencecount": 2,
```

Figure 18. Example of additional information.

Table 11. General information explanations and instructions.

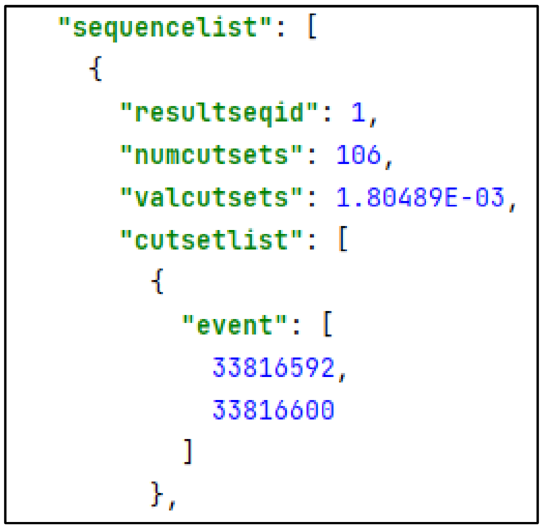
Argument	Explanation
“version”	Solver version number.
"projectpath"	The path where the model is located. This path is not related to the solution, but it is useful to identify where the solution is coming from.
“resulttype”	This explains whether the user is solving a fault tree or event tree. In the example in the figure, the user is solving the event tree.
“resulttreeid”	Event tree number.
“initevent”	The unique number of the initiating event.
Additional information	
Argument	Explanation
"flagnum"	The flag sets are used to indicate modifications to particular events on a sequence-by-sequence basis (or fault tree logic). Default is 0, which means it does not matter for the solution, but it is coming back for the solution. It is a definition of a state (i.e.,).. The user passes this information from the input.
"sequencecount"	The number of event tree sequences. This example has two: `SMALL-RELEASE` and `LARGE-RELEASE`.

3.2.2. Truncation Parameters and Workspace Pair

The truncation parameters and workspace pair blocks are information passed from the input to the output. The explanation and details can be found in Table 4 and Table 5. The truncation parameters block contains the truncation values for the event and fault trees and contains additional information. The workspace pair block includes the phase and model type.

3.2.3. Sequence List Results

The sequence list block contains the results of solving each sequence in the event tree, including the set and number of minimal cut sets and the failure probabilities. Table 12 and Figure 19 explain the sequence list block in detail.



```
"sequencelist": [
  {
    "resultseqid": 1,
    "numcutsets": 106,
    "valcutsets": 1.80489E-03,
    "cutsetlist": [
      {
        "event": [
          33816592,
          33816600
        ]
      },
```

Figure 19. Example of a sequence list block.

Table 12. Sequence list block explanations and instructions.

Argument	Explanation
"resultseqid"	Sequence number being solved.
"numcutsets"	Number of cut sets in this sequence.
"valcutsets"	Failure probability if the user is solving a fault tree. Frequency if the user is solving event tree.
"cutsetlist"	A list of all of the minimal cut sets.
"event"	Collection of events in a single cut set.

4. CONCLUSION AND FUTURE WORK

This work focuses on the extraction of the SAPHSOLVE engine from the SageRisk API, transitioning it into a standalone, remotely accessible event tree/fault tree solver. The primary objective of this endeavor is to create a comprehensive reference guide outlining the input and output files associated with SAPHSOLVE.

The ongoing project involves significant enhancements to SAPHSOLVE. Recently, comprehensive benchmarking and profiling analyses have been concluded. These findings will be expounded upon in a forthcoming report. The validation of SAPHSOLVE against quantification engines by the SAPHIRE team underscores its reliability.

The SAPHIRE team has outlined several areas of future development, including the following:

- Facilitate model exchange capabilities between SAPHSOLVE and other PRA tools.
- Leverage parallel and distributed computing to expedite the quantification process.
- Pioneer novel algorithms for precise and efficient quantification such as BDD [7] and zero-suppressed binary decision diagrams (ZBDD) [8].
- Transition from a 32-bit to a 64-bit framework to enhance computational capacity.
- Explore cloud-based solving options for heightened flexibility and scalability.
- Rigorously validate all discoveries through the utilization of SPAR models.

The future development plan aims to transition SAPHIRE from a legacy PRA tool to a state-of-art PRA tool.

5. REFERENCES

1. Smith, C. L., and S. T. Wood. 2011. "Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 8: Volume 1 Overview and Summary." INL/EXT-09-17009, Idaho National Laboratory, Idaho Falls, Idaho. https://doi.org/10.2172/1016191.
2. Russell K. D., M. B. Sattison, D. M. Rasmuson. 1990. "Living PRAs Made Easier with IRRAS." *Reliability Engineering & System Safety* 30(1–3): 239–269. https://doi.org/10.1016/0951-8320(90)90097-7.
3. Vedros, K., et al. 2021. "Enhancement of Industry Legacy Probabilistic Risk Assessment Methods and Tools." INL/EXT-21-64448-Rev000, Idaho National Laboratory, Idaho Falls, ID. https://doi.org/10.2172/1822882.
4. Miller, A., S. Hess, and C. Smith. 2020. "R&D Roadmap to Enhance Industry Legacy Probabilistic Risk Assessment Methods and Tools." INL/EXT-20-59202, Idaho National Laboratory, Idaho Falls, ID. https://lwrs.inl.gov/RiskInformed%20Safety%20Margin%20Characterization/PRA_Legacy_Tools_and%20Methods.pdf.
5. Smith, C., et al. 2016. "SAPHIRE 8 Basics: An Introduction to Probabilistic Risk Assessment via the Systems Analysis Program for Hands-On Integrated Reliability Evaluations (SAPHIRE) Software." Accessed August 27, 2021. https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_1039.pdf.
6. Smith, C., et al. 2022. "SAPHIRE 8 Advanced." Idaho National Laboratory, Idaho Falls, ID.
7. Akers, S. B. 1978. "Binary Decision Diagrams." *IEEE Transactions on Computers* 27(6): 509–516. https://doi.org/10.1109/TC.1978.1675141.
8. Jung, W. S. 2009. "ZBDD Algorithm Features for an Efficient Probabilistic Safety Assessment." *Nuclear Engineering and Design* 239(10): 2085–2092. https://doi.org/10.1016/j.nucengdes.2009.05.005.

Appendix A

Input Demo Model

```
{
 "version": "1.0",
 "saphiresolveinput": {
  "header": {
   "projectpath": "\"E:\\Development\\Training\\Models\\WKSP-2\"",
   "eventtree": {
    "name": "LOSP",
    "number": 1,
    "initevent": 5,
    "seqphase": 1
   },
   "flagnum": 0,
   "ftcount": 2,
   "fthigh": 2,
   "sqcount": 2,
   "sqhigh": 2,
   "becount": 31,
   "behigh": 32,
   "mthigh": 1,
   "phhigh": 1,
   "truncparam": {
    "ettruncopt": "NormalProbCutOff",
    "fttruncopt": "GlobalProbCutOff",
    "sizeopt": "ENoTrunc",
    "ettruncval": 1.000E-12,
    "fttruncval": 1.000E-12,
    "sizeval": 99,
    "transrepl": false,
    "transzones": false,
    "translevel": 0,
    "usedual": false,
    "dualcutoff": 0.000E+00
   },
   "workspacepair": {
    "ph": 1,
    "mt": 1
   },
   "iworkspacepair": {
    "ph": 1,
    "mt": 1
   }
  },
  "sysgatelist": [
   {
    "name": "ECS",
    "id": 1,
    "gateid": 1,
    "gateorig": 1,
    "gatepos": 0,
    "eventid": 6,
    "gatecomp": 1,
    "comppos": 0,
    "compflag": " ",
    "gateflag": " ",
    "gatet": " ",
    "bddsuccess": false,
    "done": false
   },
   {
    "name": "CCS",
    "id": 2,
    "gateid": 11,
    "gateorig": 11,
    "gatepos": 0,
    "eventid": 7,
    "gatecomp": 11,
    "comppos": 0,
    "compflag": " ",
    "gateflag": " ",
    "gatet": " ",
    "bddsuccess": false,
    "done": false
   }
   ],
 "faulttreelist": [
 {
  "ftheader": {
    "ftid": 1,
    "gtid": 1,
    "evid": 6,
    "defflag": 0,
    "numgates": 10
   },
  "gatelist": [
    {
     "gateid": 1,
     "gatetype": "or",
     "numinputs": 2,
     "gateinput": [
       2,
       9
     ]
   },
    {
     "gateid": 2,
     "gatetype": "and",
     "numinputs": 2,
     "gateinput": [
       3,
       6
    ]
  },
    {
     "gateid": 3,
     "gatetype": "or",
     "numinputs": 4,
     "gateinput": [
       4,
       5
     ],
     "eventinput": [
       12,
       13
     ]
  },
    {
     "gateid": 4,
     "gatetype": "or",
     "numinputs": 2,
     "eventinput": [
       10,
       11
     ]
  },
    {
     "gateid": 5,
     "gatetype": "or",
     "numinputs": 2,
     "eventinput": [
       20,
       21
     ]
  },
    {
     "gateid": 6,
     "gatetype": "or",
     "numinputs": 4,
     "gateinput": [
       7,
       8
     ],
     "eventinput": [
       16,
       17
     ]
  },
    {
     "gateid": 7,
     "gatetype": "or",
     "numinputs": 2,
     "eventinput": [
       14,
       15
    ]
  },
    {
     "gateid": 8,
     "gatetype": "or",
     "numinputs": 2,
     "eventinput": [
       22,
       23
     ]
  },
    {
     "gateid": 9,
     "gatetype": "or",
     "numinputs": 2,
     "gateinput": [
       10
     ],
     "eventinput": [
       8
     ]
  },
    {
     "gateid": 10,
     "gatetype": "or",
     "numinputs": 2,
     "gateinput": [
       5
     ],
     "eventinput": [
       9
     ]
  }
  ]
    },
{
 "ftheader": {
   "ftid": 2,
   "gtid": 11,
   "evid": 7,
   "defflag": 0,
   "numgates": 10
  },
  "gatelist": [
    {
     "gateid": 5,
     "gatetype": "or",
     "numinputs": 2,
     "eventinput": [
       20,
       21
     ]
  },
  {
   "gateid": 8,
   "gatetype": "or",
   "numinputs": 2,
   "eventinput": [
     22,
     23
   ]
 },
  {
   "gateid": 11,
   "gatetype": "or",
   "numinputs": 2,
   "gateinput": [
     12,
     17
   ]
 },
  {
   "gateid": 12,
   "gatetype": "and",
   "numinputs": 2,
   "gateinput": [
     13,
     15
   ]
 },
  {
   "gateid": 13,
   "gatetype": "or",
   "numinputs": 4,
   "gateinput": [
     5,
     14
   ],
   "eventinput": [
     29,
     30
   ]
 },
  {
   "gateid": 14,
   "gatetype": "or",
   "numinputs": 2,
   "eventinput": [
     31,
     32
   ]
 },
  {
   "gateid": 15,
   "gatetype": "or",
   "numinputs": 4,
            "gateinput": [
              8,
              16
            ],
            "eventinput": [
              25,
              26
            ]
          },
          {
            "gateid": 16,
            "gatetype": "or",
            "numinputs": 2,
            "eventinput": [
              27,
              28
            ]
          },
          {
            "gateid": 17,
            "gatetype": "or",
            "numinputs": 2,
            "gateinput": [
              18
            ],
            "eventinput": [
              8
            ]
          },
          {
            "gateid": 18,
            "gatetype": "or",
            "numinputs": 2,
            "gateinput": [
              8
            ],
            "eventinput": [
              24
            ]
          }
        ]
      }
    ],
    "sequencelist": [
      {
        "seqid": 1,
        "etid": 1,
        "initid": 5,
        "qmethod": "M",
        "qpasses": 0,
        "numlogic": 2,
        "blocksize": 2,
        "logiclist": [
        262145,
        262146
       ]
    },
    {
      "seqid": 2,
      "etid": 1,
      "initid": 5,
      "qmethod": "M",
      "qpasses": 0,
      "numlogic": 2,
      "blocksize": 2,
      "logiclist": [
        262145,
        2147745794
       ]
    }
  ],
 "eventlist": [
  {
  "id": "1",
  "corrgate": "0",
  "name": "<TRUE>",
  "evworkspacepair": {
    "ph": 1,
    "mt": 1
  },
  "value": 1.00000E+00,
  "initf": " ",
  "processf": " ",
  "calctype": "1"
  },
  {
  "id": "2",
  "corrgate": "0",
  "name": "<FALSE>",
  "evworkspacepair": {
    "ph": 1,
    "mt": 1
  },
  "value": 0.00000E+00,
  "initf": " ",
  "processf": " ",
  "calctype": "1"
  },
  {
  "id": "3",
  "corrgate": "0",
  "name": "<PASS>",
  "evworkspacepair": {
    "ph": 1,
    "mt": 1
  },
"value": 1.00000E+00,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "5",
"corrgate": "0",
"name": "LOSP",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.30000E+00,
"initf": "I",
"processf": " ",
"calctype": "N"
},
{
"id": "6",
"corrgate": "1",
"name": "ECS",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E+00,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "8",
"corrgate": "0",
"name": "S-TNK-FC-T1",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.40000E-06,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "9",
"corrgate": "0",
"name": "E-MOV-CC-1",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.00000E-04,
```

"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "10",
"corrgate": "0",
"name": "E-PMP-FR-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 5.03900E-04,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "11",
"corrgate": "0",
"name": "E-PMP-FS-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.20000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "12",
"corrgate": "0",
"name": "E-CKV-CC-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E-04,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "13",
"corrgate": "0",
"name": "E-MOV-CC-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E-03,
"initf": " ",

```
"processf": " ",
"calctype": "1"
},
{
"id": "14",
"corrgate": "0",
"name": "E-PMP-FR-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 5.03900E-04,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "15",
"corrgate": "0",
"name": "E-PMP-FS-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.20000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "16",
"corrgate": "0",
"name": "E-CKV-CC-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E-04,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "17",
"corrgate": "0",
"name": "E-MOV-CC-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "18",
"corrgate": "5",
"name": "SUP-DGN-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E+00,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "19",
"corrgate": "8",
"name": "SUP-DGN-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E+00,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "20",
"corrgate": "0",
"name": "S-DGN-FR-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.11300E-02,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "21",
"corrgate": "0",
"name": "S-DGN-FS-A",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 4.00000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
```

```
},
{
"id": "22",
"corrgate": "0",
"name": "S-DGN-FR-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.11300E-02,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "23",
"corrgate": "0",
"name": "S-DGN-FS-B",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 4.00000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "7",
"corrgate": "11",
"name": "CCS",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 1.00000E+00,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "24",
"corrgate": "0",
"name": "C-MOV-CC-1",
"evworkspacepair": {
  "ph": 1,
  "mt": 1
},
"value": 2.00000E-04,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "25",
"corrgate": "0",
"name": "C-CKV-CC-B",
"evworkspacepair": {
 "ph": 1,
 "mt": 1
},
"value": 1.00000E-04,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "26",
"corrgate": "0",
"name": "C-MOV-CC-B",
"evworkspacepair": {
 "ph": 1,
 "mt": 1
},
"value": 1.00000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
"id": "27",
"corrgate": "0",
"name": "C-PMP-FR-B",
"evworkspacepair": {
 "ph": 1,
 "mt": 1
},
"value": 5.03900E-04,
"initf": " ",
"processf": " ",
"calctype": "3"
},
{
"id": "28",
"corrgate": "0",
"name": "C-PMP-FS-B",
"evworkspacepair": {
 "ph": 1,
 "mt": 1
},
"value": 1.20000E-03,
"initf": " ",
"processf": " ",
"calctype": "1"
},
{
    "id": "29",
    "corrgate": "0",
    "name": "C-CKV-CC-A",
    "evworkspacepair": {
      "ph": 1,
      "mt": 1
    },
    "value": 1.00000E-04,
    "initf": " ",
    "processf": " ",
    "calctype": "1"
    },
    {
    "id": "30",
    "corrgate": "0",
    "name": "C-MOV-CC-A",
    "evworkspacepair": {
      "ph": 1,
      "mt": 1
    },
    "value": 1.00000E-03,
    "initf": " ",
    "processf": " ",
    "calctype": "1"
    },
    {
    "id": "31",
    "corrgate": "0",
    "name": "C-PMP-FR-A",
    "evworkspacepair": {
      "ph": 1,
      "mt": 1
    },
    "value": 5.03900E-04,
    "initf": " ",
    "processf": " ",
    "calctype": "3"
    },
    {
    "id": "32",
    "corrgate": "0",
    "name": "C-PMP-FS-A",
    "evworkspacepair": {
      "ph": 1,
      "mt": 1
    },
    "value": 1.20000E-03,
    "initf": " ",
    "processf": " ",
    "calctype": "1"
    }
  ]
}
```

```
}
```

Page intentionally left blank

Appendix B

Output Demo Model

```
{
 "version": "1.0",
 "saphireresults": {
  "projectpath": "\"E:\\Development\\Training\\Models\\WKSP-2\"",
  "resulttype": "eventtree",
  "resulttreeid": 1,
  "initevent": 1,
  "truncparam": {
   "ettruncopt": "NormalProbCutOff",
   "fttruncopt": "GlobalProbCutOff",
   "sizeopt": "ENoTrunc",
   "ettruncval": 1.000E-12,
   "fttruncval": 1.000E-12,
   "sizeval": 99,
   "transrepl": false,
   "transzones": false,
   "translevel": "99",
   "usedual": false,
   "dualcutoff": 0.000E+00
  },
  "workspacepair": [
   {
   "ph": 1,
   "mt": 1
   }
  ],
  "flagnum": 0,
  "sequencecount": 2,
  "sequencelist": [
          {
   "resultseqid": 1,
   "numcutsets": 106,
   "valcutsets": 1.80489E-03,
   "cutsetlist": [
      {
       "event": [
        33816585,
        33816600
       ]
      },
      {
       "event": [
        33816584
       ]
      },
      {
       "event": [
    33816586,
    33816590,
    33816600
  ]
},
{
  "event": [
    33816586,
    33816591,
    33816600
  ]
},
{
  "event": [
    33816586,
    33816592,
    33816600
  ]
},
{
  "event": [
    33816586,
    33816593,
    33816600
  ]
},
{
  "event": [
    33816587,
    33816590,
    33816600
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816600
  ]
},
{
  "event": [
    33816587,
    33816592,
    33816600
  ]
},
{
  "event": [
    33816587,
    33816593,
    33816600
  ]
```

```
},
{
 "event": [
   33816588,
   33816590,
   33816600
 ]
},
{
 "event": [
   33816588,
   33816591,
   33816600
 ]
},
{
 "event": [
   33816588,
   33816592,
   33816600
 ]
},
{
 "event": [
   33816588,
   33816593,
   33816600
 ]
},
{
 "event": [
   33816589,
   33816590,
   33816600
 ]
},
{
 "event": [
   33816589,
   33816591,
   33816600
 ]
},
{
 "event": [
   33816589,
   33816592,
   33816600
 ]
},
{
 "event": [
   33816589,
    33816593,
    33816600
  ]
},
{
  "event": [
    33816596,
    33816600
  ]
},
{
  "event": [
    33816597,
    33816600
  ]
},
{
  "event": [
    33816586,
    33816598
  ]
},
{
  "event": [
    33816586,
    33816599
  ]
},
{
  "event": [
    33816587,
    33816598
  ]
},
{
  "event": [
    33816587,
    33816599
  ]
},
{
  "event": [
    33816588,
    33816598
  ]
},
{
  "event": [
    33816588,
    33816599
  ]
},
{
  "event": [
    33816589,
    33816598
  ]
},
{
  "event": [
    33816589,
    33816599
  ]
},
{
  "event": [
    33816585,
    33816598
  ]
},
{
  "event": [
    33816585,
    33816599
  ]
},
{
  "event": [
    33816596,
    33816598
  ]
},
{
  "event": [
    33816596,
    33816599
  ]
},
{
  "event": [
    33816597,
    33816598
  ]
},
{
  "event": [
    33816597,
    33816599
  ]
},
{
  "event": [
    33816596,
    33816601
  ]
},
{
 "event": [
   33816596,
   33816602
 ]
},
{
 "event": [
   33816596,
   33816603
 ]
},
{
 "event": [
   33816596,
   33816604
 ]
},
{
 "event": [
   33816597,
   33816601
 ]
},
{
 "event": [
   33816597,
   33816602
 ]
},
{
 "event": [
   33816597,
   33816603
 ]
},
{
 "event": [
   33816597,
   33816604
 ]
},
{
 "event": [
   33816586,
   33816591,
   33816602,
   33816606
 ]
},
{
 "event": [
   33816586,
    33816591,
    33816602,
    33816608
  ]
},
{
  "event": [
    33816586,
    33816591,
    33816604,
    33816606
  ]
},
{
  "event": [
    33816586,
    33816591,
    33816604,
    33816608
  ]
},
{
  "event": [
    33816586,
    33816593,
    33816602,
    33816606
  ]
},
{
  "event": [
    33816586,
    33816593,
    33816602,
    33816608
  ]
},
{
  "event": [
    33816586,
    33816593,
    33816604,
    33816606
  ]
},
{
  "event": [
    33816586,
    33816593,
    33816604,
    33816608
  ]
},
{
 "event": [
   33816587,
   33816590,
   33816602,
   33816606
 ]
},
{
 "event": [
   33816587,
   33816590,
   33816602,
   33816608
 ]
},
{
 "event": [
   33816587,
   33816590,
   33816604,
   33816606
 ]
},
{
 "event": [
   33816587,
   33816590,
   33816604,
   33816608
 ]
},
{
 "event": [
   33816587,
   33816591,
   33816602,
   33816606
 ]
},
{
 "event": [
   33816587,
   33816591,
   33816602,
   33816607
 ]
},
{
 "event": [
   33816587,
   33816591,
   33816602,
    33816608
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816603,
    33816606
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816603,
    33816608
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816604,
    33816606
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816604,
    33816607
  ]
},
{
  "event": [
    33816587,
    33816591,
    33816604,
    33816608
  ]
},
{
  "event": [
    33816587,
    33816593,
    33816602,
    33816606
  ]
},
{
  "event": [
      33816587,
      33816593,
      33816602,
      33816607
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816602,
      33816608
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816603,
      33816606
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816603,
      33816608
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816604,
      33816606
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816604,
      33816607
    ]
  },
  {
    "event": [
      33816587,
      33816593,
      33816604,
      33816608
    ]
```

```
},
{
 "event": [
   33816589,
   33816590,
   33816602,
   33816606
 ]
},
{
 "event": [
   33816589,
   33816590,
   33816602,
   33816608
 ]
},
{
 "event": [
   33816589,
   33816590,
   33816604,
   33816606
 ]
},
{
 "event": [
   33816589,
   33816590,
   33816604,
   33816608
 ]
},
{
 "event": [
   33816589,
   33816591,
   33816602,
   33816606
 ]
},
{
 "event": [
   33816589,
   33816591,
   33816602,
   33816607
 ]
},
{
 "event": [
   33816589,
   33816591,
    33816602,
    33816608
  ]
},
{
  "event": [
    33816589,
    33816591,
    33816603,
    33816606
  ]
},
{
  "event": [
    33816589,
    33816591,
    33816603,
    33816608
  ]
},
{
  "event": [
    33816589,
    33816591,
    33816604,
    33816606
  ]
},
{
  "event": [
    33816589,
    33816591,
    33816604,
    33816607
  ]
},
{
  "event": [
    33816589,
    33816591,
    33816604,
    33816608
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816602,
    33816606
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816602,
    33816607
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816602,
    33816608
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816603,
    33816606
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816603,
    33816608
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816604,
    33816606
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816604,
    33816607
  ]
},
{
  "event": [
    33816589,
    33816593,
    33816604,
    33816608
  ]
},
{
  "event": [
    33816585,
    33816601,
    33816605
  ]
},
{
  "event": [
    33816585,
    33816601,
    33816606
  ]
},
{
  "event": [
    33816585,
    33816601,
    33816607
  ]
},
{
  "event": [
    33816585,
    33816601,
    33816608
  ]
},
{
  "event": [
    33816585,
    33816602,
    33816605
  ]
},
{
  "event": [
    33816585,
    33816602,
    33816606
  ]
},
{
  "event": [
    33816585,
    33816602,
    33816607
  ]
},
{
  "event": [
    33816585,
    33816602,
    33816608
  ]
 },
 {
  "event": [
    33816585,
    33816603,
    33816605
  ]
 },
 {
  "event": [
    33816585,
    33816603,
    33816606
  ]
 },
 {
  "event": [
    33816585,
    33816603,
    33816607
  ]
 },
 {
  "event": [
    33816585,
    33816603,
    33816608
  ]
 },
 {
  "event": [
    33816585,
    33816604,
    33816605
  ]
 },
 {
  "event": [
    33816585,
    33816604,
    33816606
  ]
 },
 {
  "event": [
    33816585,
    33816604,
    33816607
  ]
```

```
      },
      {
        "event": [
          33816585,
          33816604,
          33816608
        ]
      }
    ]
  },
          {
    "resultseqid": 2,
    "numcutsets": 19,
    "valcutsets": 5.80707E-02,
    "cutsetlist": [
      {
        "event": [
          33816585
        ]
      },
      {
        "event": [
          33816586,
          33816590
        ]
      },
      {
        "event": [
          33816586,
          33816591
        ]
      },
      {
        "event": [
          33816586,
          33816592
        ]
      },
      {
        "event": [
          33816586,
          33816593
        ]
      },
      {
        "event": [
          33816587,
          33816590
        ]
      },
      {
        "event": [
          33816587,
      33816591
    ]
  },
  {
    "event": [
      33816587,
      33816592
    ]
  },
  {
    "event": [
      33816587,
      33816593
    ]
  },
  {
    "event": [
      33816588,
      33816590
    ]
  },
  {
    "event": [
      33816588,
      33816591
    ]
  },
  {
    "event": [
      33816588,
      33816592
    ]
  },
  {
    "event": [
      33816588,
      33816593
    ]
  },
  {
    "event": [
      33816589,
      33816590
    ]
  },
  {
    "event": [
      33816589,
      33816591
    ]
  },
  {
    "event": [
            33816589,
            33816592
          ]
        },
        {
          "event": [
            33816589,
            33816593
          ]
        },
        {
          "event": [
            33816596
          ]
        },
        {
          "event": [
            33816597
          ]
        }
      ]
  }
]
}
}
```