Advanced SAPHIRE

Modeling Methods for Probabilistic Risk Assessment via the Systems Analysis Program for Hands-On Integrated Reliability Evaluations (SAPHIRE) Software

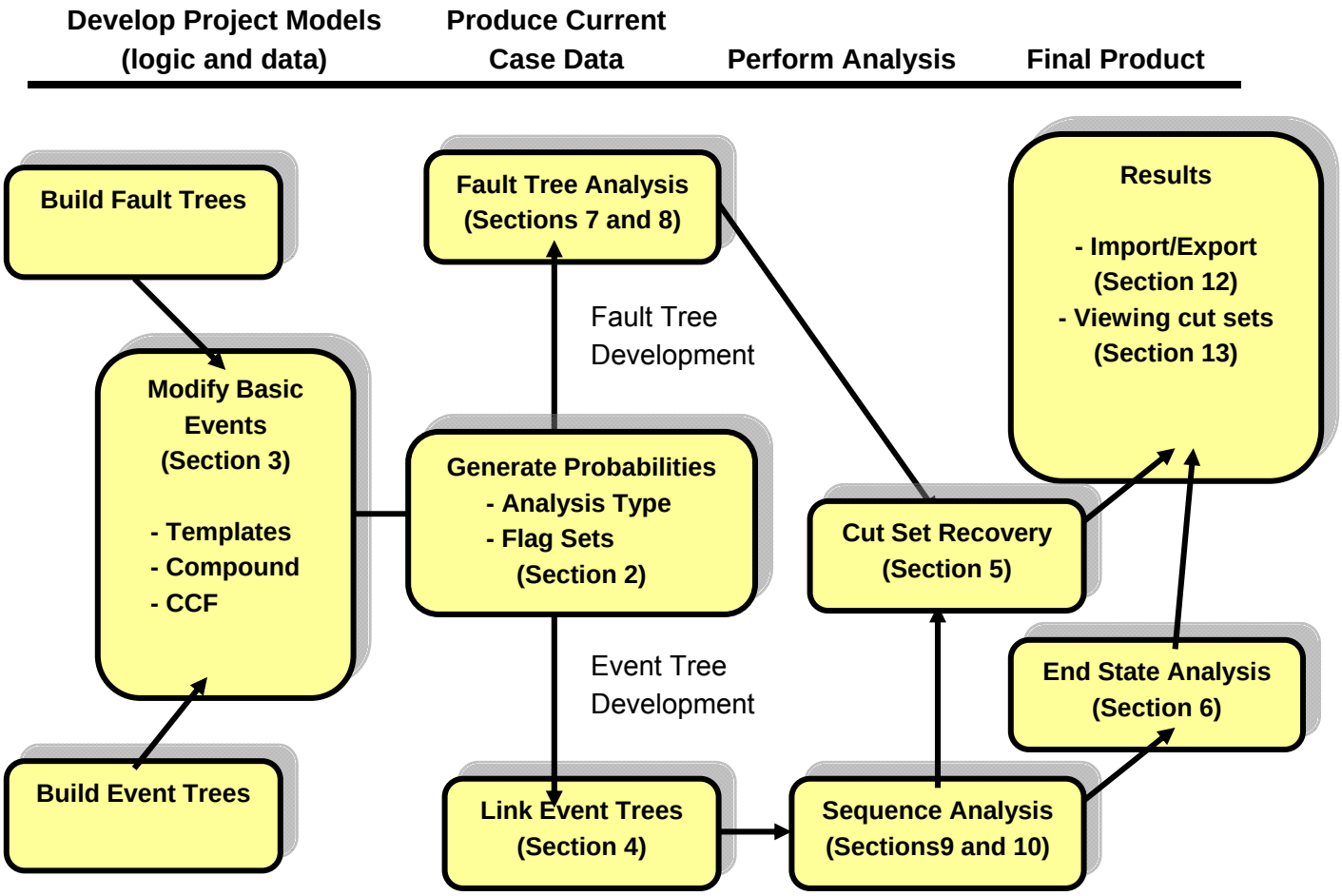
Curtis Smith
James Knudsen
Ted Wood

Idaho National Laboratory

February 2009

Idaho National Laboratory

SAPHIRE – The "Big Picture"

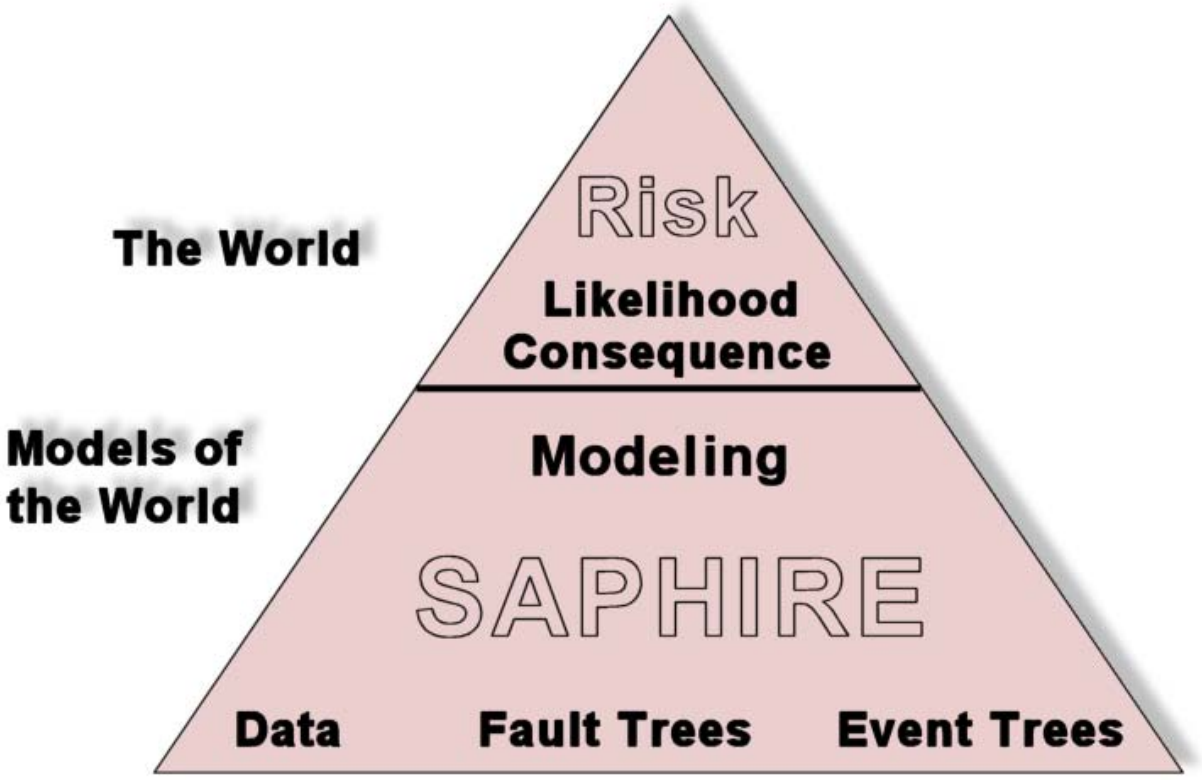


CONTENTS

This page intentionally left blank

1 INTRODUCTION

Section 1 contains an introduction to the SAPHIRE Advanced course material and an overview of the SAPHIRE software.



1.1. Overview of the Advanced SAPHIRE Material

The Advanced SAPHIRE course material is intended to both (1) provide guidance for learning advanced SAPHIRE features during the class and (2) become a stand-alone reference document after finishing the class. Thus, the format for the class material is a combination of the traditional "overhead-type" of presentation information with a structured, reference-type document.

Major topics that are covered in the Advanced SAPHIRE class include:

- Advanced data base concepts such as Analysis Type (e.g., random, seismic, fire, flood).
- Definition of house events and how they are used on a sequence-by-sequence basis to manipulate individual fault trees using sequence flag sets or fault tree flag sets.
- Basic event templates, compound basic event equation editors, and human error worksheets.
- A rule-based event tree top event substitution feature (called the Link Event Tree Rule Editor) which allows for top event substitutions.
- A rule-based cut set post-processing feature (called the Recovery Rules) which allows cut set manipulation.
- Cut set analysis based upon rule-based end state categories (called End State Analysis).
- Cut set generation options for both fault trees and event tree sequences.
- The large event tree methodology and how SAPHIRE can be used to generate sequence cut sets for this method.
- The use of the MAR-D module for the transfer of data between SAPHIRE data bases and between other PRA codes.

SAPHIRE screen displays will be shown as they appear on your video display (as shown below).

When discussing a particular sequence of menu options, the nomenclature

MAIN Menu ➔ Submenu Option

will be used to indicate the main SAPHIRE menu option and any successive submenu options (only the tool buttons will be discussed as the means for maneuvering through SAPHIRE).

1.2. SAPHIRE - What Is It and What Can It Do?

- SAPHIRE is an integrated PRA software tool that gives the user the ability to create and analyze fault trees and event trees using a personal computer.

- IRRAS was originally released in 1987 (version 1.0). Other versions of IRRAS include 2.0, 2.5, and 4.0. Additions and improvements have been added to each version of the code.

- Creation of 32-bit IRRAS, version 5.0, in 1992 resulted in an order of magnitude decrease in analysis time. New features included: individual code modules combined into a single module; end state analysis; fire, flood, and seismic modules; rule-based cut set processing; and rule-based fault tree to event tree linking.

- SAPHIRE for Windows, version 6.0, is released in 1997. Use of a Windows user interface makes SAPHIRE easier to learn and use.

- SAPHIRE for Windows, version 7.0, is released in 1999. This is the latest version of the SAPHIRE code. This manual is written for version 7.x of the software.

- SAPHIRE contains several features:

    - PC-based fault tree and event tree graphical and text editors
    - Cut set generation and quantification
    - Importance measures and uncertainty modules
    - Relational database with cross-referencing features
    - External events analysis (e.g., seismic, location transformation)
    - Rule-based recovery and end-state analysis
    - Common cause failure (CCF) basic event capabilities

- SAPHIRE minimal hardware requirements:

    - Windows 98 or later.
    - Pentium class IBM-PC compatible with 2-button mouse.
    - 50 MB free disk space (minimum for installation).

1.3. The Class Workbook

- The workshop problems for the SAPHIRE class are contained in a separate handout, referred to as the “workbook” or "workshop manual."

- The workbook allows the Advanced SAPHIRE class to be tailored to specific audiences. This “tailored-problem” format gives the freedom to present specific topics or problems centered around the expected needs of the students.

- The workbook follows the same format as the course material, and together provides an integrated reference package for the SAPHIRE code.

1.4. Installation of SAPHIRE

- To install:

  - Input the media containing the SAPHIRE executable (i.e., memory stick).
  - Double click on the SAPHIRE executable (SAPHIRE-7-27.exe).
  - Follow the installation program instructions.

- The installation program will make a subdirectory on your hard drive to store SAPHIRE.

  - Databases (such as the DEMO database) can be contained in any subdirectory that is chosen (e.g., C:\DEMO or C:\SAPHIRE7\DEMO).
  - The database subdirectory will contain the relational database files.
    - *.IDX files contain data indices.
    - *.BLK files contain variable length data (e.g., cut sets).
    - *.DAT files contain actual data and data pointers.

1.5. Lists and Masks

List Boxes

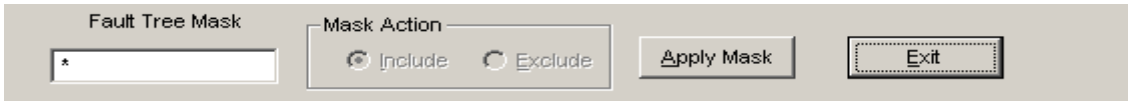
- Many dialogs in SAPHIRE contain list boxes. In some list boxes, multiple items can be selected for processing. An item in a list is selected if it is highlighted. There are various ways to select items from a list.

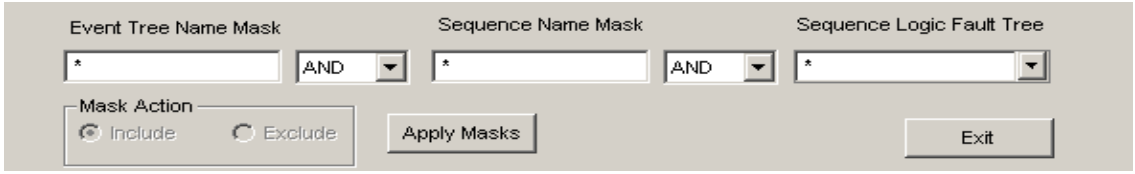
  - ◊ To select a single item, click with the left mouse button on the desired item and let go of the mouse button.
  - ◊ To select multiple continuous list items, you can click with the left mouse button the on first desired item and drag up or down the list to the last desired item and then let go of the mouse button. Alternately, click the first desired item then, holding down the ***Shift*** key, click the last desired item.
  - ◊ To select multiple non-continuous items in the list, click several items while holding down the ***Control*** key.

Masks

- Some dialogs with list boxes provide a "Mask" capability which allows the user to select items from the list based on matched criteria. Generally, the mask is applied to the name of the item (e.g. Fault Tree name or Event Tree name). To use the mask capability,

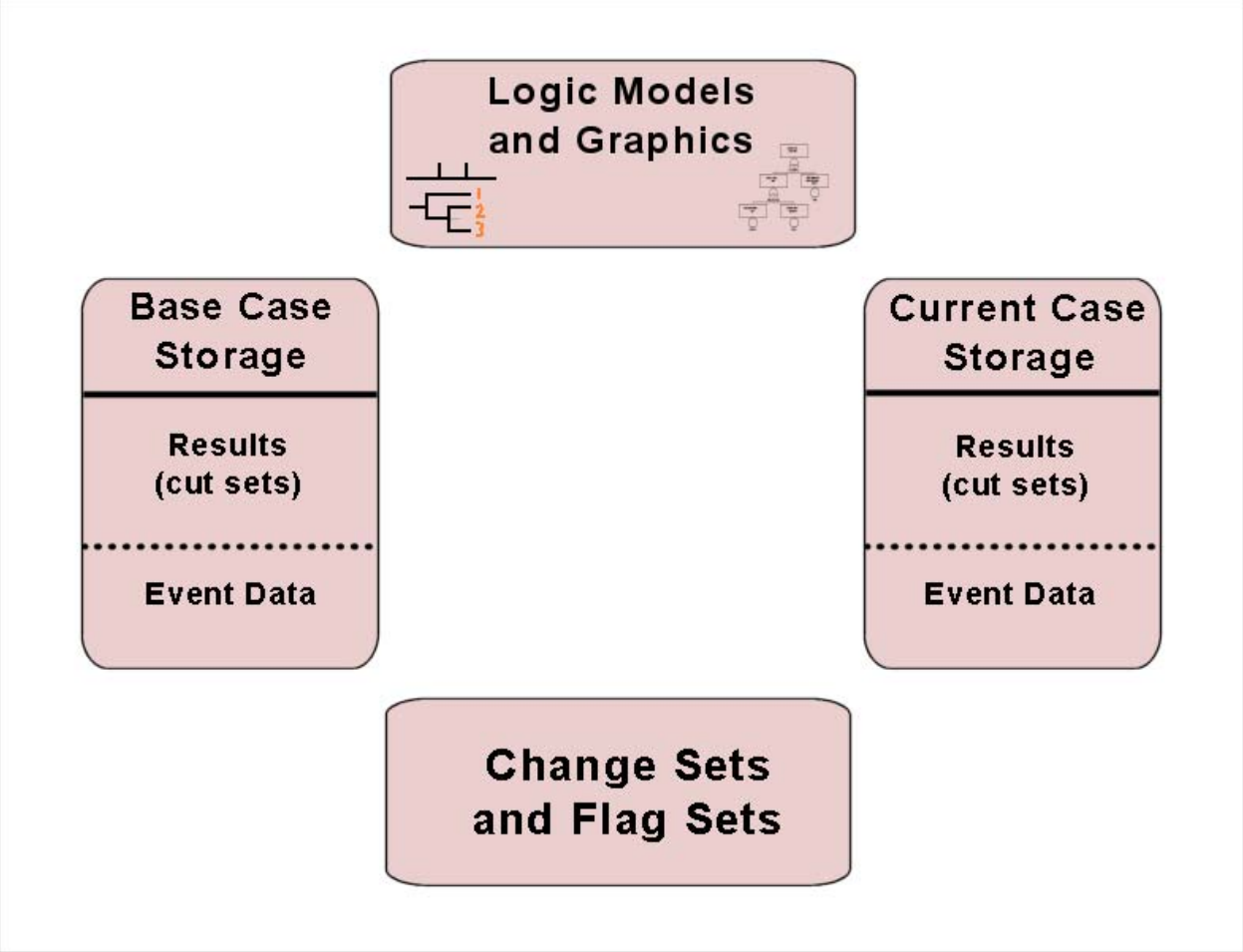
  1. In the mask entry field type the common characters of the names you wish to match. The wildcard characters, asterisk (*) and question mark (?), can be used in the mask. The asterisk represents one or more characters that a group has in common. The question mark represents a single character in that position of the string that a group has in common.
  2. Click either the Include or Exclude radio button, depending on whether you want these items included in the selection or excluded from it.
  3. Choose the Apply Mask button. All list items with names matching the criteria will be selected or deselected depending upon the option.





2 DATABASE CONCEPTS

Section 2 presents an overview of the SAPHIRE database structure. Included in this section are discussions of SAPHIRE projects, base case versus current case, and base case updates. Advanced data base features that are discussed include: (1) analysis types and (2) flag sets and change sets.

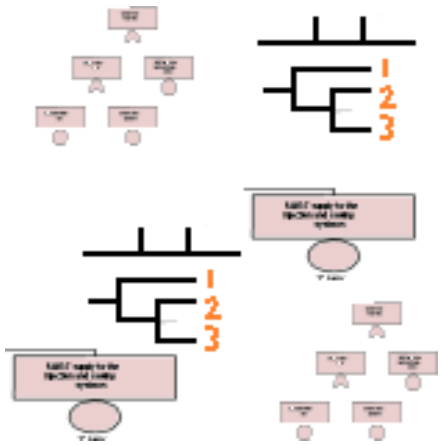


2.1. SAPHIRE Projects

- In SAPHIRE, the term “project” represents a single, specific database.

Project (Definition)

A group of fault tree logic and graphics; event trees and sequences; basic events and related data; cut sets; analysis results; and descriptions.



= SAPHIRE Project

- To select an existing project.

1. Use the **Open** icon or use the **File** → **Open Project** option.
2. An “Open Project” window will appear. Use the various Window Explorer options to find the folder containing the existing project.
3. Select and open either the *.SRA (SAPHIRE Risk Assessment) or FAM.DAT file in the opened folder.

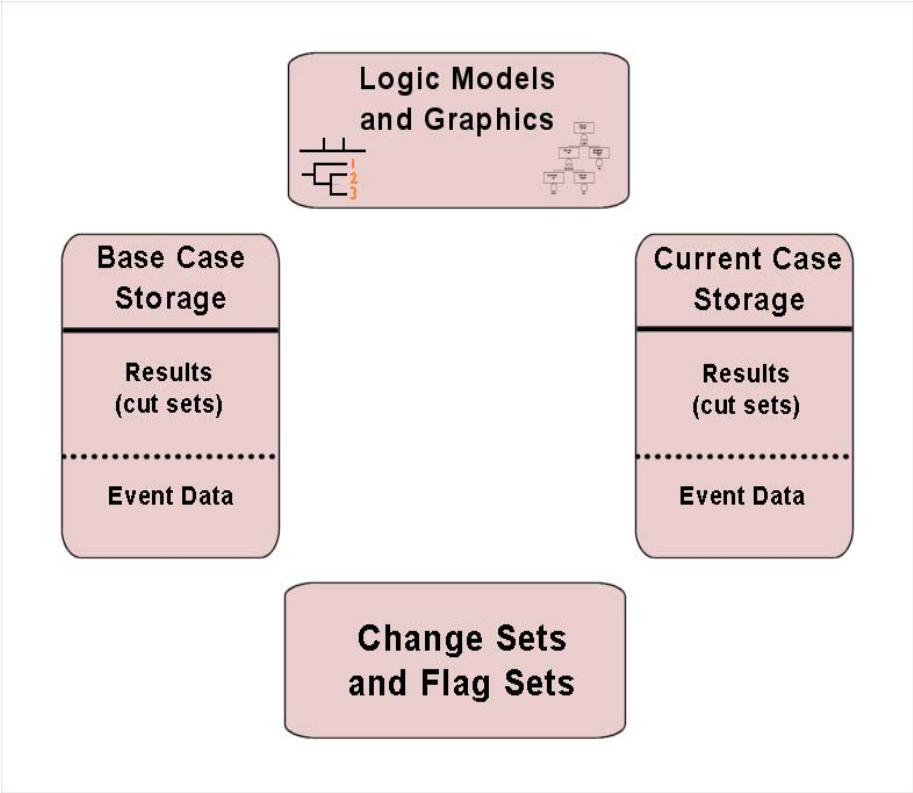
- To make a new project from within SAPHIRE:

1. Use the **New** icon or use the **File** → **New Project** option.
2. Indicate “Yes” when asked to create a new project.
3. Type the project name. Click **Ok**.
4. If the path\name is adequate, click **Ok**. The new project is then created and selected.

SAPHIRE will automatically open the last project used when started the next time. In addition, a drop-down listing of the last several data bases used is available under the **File** menu option.

2.2. Base Case Versus Current Case Data

- Base case and current case are two separate parts of a project database.
  - ◊ **Base Case** data is stored in the data base files as a “permanent” record
  - ◊ **Current Case** data is used to perform an analysis (e.g., cut set generation and quantification)



Current Case Is

- ◊ Created (via the **Generate** option) by applying change sets to base case data
- ◊ Used for sensitivity or event analysis

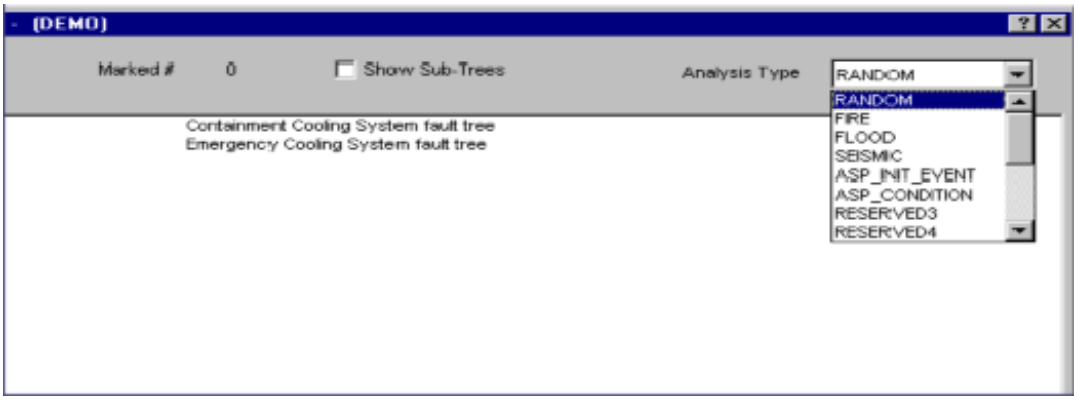
- All SAPHIRE calculations use the data stored in the **current** case.

2.3. Using the Analysis Type Option

- Versions previous to IRRAS 5.0 only had the capability to handle random analysis (i.e., Level 1, internal events PRA).

- SAPHIRE 7.0 is able to handle random (Level 1, internal events PRA) and other types of analyses, including:

    - ◊ SEISMIC (external events related to earthquakes or other ground disturbances).
    - ◊ FIRE (external events related to fires).
    - ◊ FLOOD (external events related to flooding).
    - ◊ ASP_INIT_EVENT (Accident sequence precursor related to initiating event assessments).
    - ◊ ASP_CONDITION (Accident sequence precursor related to condition assessments).



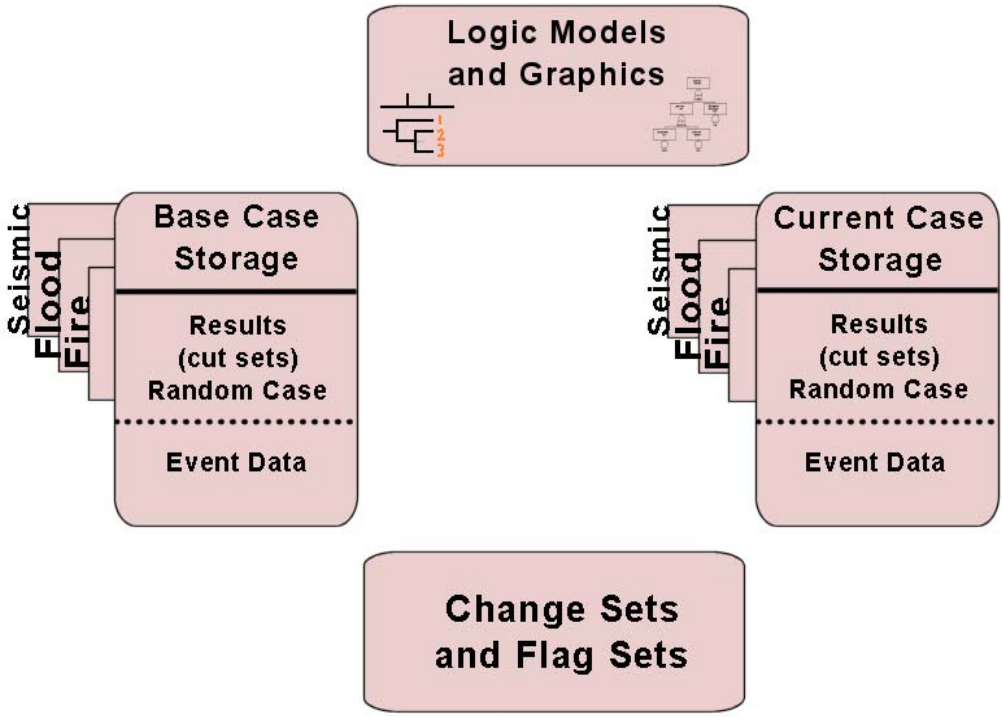
There are two additional reserved Analysis Types (which are reserved for future code enhancements) and eight additional user defined analysis types. Consequently, a total of 16 different Analysis Types are available.

The default Analysis Type is set to RANDOM. The default Analysis Type can be changed by using the **Utilities ➔ Constants** option. The first "constants" screen contains the field for the default Analysis Type.

- Changing the Analysis Type causes SAPHIRE to put the current analysis into a new case.

◊ The new current case corresponds to the Analysis Type that was selected.

◊ Since 16 different Analysis Types are available, the user could potentially have 16 different places to perform/store cut sets (one for each Analysis Type).



◊ If an event is used for one Analysis Type, for example, in a seismic analysis cut set, that event cannot be deleted from the database even if you are not currently using the seismic Analysis Type.

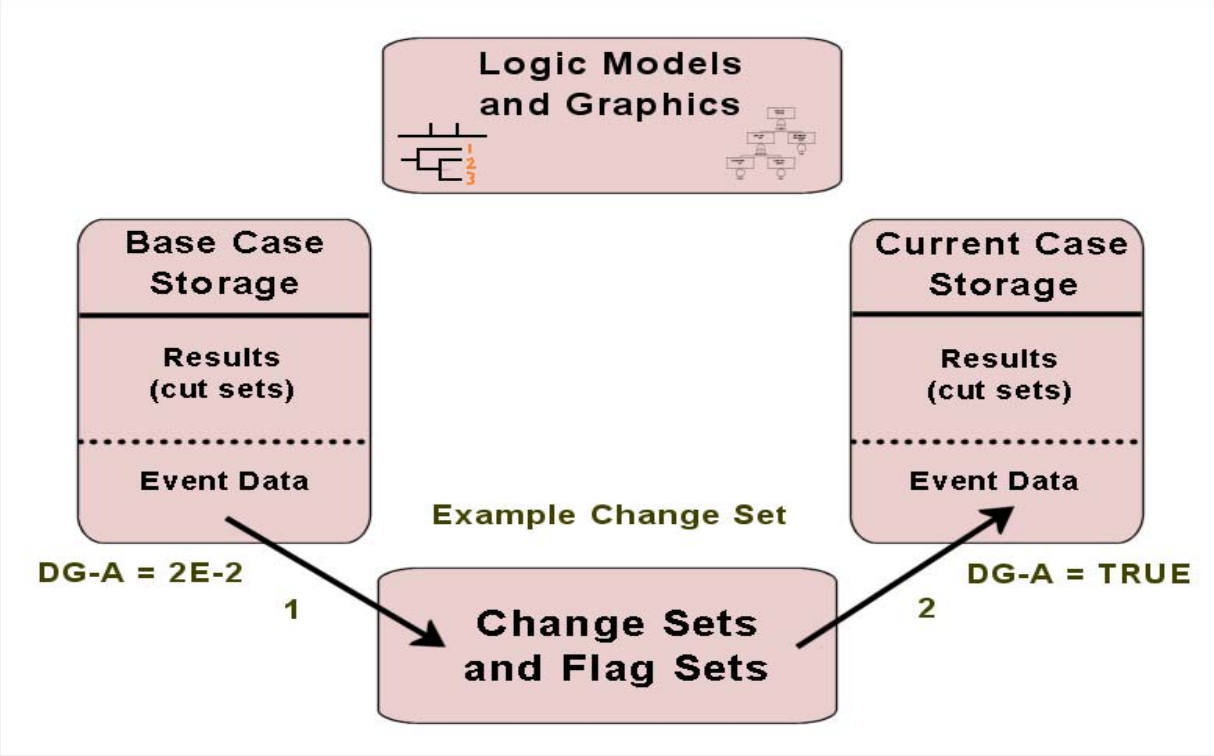
The sixteen different Analysis Types are:

#	Name	Description	#	Name	Description
1	Random	Random Failures	9	User1	User-definable
2	Fire	Fire	10	User2	User-definable
3	Flood	Flood	11	User3	User-definable
4	Seismic	Earthquakes	12	User4	User-definable
5	ASP_INIT_EVENT	ASP initiating event assessment	13	User5	User-definable
6	ASP_CONDITION	ASP condition assessment	14	User6	User-definable
7	Reserved3	Reserved for system	15	User7	User-definable
8	Reserved4	Reserved for system	16	User8	User-definable

2.4. Flag Sets and Change Sets

CHANGE SETS

Change Sets are a user-defined set of changes (think data filter) that will be applied (on the base case data) when data is transferred to the current case via the **Generate** option. Multiple change sets can be defined and applied singly or in combination.



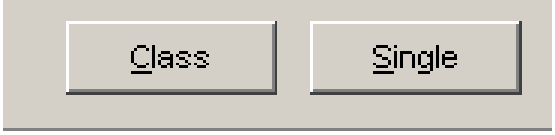
Rules for Creating and Using Change Sets

- No limit to the number of change sets that can be added to the data base.
- Change set name is limited to 24 characters; the description is limited to 120 characters.
- A change set can contain one class change and unlimited individual probability changes.
- Multiple change sets can be used in combination to create different sensitivity studies.

Class Changes

Class changes use a basic event's attribute to search for a class of basic events to which the defined change applies

- The search criteria are defined first
- The change to be applied is then defined



Single Changes

Single changes only modify individual, user-identified basic events

- The desired basic event is selected
- The changes to the basic event are then defined

The order of "marking" a change set is important. (Change sets are marked by double-clicking the line containing the change set.)

◊ The first selected change set will be the first one that is applied.
◊ Later changes will overwrite earlier ones if there is any overlap.
◊ A particular change set may include both a Class change and Single changes. *The Class change is applied first and then the Single changes are applied second*. Thus, the individual probability changes will overwrite a class change if both types are in a particular change set.

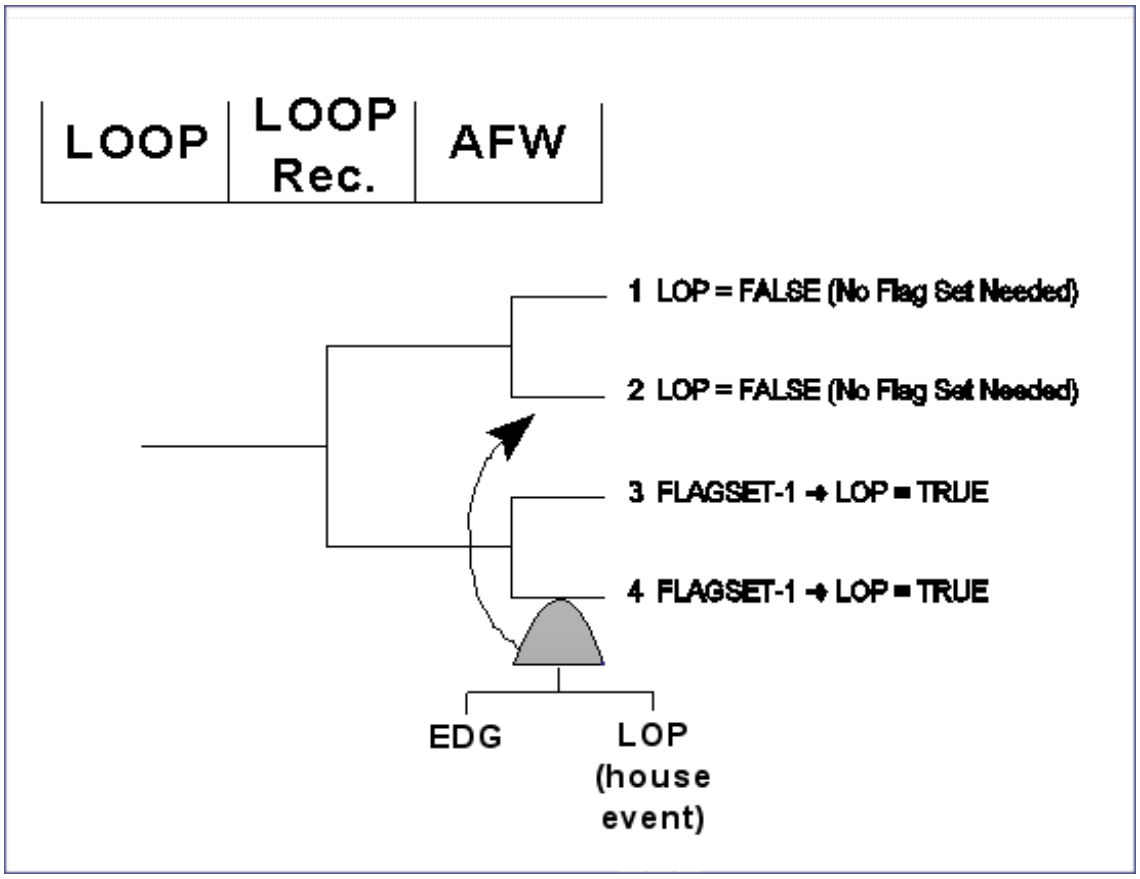
FLAG SETS

Flag Sets are a special type of Change Set. Flag Sets are created, modified, and stored in SAPHIRE under the **Modify ➔ Flags** option.

◊ Flag Sets can only contain individually specified types of changes. No "Class Changes" are allowed in a Flag Set.

◊ Flag Sets are used to indicate modifications to particular events on a sequence-by-sequence basis (or fault tree logic).

The example shows that the LOP house event is turned on (TRUE) for sequences 3 and 4. For sequences 1 and 2, the LOP house event is left off (FALSE). The setting of the house event is dependent upon the success or failure of recovering offsite ac power.



Flag Sets can only contain either house flag changes to the calculation type or process type changes. Consequently, the allowable changes that can be made in a Flag Set are:

Type of change	Allowable values
Calculation type	T (TRUE) F (FALSE) I (IGNORE)
Process Flag	X Y W I

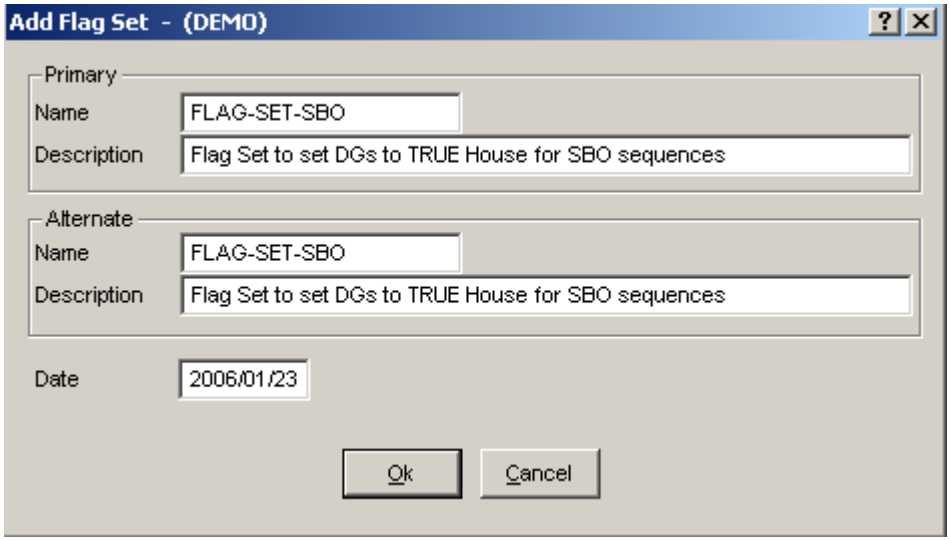
You **can not** make changes to the probability of failure (e.g., change the probability from 2E-3 to 1E-1) for events in a Flag Set.

Making a Flag Set

Enter the **Modify ➔ Flags** option.

To create a Flag Set, right click the mouse and select **Add**, and enter the Flag Set name and description.

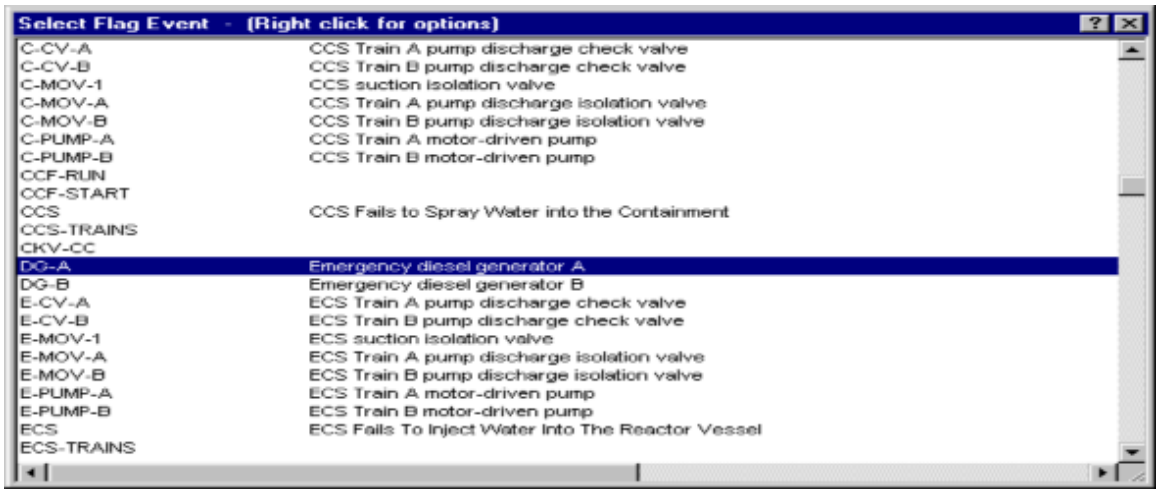
As shown in the figure, a Flag Set (named FLAG-SET-SBO) was created in this example.



To add the basic events that will be modified by the Flag Set named FLAG-SET-SBO, highlight the Flag Set and click the Flags button on bottom of the list box.

Right click and select **Add**. All of the basic events will show up in a list. Highlight the event to be modified, right click the mouse and select the **Add** option. In this example, event DG-A was edited.

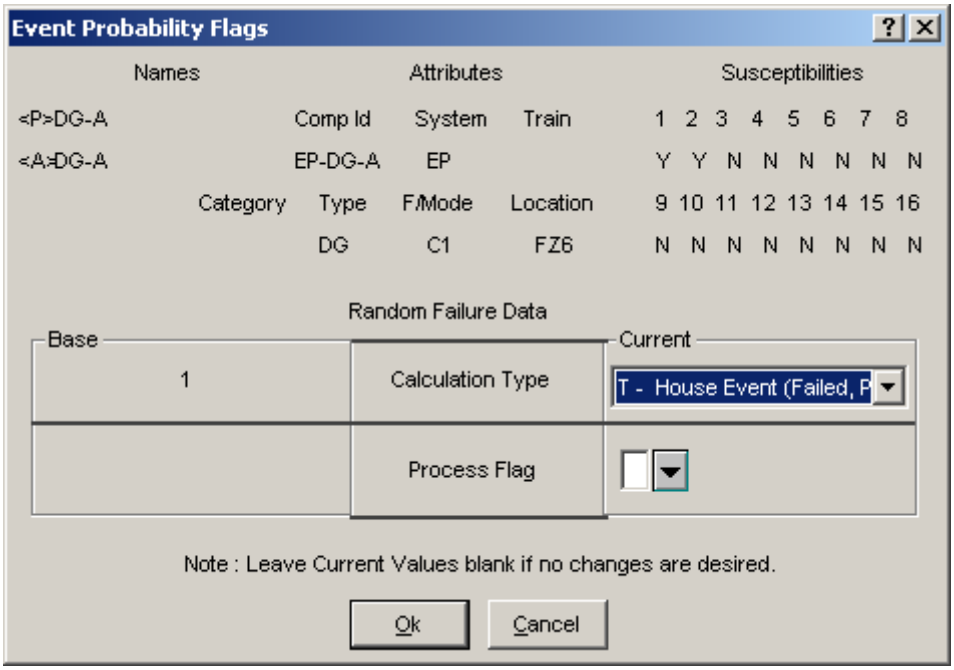
To delete a flag set, highlight the flag set to be deleted, right click and select **Delete**. A window confirming the deletion will appear. Select **Yes** to remove the flag set from the project.



Flag Set Data Entry Screen

The Flag Set data screen allows you to enter changes to the calculation type or process flag on the lower right side of the screen.

In this example, the calculation type was changed from Type 1 (i.e., a probability) with failure probability of 2.0E-2 to a TRUE house event (i.e., set to failed).



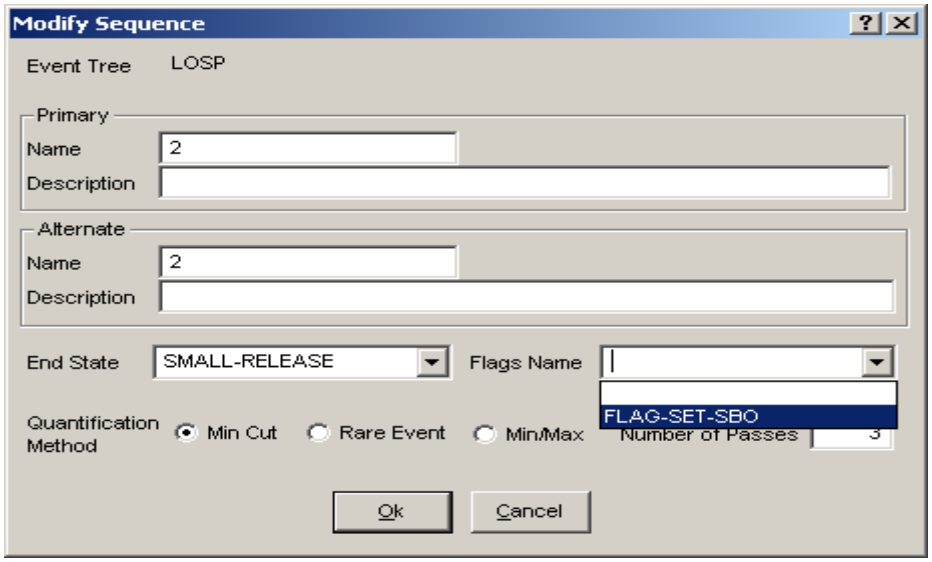
Using the Flag Set

To use a Flag Set after it has been created, you must assign the Flag Set name to a sequence or fault tree.

◊ To add the Flag set to the sequence(s), choose the **Modify ➔ Event Trees** option, then highlight the event tree and select **Sequences** on the bottom of the list box. Select the sequence that the flag set is going to be assigned to.

◊ The Flag Set name is entered in the field labeled "Flags Name."

◊ The example shows the Flag Set "FLAG-SET-SBO" is assigned to sequence 2 in the LOSP event tree.

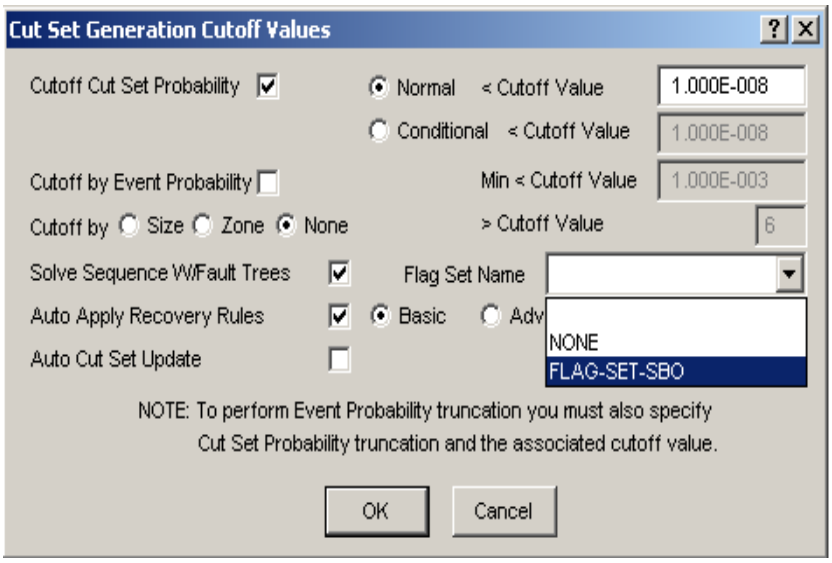


An alternative way to assign a Flag Set to sequences would be to specify the Flag Set name when generating sequence cut sets. But, this approach is only a temporary change and, as such, is not recommended.

If the "Flag Set Name" field is left blank, SAPHIRE will use the assigned sequence Flag Sets.

If the word "NONE" appear in the "Flag Set Name" field, SAPHIRE will ignore all assigned sequence Flag Sets.

A valid Flag Set in the "Flag Set Name" field will be used by SAPHIRE as the default Flag Set for that analysis.



2.5. Dynamic Flag Sets

"Dynamic" Flag Sets are a special type of Flag Set that are assigned to sequences by the use of event tree rules. In other words, they are rule-based flag sets.

A Dynamic Flag Set is assigned to a sequence if the sequence meets the search criteria contained in the rule.

Advantages of "Dynamic" Flag Sets versus tradition Flag Sets (discussed above) are:

- ◊ Given a change in the event tree logic, the Dynamic Flag Set will automatically assign itself to the sequence that meets the search criteria contained in the rule.

- ◊ For example, if the rule assigns a Flag Set to sequence LOOP 05 and the event tree logic changes to make this sequence LOOP 08, then the Flag Set will automatically be assigned to LOOP 08 once the event tree sequences are regenerated.

- ◊ Dynamic Flag Sets are created and assigned to the sequences every time the event tree sequence logic is generated via the "link event tree" option.

Dynamic Flag Sets are the same as Flag Sets. Thus, only basic event calculation types can be changed and the change can only be specified to individual basic events (i.e., no class changes).

The Dynamic Flag Set name will appear in the Flag Set field under **Modify ➔ Flags** after the flag set is created during the event tree linking process. The Dynamic Flag Set name is assigned by SAPHIRE and is based upon the event tree, sequence name, and number of Flag Sets used. The user does not have control over the naming process.

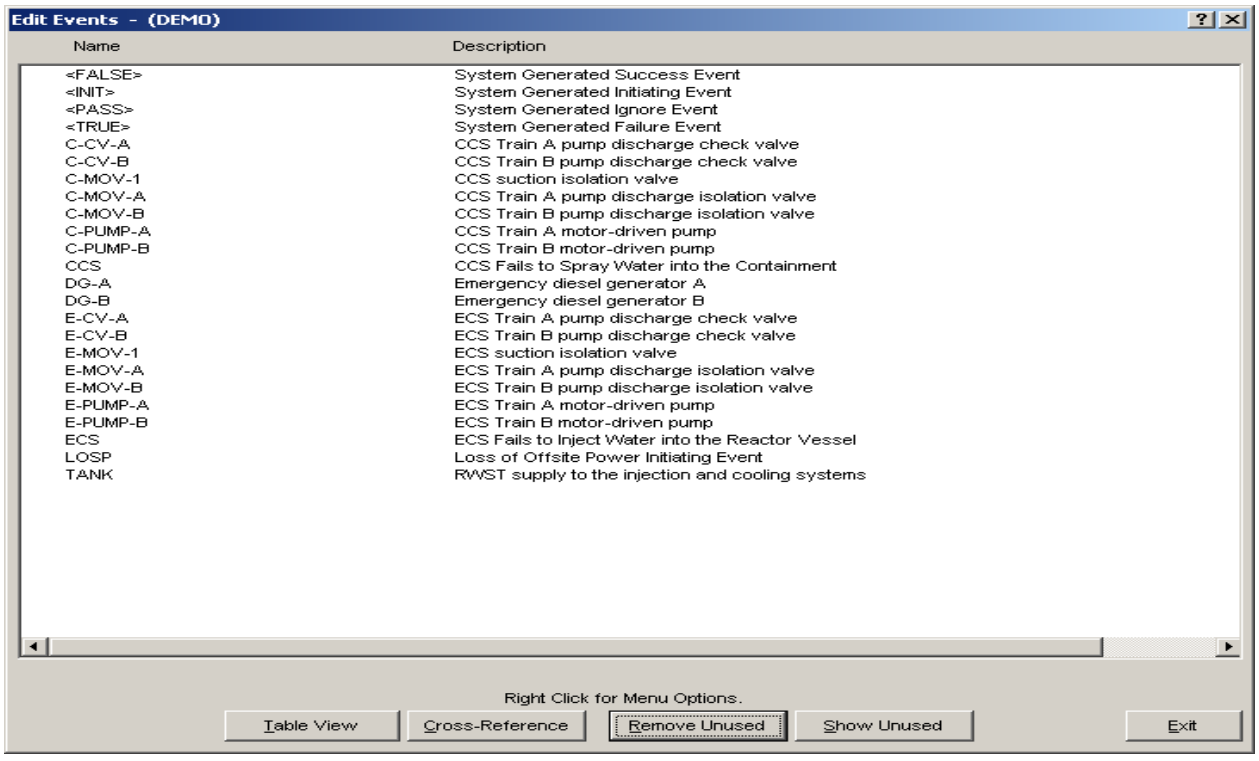
Dynamic Flag Sets will be discussed in greater detail in Section 9. This section will demonstrate the how to use the Dynamic Flag Sets and the different features of the Dynamic Flag Set rules.

3 BASIC EVENT INFORMATION

Section 3 introduces the template event, compound event features, and human error worksheets found in SAPHIRE. The template event option allows you to specify basic event information that can be used by multiple basic events. The compound event allows SAPHIRE to use built-in numerical library to determine a basic event's probability. The human error worksheets are used to calculate human error probabilities.

3.1. Modify Basic Events

- To enter basic event data, select **Modify** from the menu. Then select **Basic Events**.



- To modify data for an existing event, double-click on the event you want to edit or right-click to invoke the pop-up menu and select **Modify**.

3.2. Template Event

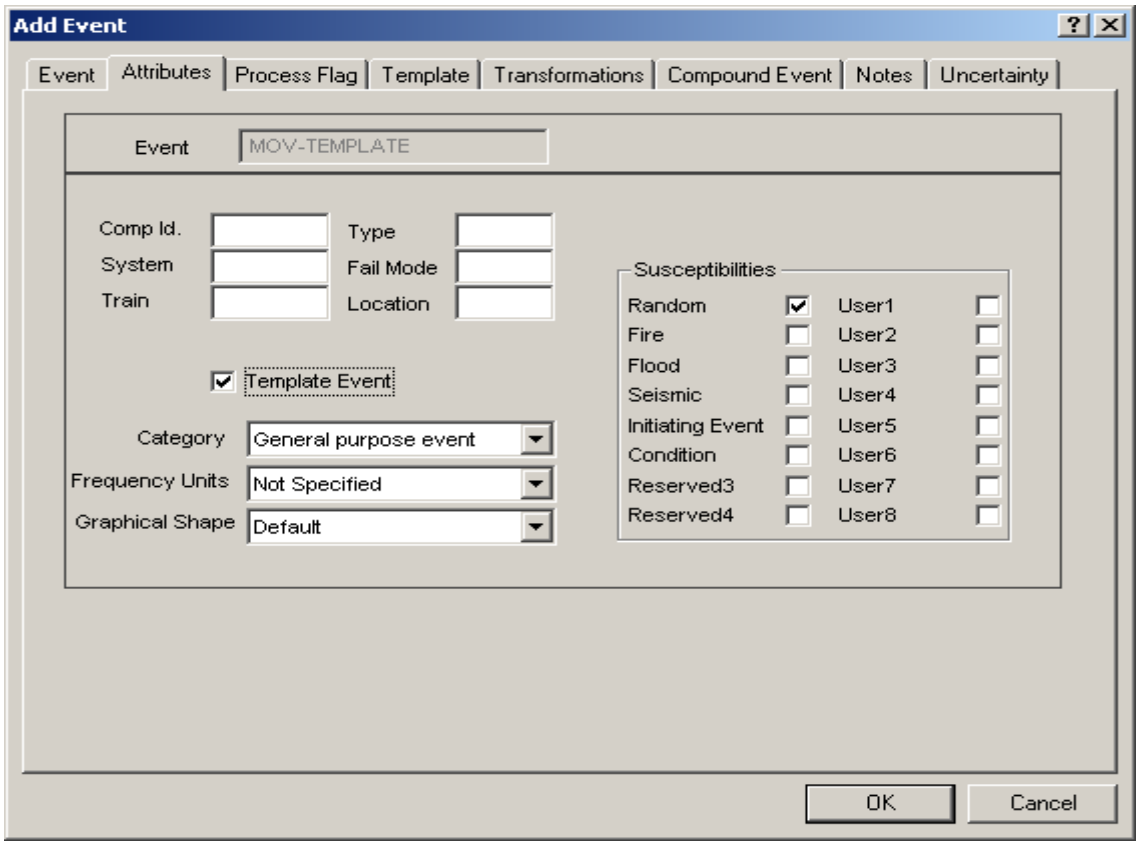
Template events are basic events that can be shared. For example, if a PRA contains 15 individual motor operated valve (MOV) basic events, one can refer to a single template MOV instead of typing the MOV failure rate 15 times. Template events can be set up so that the event description, failure data, uncertainty data, attributes, and/or process flags can be used multiple times.

To create a template event, go to **Modify ➔ Basic Events** then…

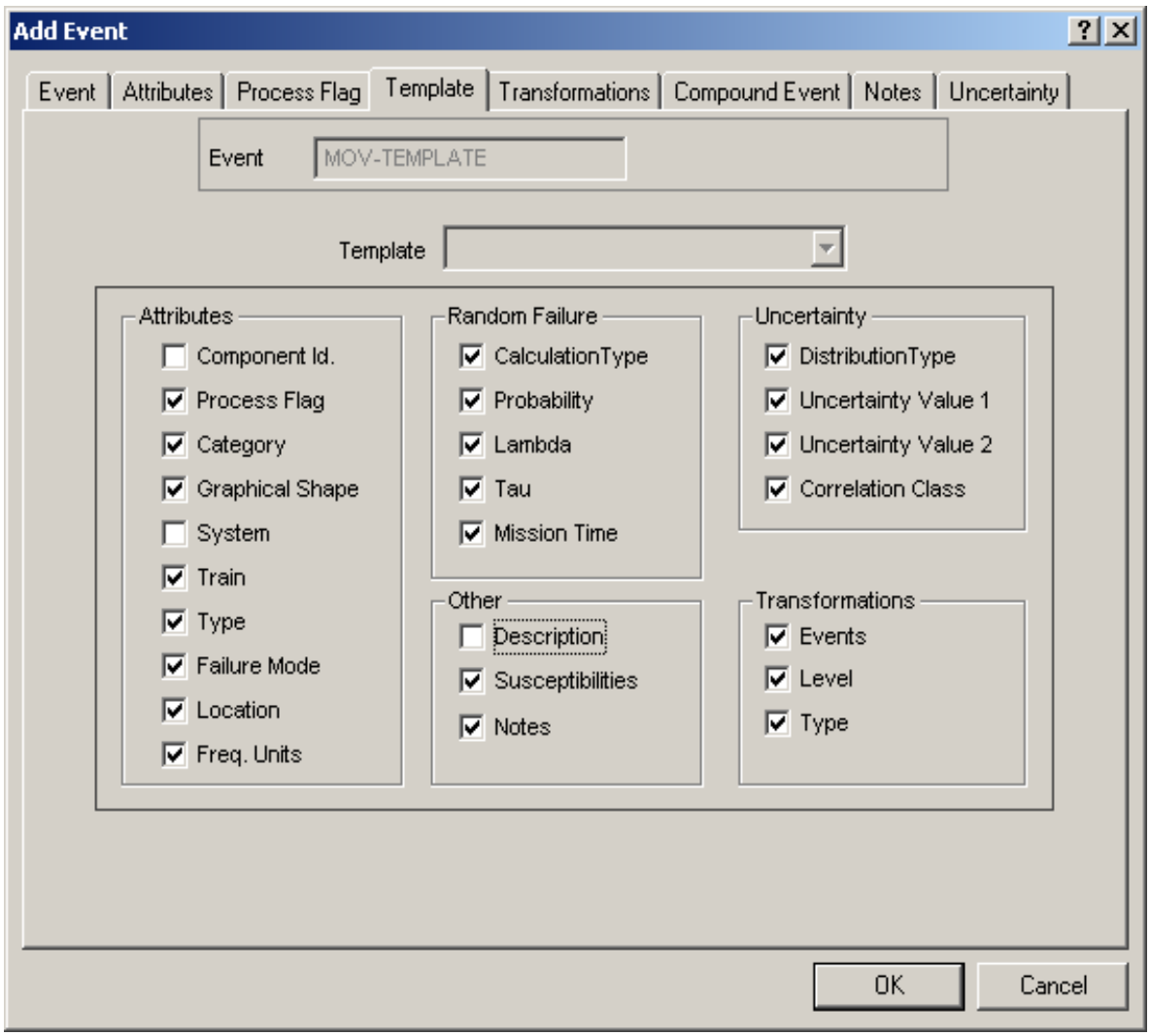
◊ Click the right mouse button and select **Add**.

◊ Input the basic event description, failure data, and uncertainty data that are associated with this template.

◊ To make SAPHIRE use this event as a template event, select the Template Event check box under the **Attributes** tab.



Now, click the **Template** tab. You will need to indicate which information is to be shared by this template event. If a box is checked (see below) then that information will be shared.



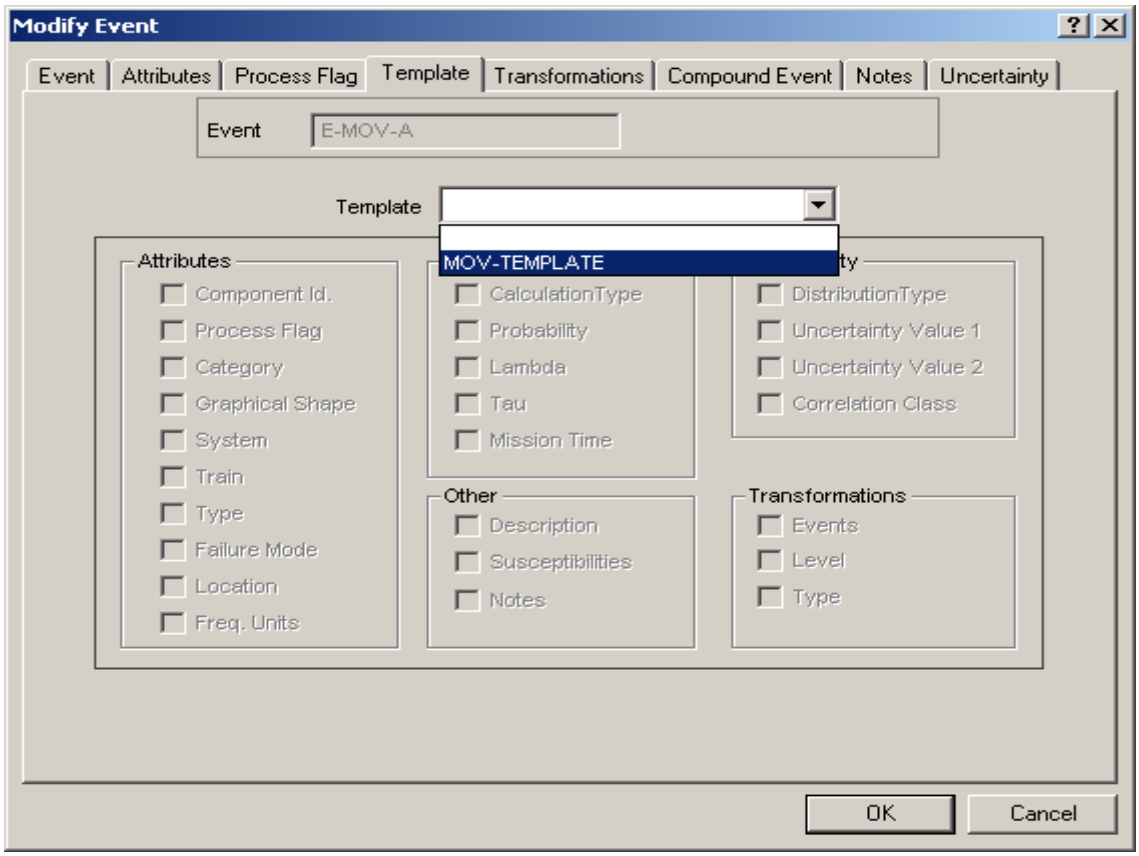
Notice that one generally does not want to share the component ID, system, and description since the events that use the template will have (possibly) a unique ID, system, and description.

After the template is created, a "-" will appear on the left hand side of the template event. The "-" indicates this event is not used in the database. This "-" will remain next to the template event until it is used by another basic event. This nomenclature looks like…

To use a template event…

◊ Highlight the basic event that will use the template event. (In this case, basic event E-MOV-A was selected.)

◊ Click the **Template** tab and click the "Template" down arrow to list all of the template events in the database. Select the applicable template event.

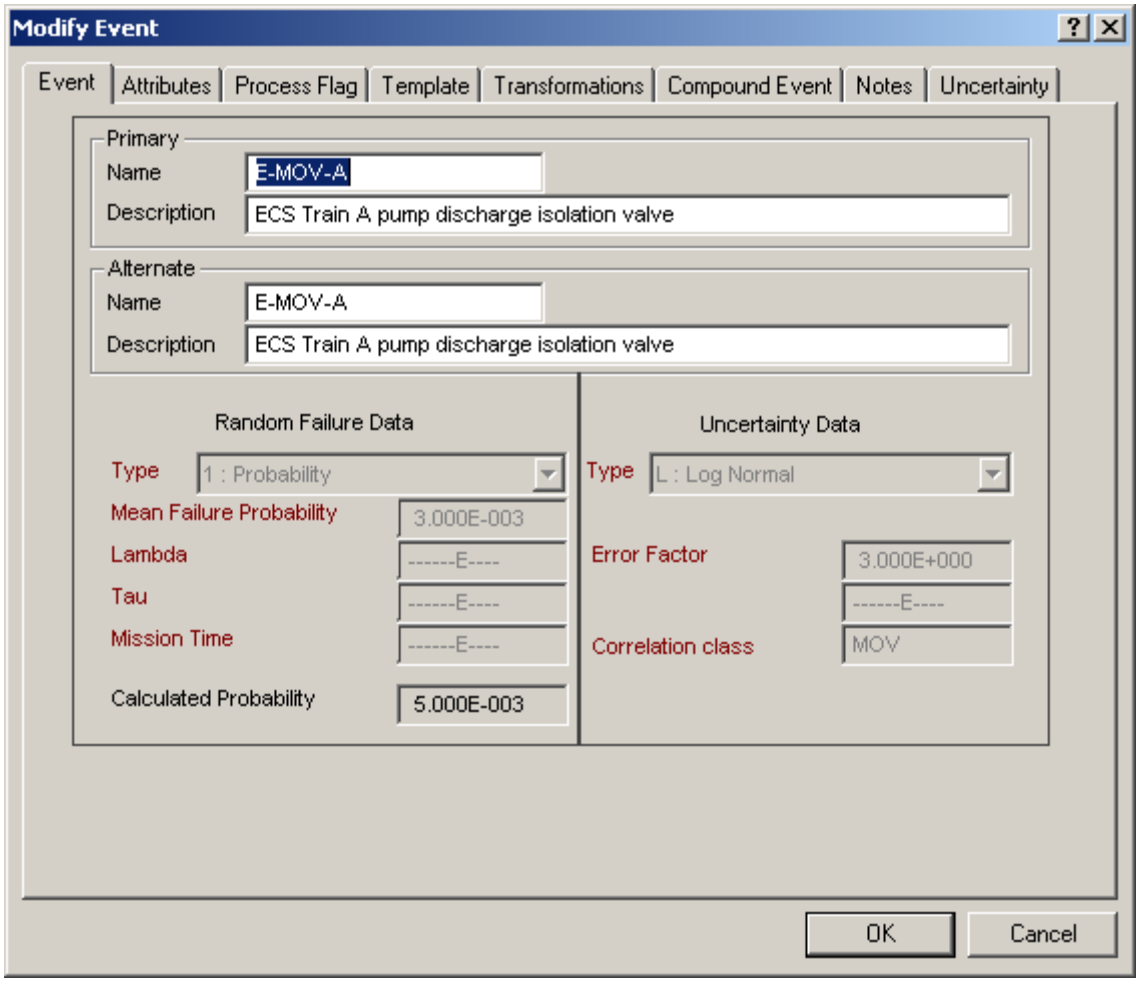


Now, basic event E-MOV-A will use the MOV-TEMPLATE information that is indicated by the checked boxes.

◊ If desired, information from the template event can be ignored by unchecking any of the applicable check boxes.

◊ If a box is unchecked, the user will need to supply that information. For example, if the template failure rate "lambda" is not used, the failure rate would need to be specified just like a traditional basic event.

Once the user indicates the template event to use, SAPHIRE copies the template information into the applicable basic event fields.



Data shown with labels in red text (and the field is grayed out) represents the information that was carried over from the template event. This information cannot be modified unless you deselect the particular input field (back on the template tab).

The advantage of using template events is if a parameter changes, the parameter only has to be changed once at the template event. Then, all the basic events using the template event will be updated automatically.

3.3. Compound Events

Compound events are basic events that use an external calculation to determine the event probability. Simple examples of compound events include the arithmetic addition of multiple basic events or the product of multiple basic events. More complex compound events include calculations for "supercomponents," common-cause failure, and flow accelerated corrosion.

A compound event is generally expressed as a function of other basic events (within the same project). For example, in the "supercomponent" case, one would identify the components (up to 20) that make up the supercomponent.

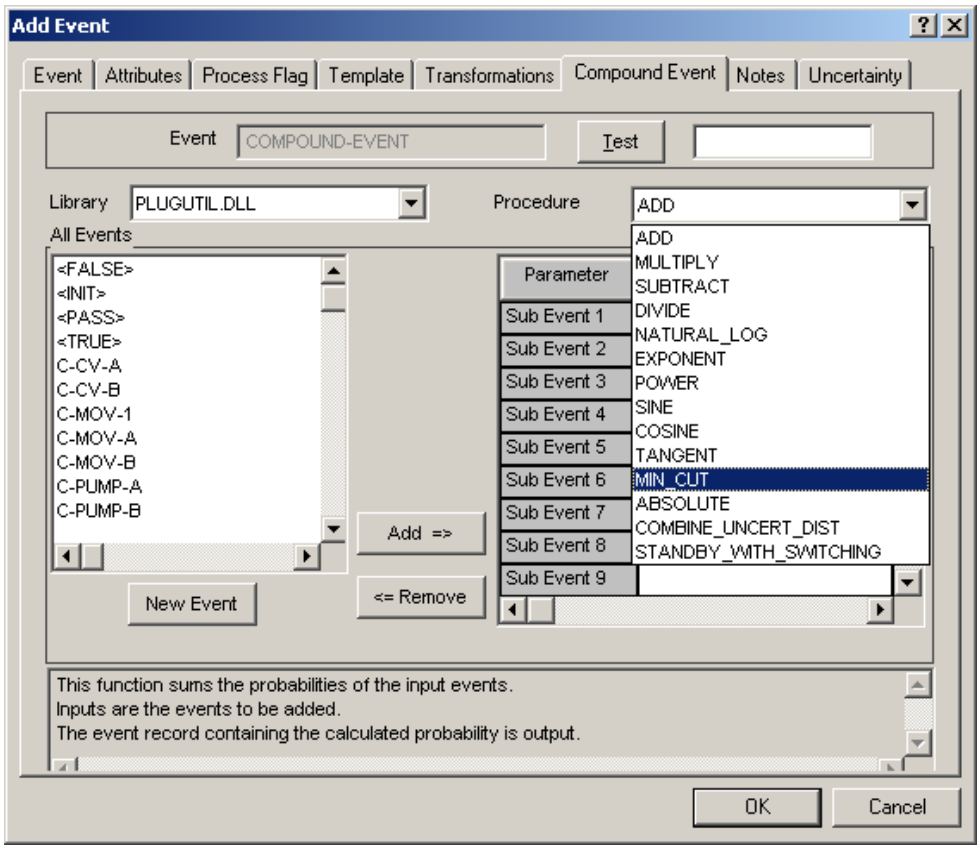
To create a compound event, go to **Modify ➔ Basic Events** then…

- ◊ Highlight the basic event to be the compound event, click the right mouse button, and select **Modify**.
- ◊ Under the Random Failure Data, the drop down list for **Type** needs to be changed to **C : Compound Event**.
- ◊ Click the **Compound Event** tab.

The "Library" drop down option lists the different modules available to the analyst.

To create a super-component event, the PLUGUTIL.DLL library is selected.

Then, for the "Procedure" option, select the MIN_CUT equation.

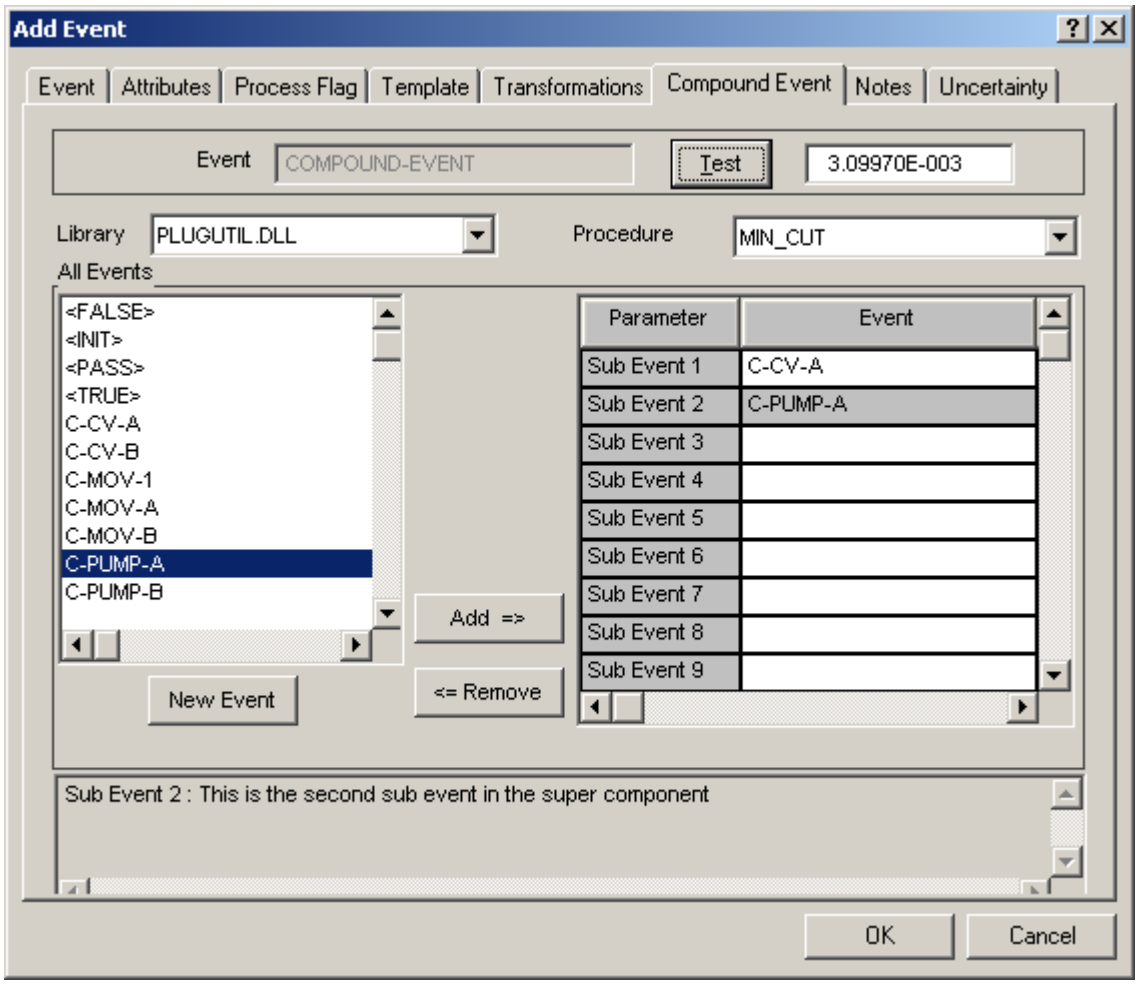


The MIN_CUT joins several basic events together (as if they were in an OR gate) using the minimal cut set upperbound approximation to determine the probability.

Next, the column on the right lists what basic events (listed by Sub Event #) are going to be used to make-up the compound event. The list on the left-hand side lists every basic event in the project.

Highlight the "Sub Event #" line where a basic event is to be used. Then, highlight the basic event from the list of all basic events on the left and click the **Add** button. Continue this process until all of the basic events that make-up the compound event have been added to the "Selected Events" list.

The **Test** button calculates the probability that will be used for the compound event. You can use this test to check that data is correct and the calculation is being performed as desired.



3.4. Common-Cause Failure Compound Events

Common cause failure basic events are used to represent simultaneous failures of multiple components due to a single cause or mechanism. The common-cause basic event represents a model that calculates the probability of a shared cause failing multiple trains of similar components.

Within SAPHIRE, there are two basic types of common-cause models. The first is known as the Multiple Greek Letter (MGL) method. The second is known as the Alpha Factor method.

Multiple Greek Letter (MGL) method

To use the MGL model in SAPHIRE, go to **Modify ➔ Basic Events** then…

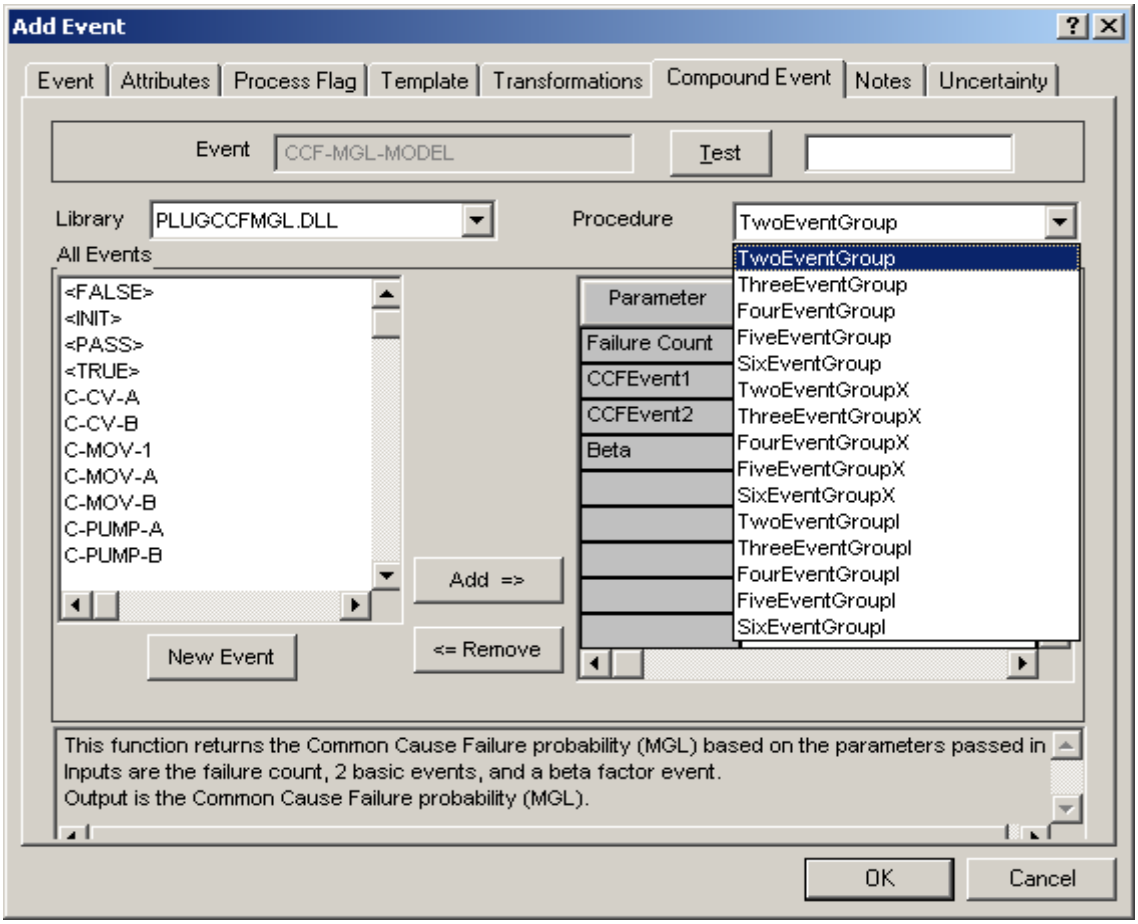
- ◊ Highlight the event that represents the common-cause failure, right click, and select **Modify**.
- ◊ Select the **C : Compound Event** calculation type, then click the **Compound Event** tab.
- ◊ The “Library” drop down option lists the different modules available to the analyst. To use the MGL model select PLUGCCFMGL.DLL.

Once the PLUGCCFMGL.DLL library is selected, the “Procedure” option allows the analyst to specify the number of components that comprise this common-cause failure basic event.

If there are two redundant components represented by the common-cause event, select the TwoEventGroup procedure. Once this is selected, the required inputs are:

- ◊ The failure count
- ◊ The two independent component basic events
- ◊ The beta factor

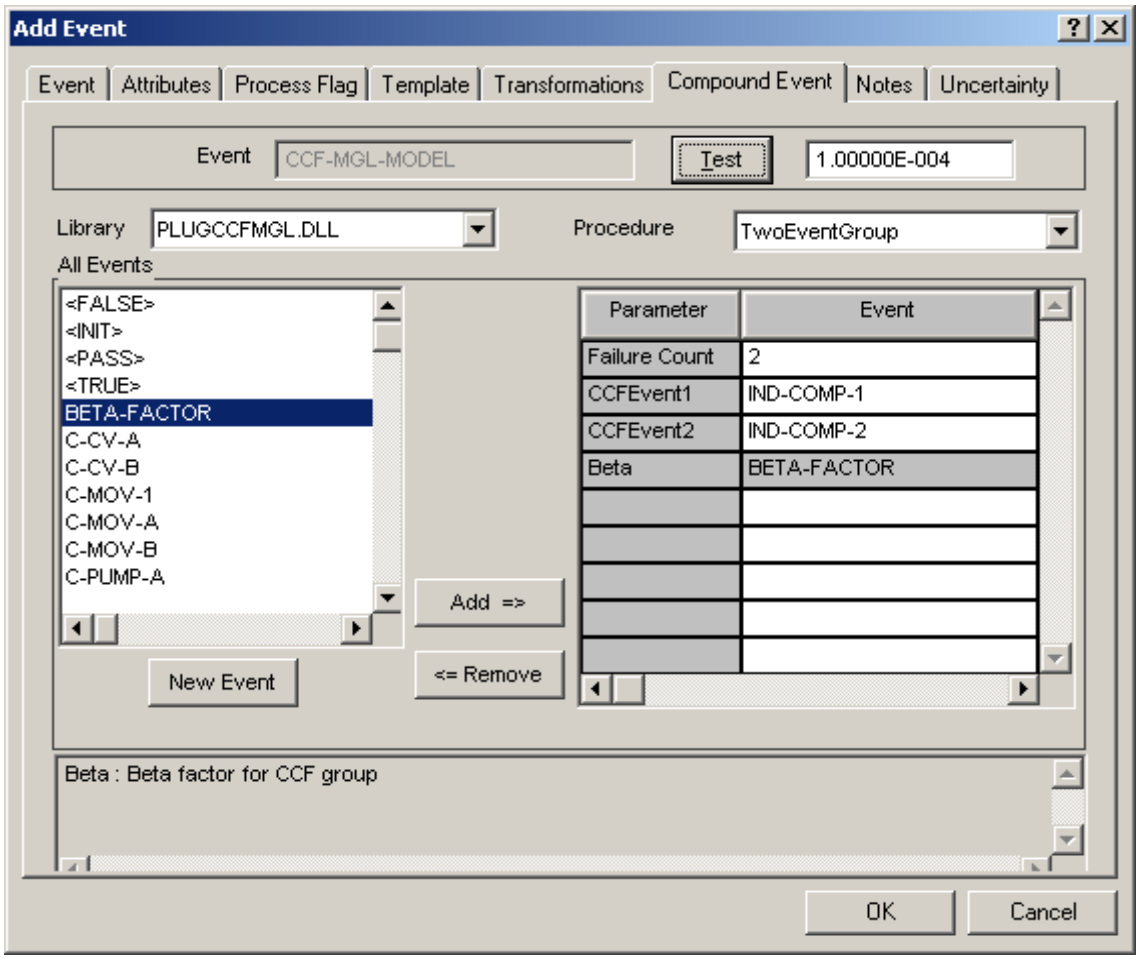
SAPHIRE will use this information to calculate the common-cause failure probability.



The advantages of using the MGL equation built into SAPHIRE are:

- ◊ Automatic calculation of the nominal common-cause failure probability
- ◊ Utilization of the uncertainty defined for the independent events.
- ◊ SAPHIRE automatically adjusts the common-cause probability if an independent event is set to a failed state.

The New Event button may be used to create new events during the creation of the compound event itself. For example, in the Multiple Greek Letter example above, the Beta Factor would be represented by a single basic event (called BETA-FACTOR) that could be created by clicking the **New Event** button and inputting the necessary values, saving it by clicking the **OK** button and returning back to the **Compound Event** tab.



Alpha Factor method

The second common-cause model is the Alpha Factor model. In SAPHIRE, this model uses one of two different equations, depending on the type of testing for the components in question.

- ◊ The first module (PLUGCCFSTAG.DLL) is based upon a staggered testing scheme.
- ◊ The second module (PLUGCCFALPHA.DLL) is based upon a non-staggered testing scheme.

More information pertaining to the Alpha Factor model can be found in NUREG/CR-5485.

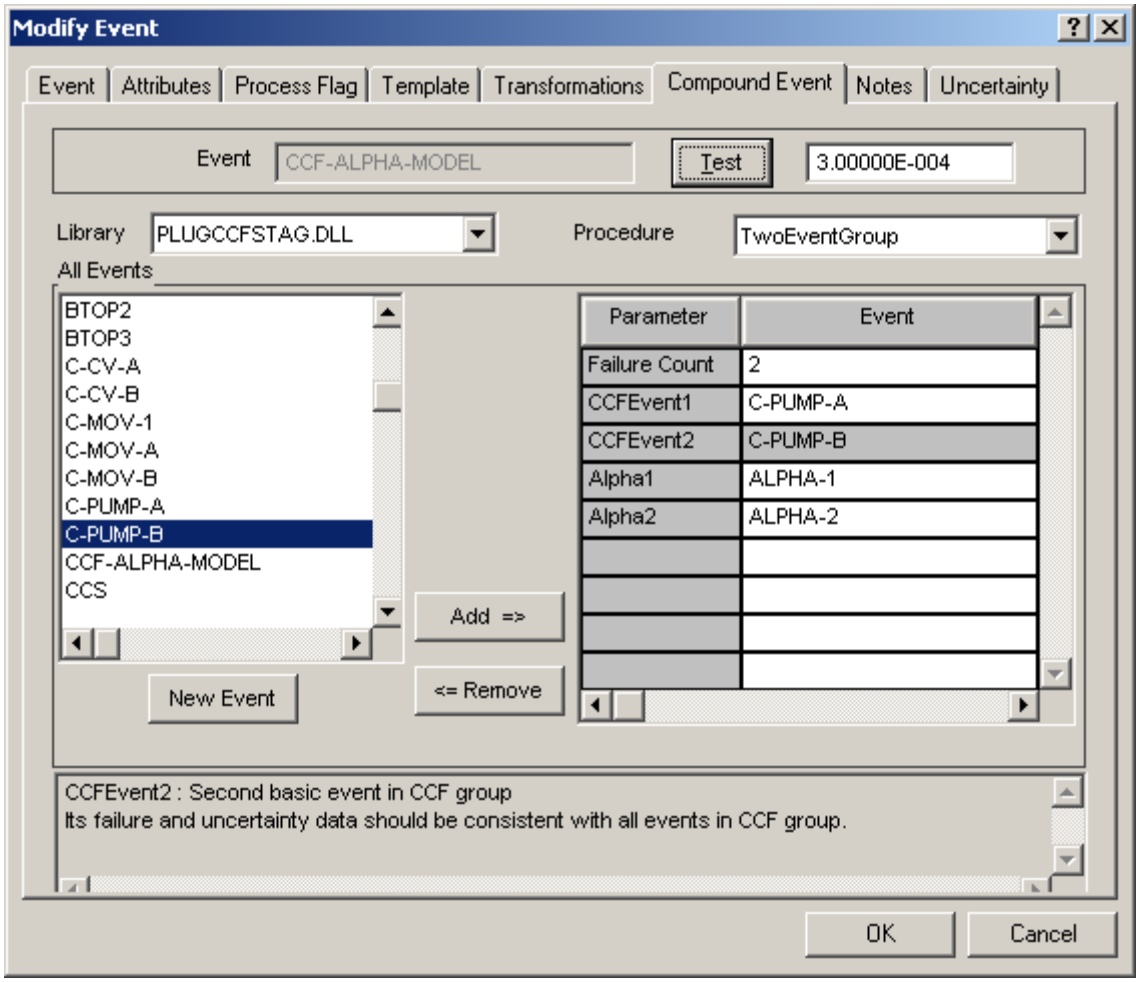
To use the Alpha Factor model in SAPHIRE…

◊ Highlight the common-cause basic event, right click, and select **Modify**.

◊ Select the **C : Compound Event** calculation type, then select the **Compound Event** tab.

◊ For the "Library" option, select either PLUGCCFALPHA.DLL or PLUGCCFSTAG.DLL.

◊ Selected the applicable "Procedure" option, representing the number of components that comprise this common-cause failure.



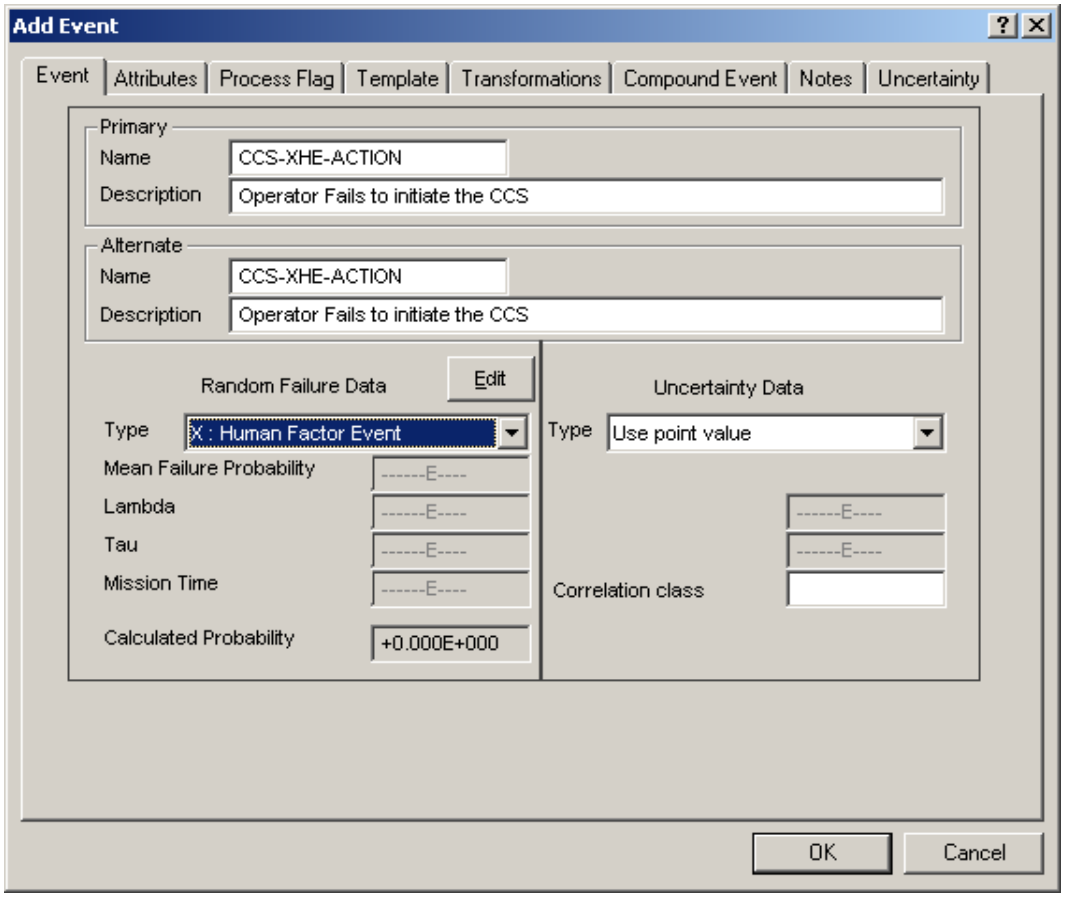
3.5. Human Error Event

Human error events are basic events that use an external calculation (via worksheets) to determine their probability. The human error probability (HEP) calculation is based on the Standardized Plant Analysis Risk (SPAR) Human Reliability (HRA) methodology (reference 3-1). A simple walkthrough of the worksheets will be presented to show the process that SAPHIRE uses to calculate the HEP using this module.

The HEP is calculated based on whether the operator action requires diagnosis or just an action. Reference 3-1 provides definitions and information about both operator diagnosis and operator action. This section is not designed to discuss the HRA methodology nor differences between an operator diagnosis or operator action but to present the different screens (i.e., worksheets) that SAPHIRE uses to calculate the HEP. For more information about this HRA, refer to reference 3-1.

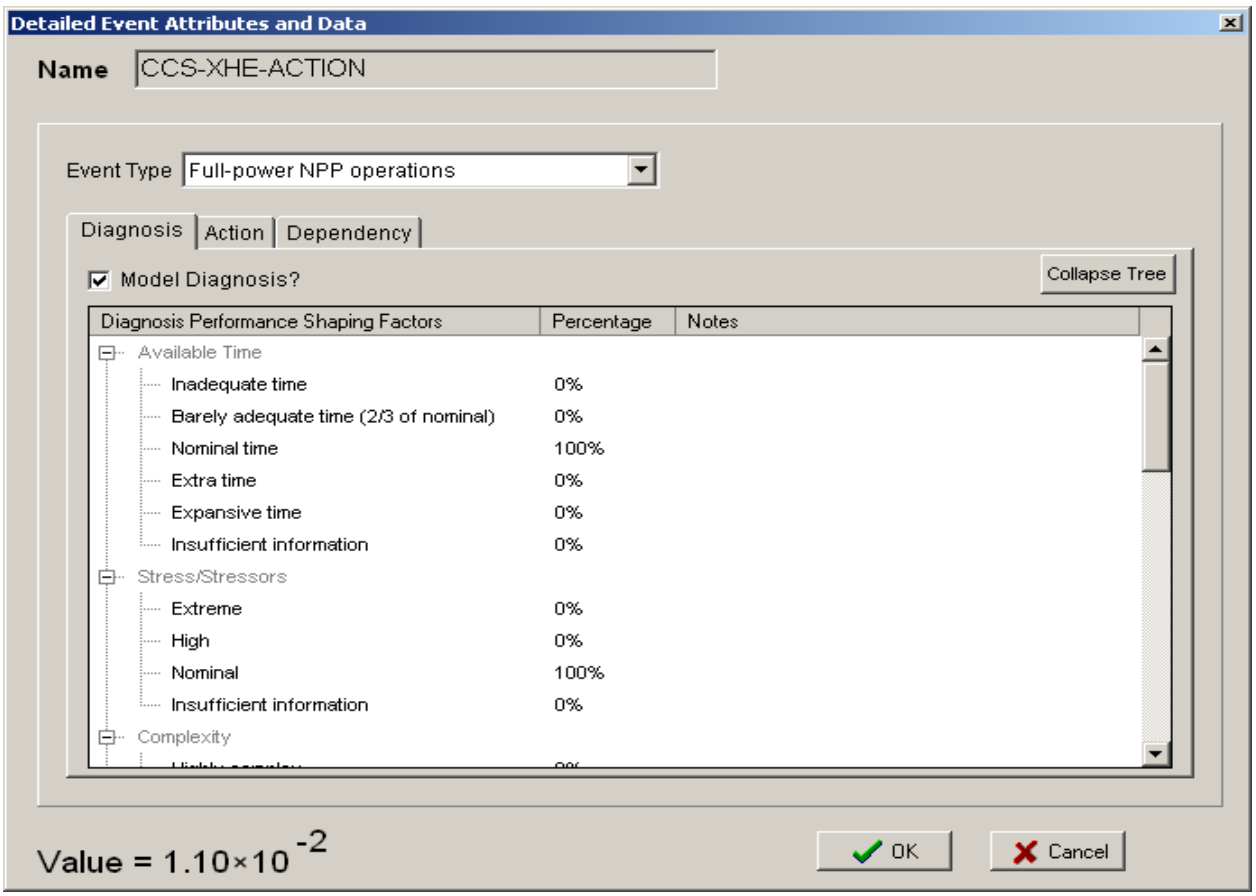
- To modify the operator actions (HEP) within the database, highlight the basic event, right click and select **Modify**.
  - Select the **X : Human Factor Event** calculation type.
  - An **Edit** button will show up right above the Type drop down option. Select this button in order to activate the worksheets used to calculate the HEP.

Note: The X Calculation Type (Human Factor Event) is only required if using the SPAR HRA methodology described above. Human error events can also be entered into SAPHIRE using Calculation Type 1 and providing the mean failure probability for the HEP.



- The first screen allows the diagnosis portion of the operator event to be modified. This screen is filled out only if the operator is required to perform some type of diagnosis prior to performing an action.

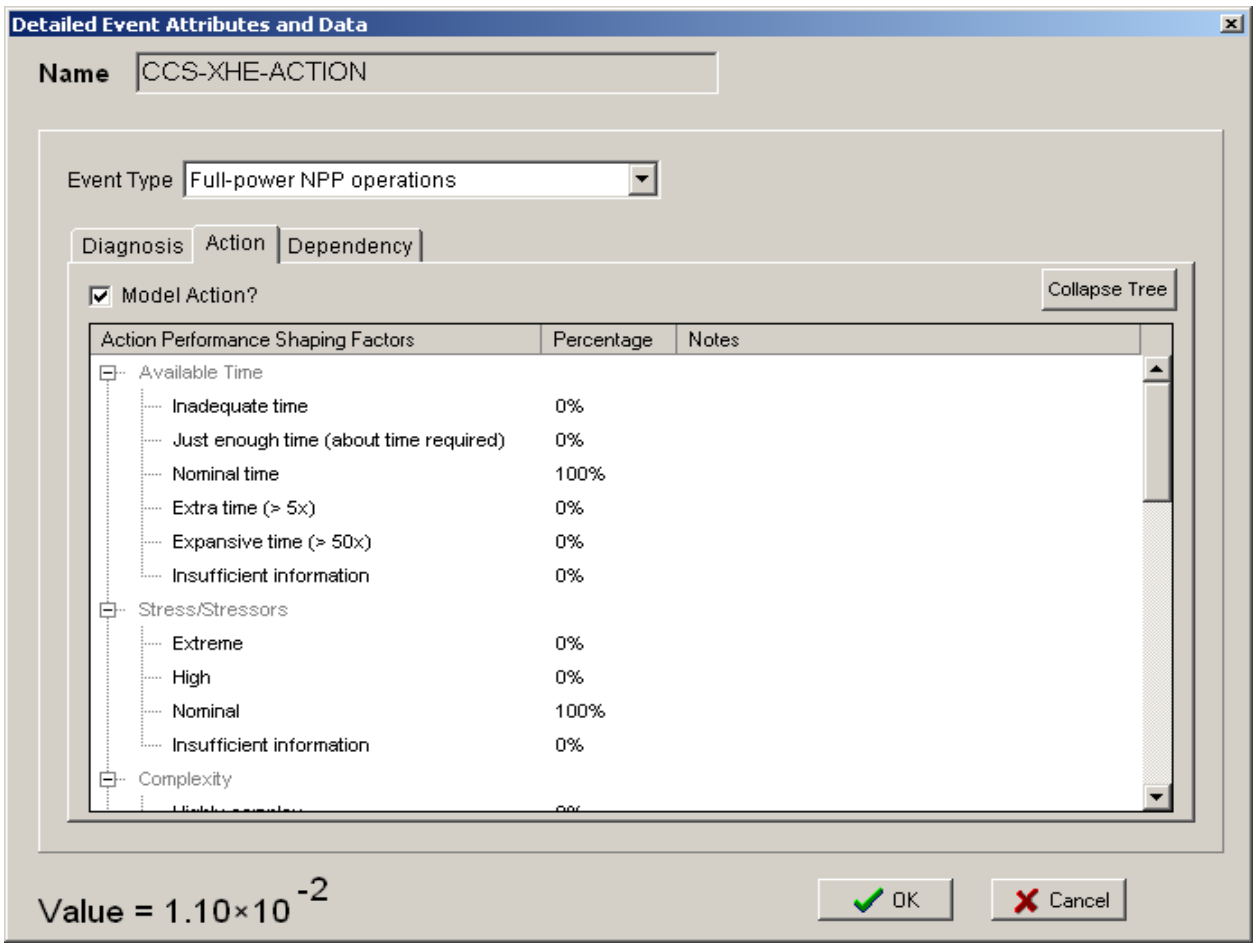
  - Each of the shaping factors can be modified individually. These shaping factors are used to modify the nominal probability for diagnosis, which is 1.0E-2. Within each of the different shaping factors, a percentage can be specified in order to determine the shaping factor value that will be multiplied to the nominal probability. If 100% is specified in the nominal field for all of the shaping factors then the nominal probability will be calculated since all shaping factors will be 1.0.



◊ If the operator event does not require a diagnosis prior to performing the action then this option can be removed from the calculation by de-selecting **Model Diagnosis?** check box.

◊ The probability is continually updated and shown on the bottom of the screen in order to show the analyst how the shaping factors are affecting the HEP.

♦ The second screen allows the action portion of the operator event to be modified. This screen is filled out only if the operator is required to perform some type of action.

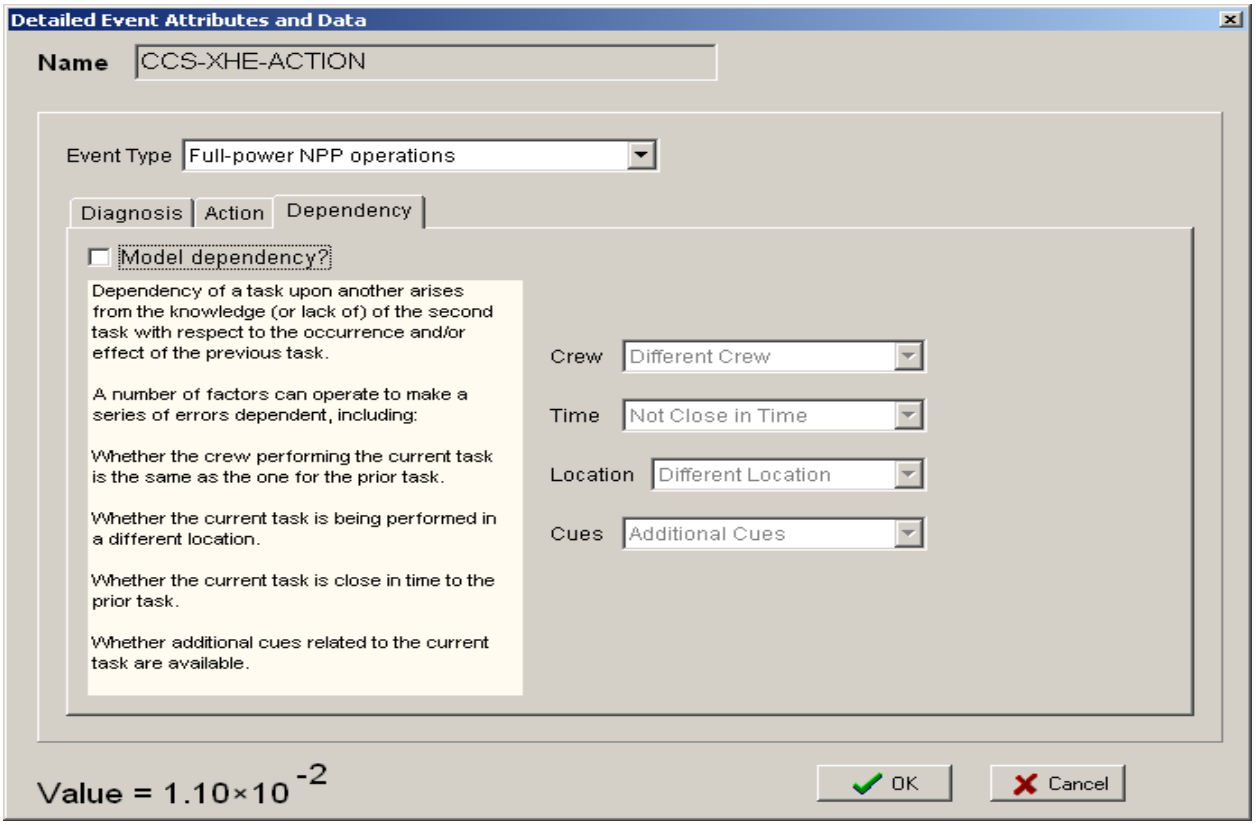


◊ As like the diagnosis worksheet, each of the shaping factors can be modified individually. These shaping factors are used to modify the nominal probability for action, which is 1.0E-3. The percentage value can be modified for each performance shaping factor in order to determine the shaping factor value, which will be multiplied to the nominal probability. If 100% is specified in the nominal field for all shaping factors then the nominal probability will be calculated since all of the shaping factors will be 1.0.

◊ If the operator action does not require an action then this option can be removed from the calculation by de-selecting **Model Action?** check box.

◊ The probability is continually updated and shown on the bottom of the screen in order to show the analyst how the shaping factors are affecting the HEP.

♦ The last screen is for dependent operator actions. A dependent operator action is one that follows a previous operator action (i.e., more than one operator action failing to perform there function within a sequence). This dependency page will change the HEP depending on the selections.



◊ To make the operator event and dependent action, check the **Model dependency?** check box.

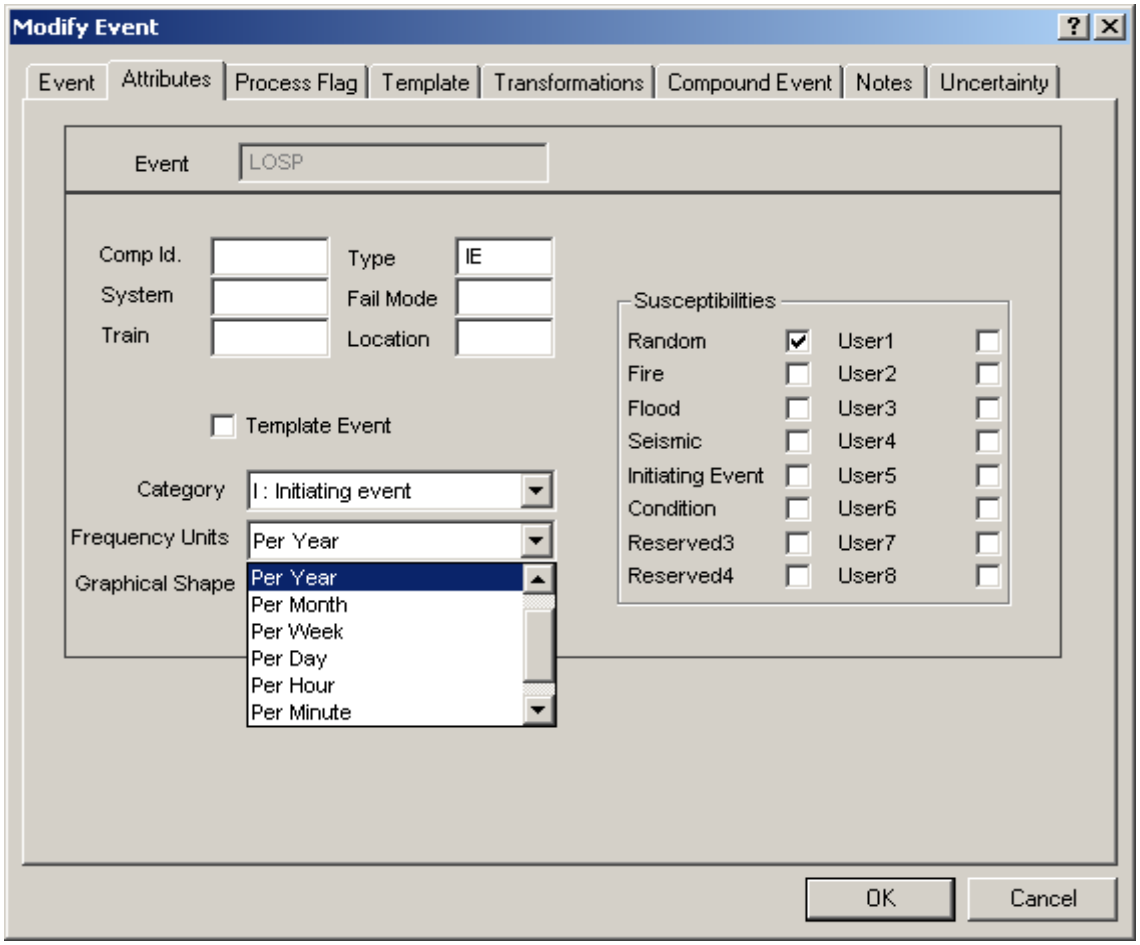
◊ The HEP for a dependent operator action is based on one of the four different questions. These questions are crew, time, location and cues. Each one of these will have an impact on the overall HEP.

3.6. Data Base Units

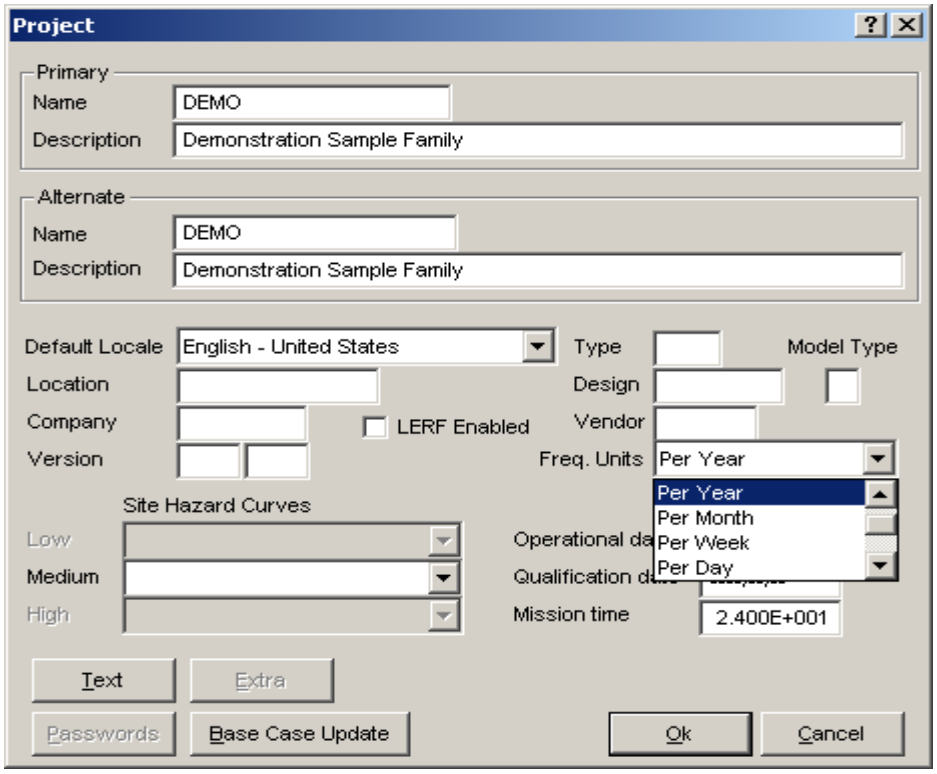
SAPHIRE 7 is designed to handle different frequency units and make proper conversions when necessary.

- The different frequency units that can be defined in SAPHIRE are:
  not specified
  per year
  per month
  per week
  per day
  per hour
  per minute
  per demand

- The frequency units for an individual initiating event can be specified in the **Modify ➔ Basic Events** option under the **Attributes** tab.



- The frequency units can also be specified in the **Modify ➔ Project** option.



- If no frequency units are specified for the individual initiating events, the default units specified in the **Modify ➔ Project** field will be used for all initiating events.

- In fact, the units specified in the **Modify ➔ Project** will be used no matter what the specified frequency units are for all initiating event. This is performed in order to ensure the final sequence results are in the same units. SAPHIRE will automatically make the conversion based on the units specified in **Modify ➔ Project**.

  - SAPHIRE will look at the units, if specified for the initiating event, prior to converting the frequency to the units specified in the **Modify ➔ Project**. This check is performed to guarantee correct conversion.

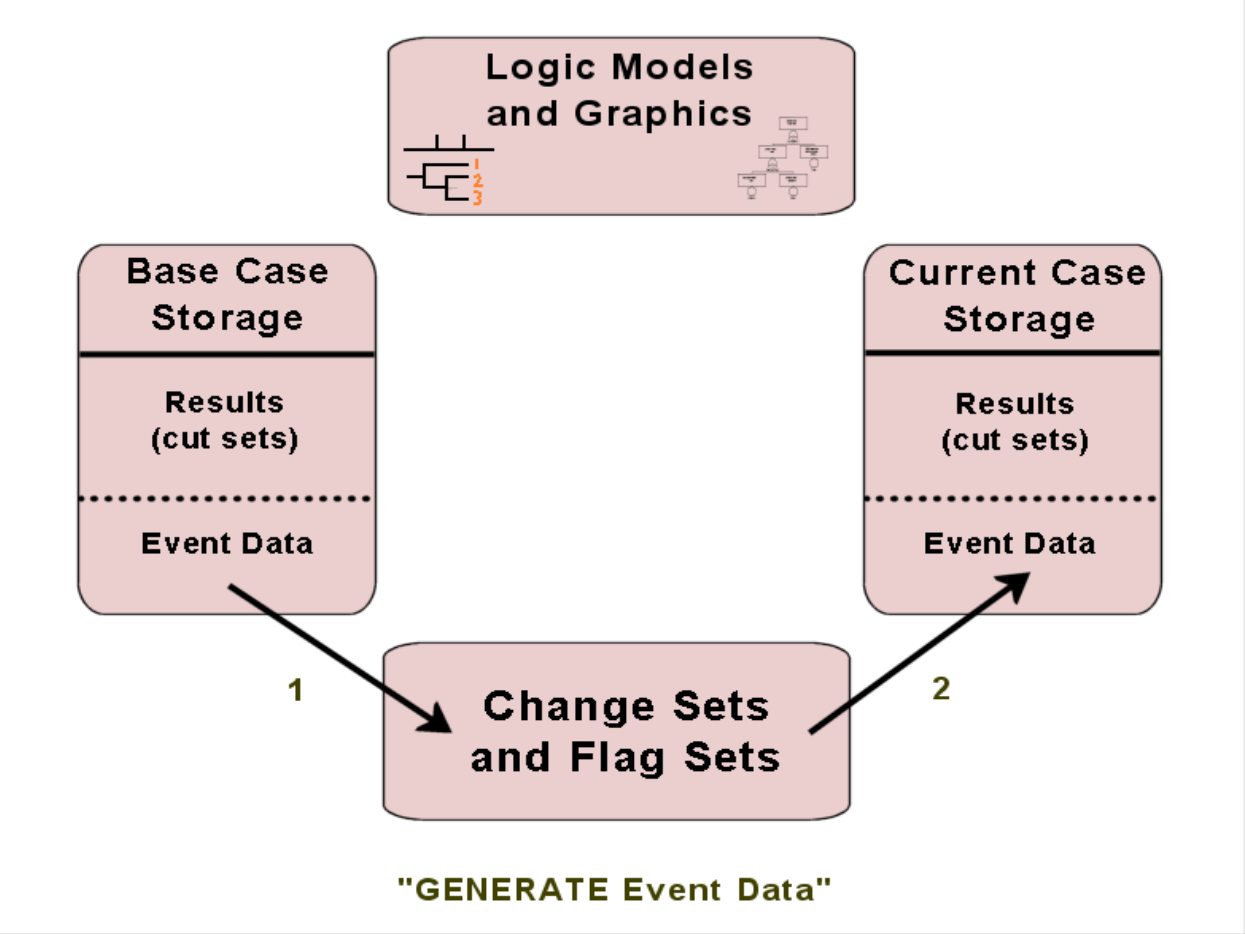
- If the "not specified" frequency units are specified in the **Modify ➔ Project**, then the units specified for the initiating events will be used. (One must be careful, since different units can be specified and the overall summation of the sequences won't be correct.)

3.7. Using “Generate” to Process Event Data

Once all of the basic event data is entered into the project, the data must be copied over (from the base case) to the current case prior to any analysis.

- Return to the main SAPHIRE menu.

- Select **Generate** from the menu bar, and select the **Generate** button. To simply copy the base case data (the data just entered under the “Modify” option above) into the current case, do not mark any change sets.



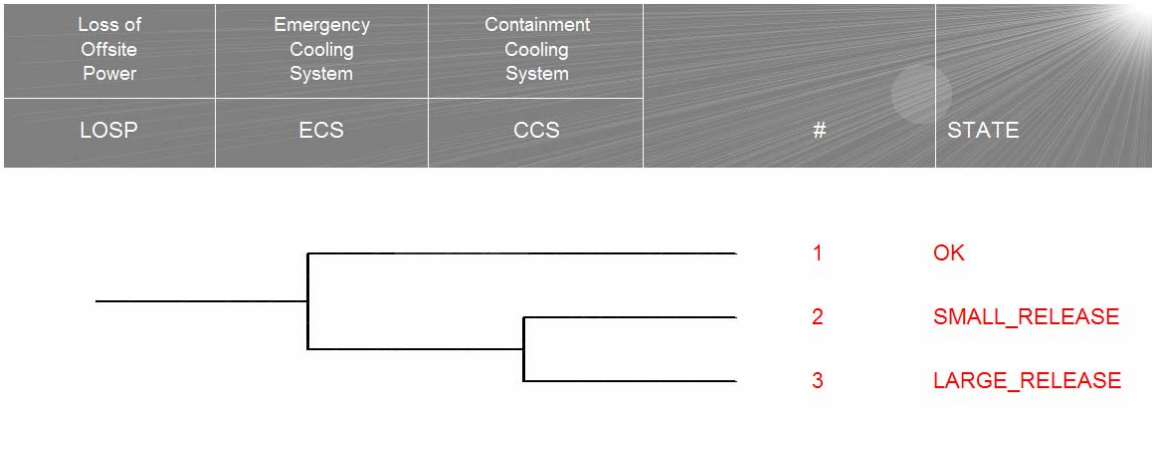
3.8. Reference

3-1. The SPAR Human Reliability Analysis Method, INEEL/EXT-02-01307.

NOTES

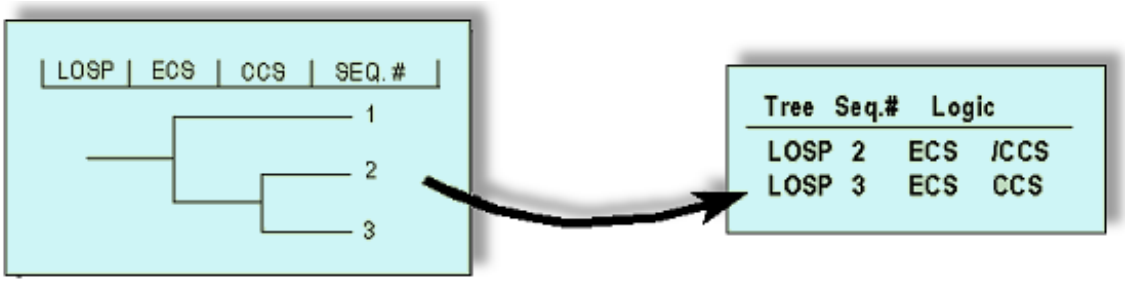
4 LINKING EVENT TREES

Section 4 introduces the rule editor in the Link Event Trees menu. The rule editor allows you to create rules that affect sequence generation. Typically, these rules are used to replace the default fault trees with either a substituted fault tree or a "split-fraction" event based on logical conditions that are specified in the rules.



4.1. Linking Event Trees

"Linking" event trees is the process of generating sequence logic using the event tree graphical files.



Menus and Options for Linking Event Tree Sequences

- Select **Event Tree** from the menu bar.
- The event trees are now displayed.
- Mark the event trees using the mask feature, or individually, using the mouse.
- Right-click to invoke the pop-up menu and select the **Link Trees** option.

If there are no “link rules” defined for the event tree, SAPHIRE simply constructs the sequence logic based upon the top events identified on the event tree graphic (as shown above). But, event tree linking rules allow us to control the logic for each sequence based upon predefined rules.

4.2. Introduction to the "Link Event Trees" Rule Editor

The rule editor provided in the Event Tree menu (**Event Tree ➔ Edit Rules**) allows rules to be written that are used when sequence logic is generated.

- These rules allow the user to replace one or more top events with a substituted top event based on the logical conditions dictated by the rule.
- These rules also allow the user to assign flag sets to sequences based on the logical conditions dictated by the rule.

Note that there are other rule editors in SAPHIRE that have different functions.

- ◊ Section 5 describes the *recovery rules* that are used to modify existing cut sets.
- ◊ Section 6 describes the *partitioning rules* that are used to bin cut sets into endstates on a "cut set by cut set" basis.

Although there are common features to all of these rule editors, they each have distinct functions and characteristics. As a convenient reference, we have listed all rule keywords (and usage) for linking rules, recovery rules, and partition rules in Appendix A.

"Link Event Trees" Rule Nomenclature and Structure

This rule editor is used when the "Link Trees" operation is being performed and the sequence logic is being created. If linking rules exist, the rule searches the event tree logic for the search criteria specified in the rule and replaces the default top event (just in the sequence logic, not on the graphic) by a new top event.

Symbols

|	Denotes a comment line	~	Operator for "not present"
*	Logical AND operator	+	Logical OR operator
/	Complement	( )	For grouping terms

Search Criteria

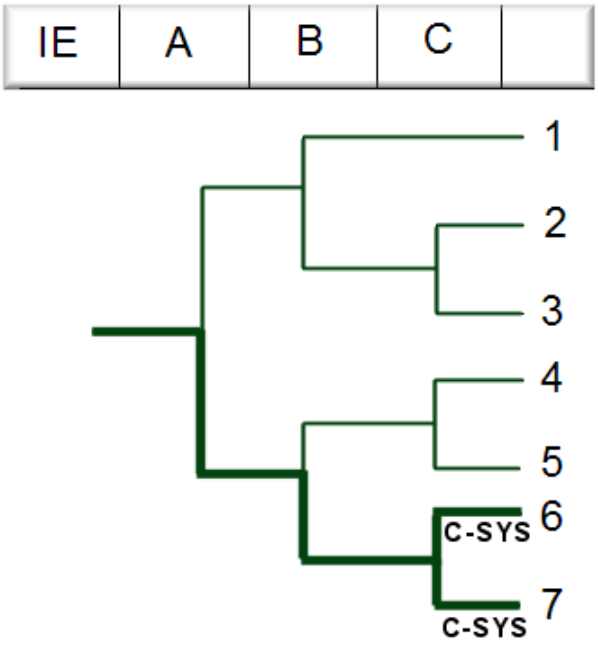
Examples are for an event tree with initiating event IE and top events A, B, and C.

init(IE)	Initiating event with the name IE
A	Failure of top event or fault tree A
IA	Success of top event or fault tree A (***I*** indicates complement)
~A	Failure of A not present (**~** indicates something is not present)
~(/A + A)	Success of A and failure of A not present (can be used to test for a "pass" condition)
A * B	Failure of A and of B occurs
(A + B) * C	Failure of A or B occurs and failure of C occurs
always	This pre-defined macro name means the criteria is always met.

Linking Rule Structure (Example 1 – If -Then)

| The "if-then" Rule Structure:
| This rule replaces C with C-SYS when
| A and B are both failed.
| (Only sequences 6 and 7 are affected
| by this rule)

```
if  A * B  then
  /C = C-SYS;
   C = C-SYS;
endif
```



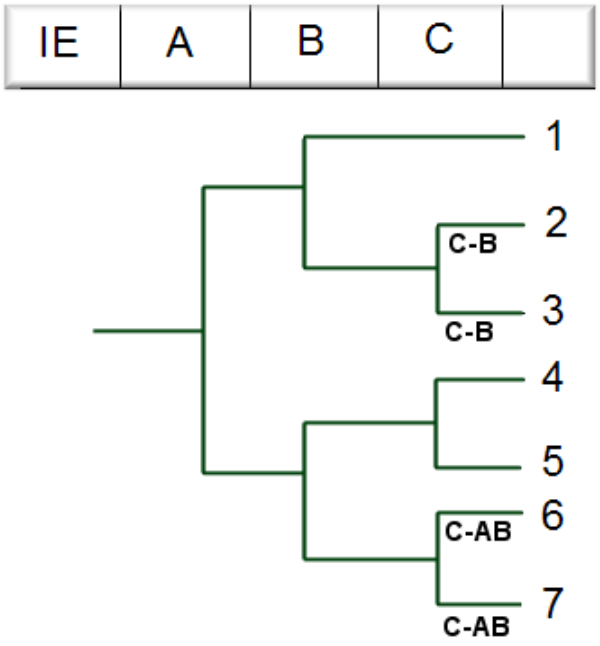
IMPORTANT REMINDERS:

- ◊ Each replacement line must end with a semicolon.
- ◊ There is no limit to the number of replacement lines that can be used in a rule.

Linking Rule Structure (Example 2 – if-then-elsif)

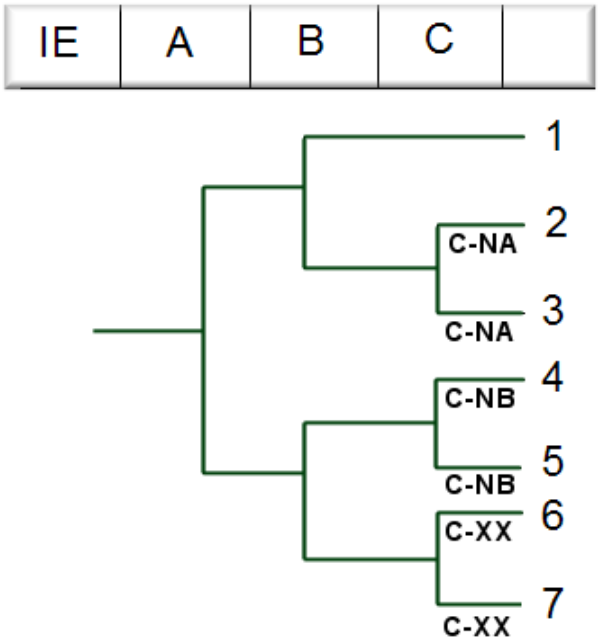
| The "if-then-elsif" Structure:
| This rule replaces C with C-AB if
| A and B are failed, and replaces C
| with C-B if only B is failed.
|

```
if A * B  then
  /C = C-AB;
   C = C-AB;
elsif  B  then
  /C = C-B;
   C = C-B;
endif
```



Linking Rule Structure (Example 3 – if-then-elsif-else)

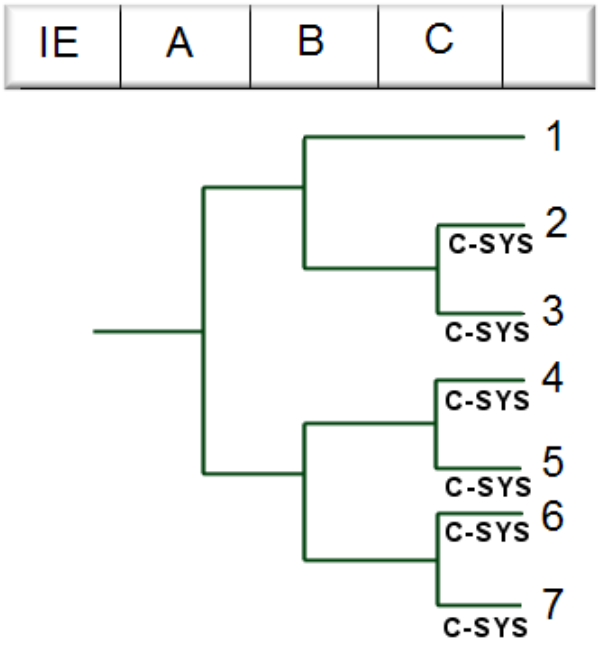
```
| The "if-then-elsif-else" Structure:
| This rule:
|   Replaces C with C-NA when A is
|   successful
|   Replaces C with C-NB if B is
|   successful
|   Replaces C with C-XX in any other
|   case
|
        if /A  then
          /C = C-NA;
           C = C-NA;
        elsif /B then
          /C = C-NB;
           C = C-NB;
        else
          /C = C-XX;
           C = C-XX;
        endif
```



Linking Rule Structure (Example 4 – always)

```
| The "if-always" Rule Structure:
| This rule replaces every occurrence
| of C with C-SYS.
|
| (Sequences 2 through 7 are
| affected)

        if always then
          /C = C-SYS;
           C = C-SYS;
        endif
```

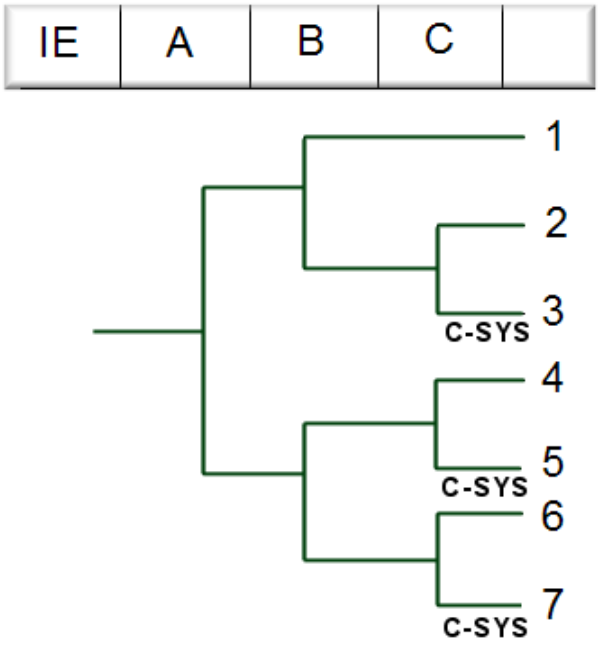


Linking Rule Structure (Example 5 – If –Then with wildcards)

| The "if-then" Rule Structure using
| wildcards:
| This rule replaces C with C-SYS
| when the initiating event occurs.
| (Sequences 3, 5 and 7 are affected)

```
if  “??”  then
  C = C-SYS;
endif
```

| The “??” finds the initiating event,
| but will key on any top of exactly
| two characters in length.



Only the First Substitution per Branch — In the ELSIF rule structure, only the first substitution that applies for every applicable branch is made. Subsequent possible substitutions are ignored.

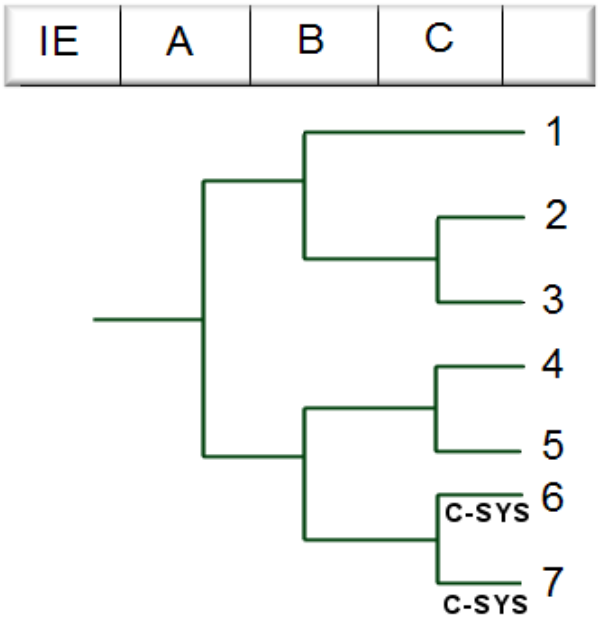
In fact, the "Event Tree Linking" rules as a whole work this way because only the first substitution for a branch is made. In other words, after a substitution has been assigned, no other rule will overwrite the substitution (this is by design). Consequently, the rules are set up such that the most restrictive (or, perhaps, most descriptive) rules will be evaluated first.

"Macro" Structures

Macros can streamline the development of complex rules. A macro is simply a statement to define a search criterion and assign a name to that search criterion. An example is provided below.

Linking Rule Structure (Example 6 – macros)

| Define a macro named AB-FAIL
AB-FAIL = A * B;
|
| Use the macro in a rule
if AB-FAIL then
/C = C-SYS;
C = C-SYS;
endif

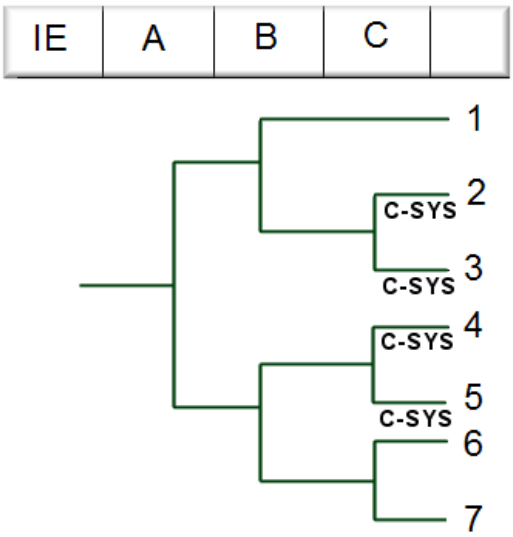


If you are creating a rule where the events in the macro do not occur, use the ~ (i.e., not present) symbol.

If looking for success events, do not "complement" the macro. Instead, complement the events of interest. For example, if looking for success of A, use A-MACRO = /A;. Do not try A-MACRO = A; if /A-MACRO then…

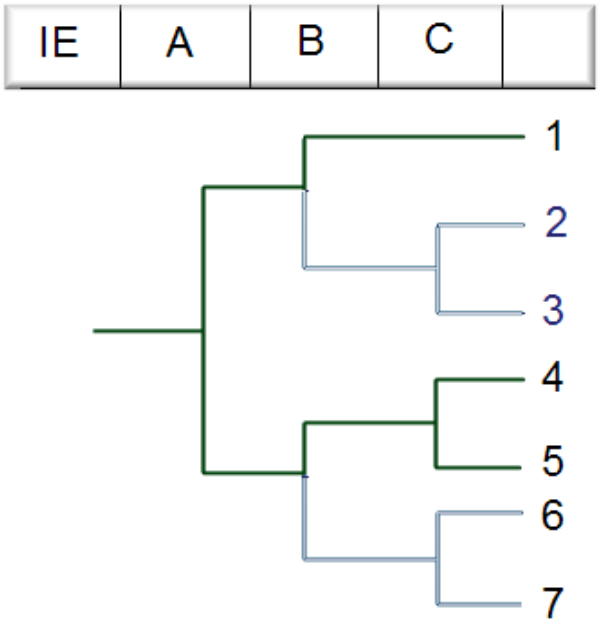
Linking Rule Structure (Example 7 – not found ~)

| Using ~macro as the search criteria:
| The rule applies when both A and B
| have not failed.
| Define a macro named AB-FAIL
AB-FAIL = A * B;
| Use the macro in a rule
if ~AB-FAIL then
/C = C-SYS;
C = C-SYS;
endif



Linking Rule Structure (Example 8 – ignoring sequences via the SKIP keyword)

| The "Skip" Structure:
| This rule provides the ability to
| “skip” sequences in the event
| tree logic.
|
| This rule “skips” C given the
| failure of B. (The sequences that
| meet the rule logic are
| not generated when the rule is
| applied, however, the sequence
| names (numbers) are left
| unchanged. Therefore, no sequence
| cut sets can be generated for skipped
|sequences.)
|




```
if B then
  /C = Skip(C);
   C = Skip(C);
endif
```

For this rule only sequences 1, 4, and 5 will be generated, since all sequences where B fails have been skipped.

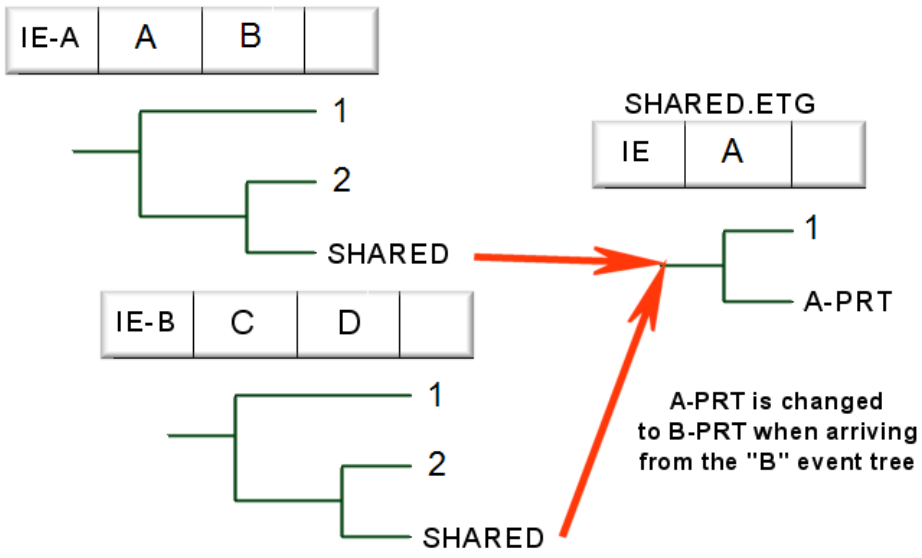
Remember: The Linking Rule Editor is a tool used to manipulate the sequence logic generated from the Event Tree during the Linking Process.

4.3. Changing Transfers Trees using Link Rules

The Link Rules may also be used to control the transfer process from one event tree to a sub tree. This ability brings in the use of a new keyword, **eventree()**. See the example below.

Linking Rule Structure (Example 9 – changing the transfer tree via EVENTREE)

```
|
|       This rule is for a transfer tree named SHARED.
|       The SHARED event tree is transferred to by two different event trees,
|       each having a unique initiating event. The first event tree has initiating
|       event IE-A, and after it transfers to SHARED, it should transfer to an event
|       tree named A-PRT.
|       The second event tree has initiating event IE-B, and after it transfers
|       to SHARED, it should transfer to an event tree named B-PRT.
|
|       The transfer name on the SHARED event tree graphic is A-PRT. This rule
|       changes the specified transfer event tree to B-PRT when the initiator is IE-B.
|
if init(IE-B) then
  eventree(A-PRT) = eventree(B-PRT);
endif
```



4.4. Event Tree Linking Rule Keywords and Nomenclature

Each of the “rules” in SAPHIRE (linking, recovery, and partition) has their own nomenclature. The table below lists the keywords available for linking rules.

Keyword or symbol	Definition	Example Usage

if then	Keyword that indicates a search criterion is being specified.	if "search criteria" then
		perform some action on the sequence;
		endif

Endif	Keyword that indicates the end of a particular rule.	if "search criteria" then
		perform some action on the sequence;
		endif

Else	Keyword that specifies some action to be taken if all the search criteria are not met. The else should be the last condition in the event tree linking rule.	if "search criteria" then
		perform some action on the sequence;
		else
		perform some other action on the sequence if search criteria not met;
		endif

Elsif	Keyword that specifies an alternative search criteria. Any number of elsifs can be used within an event tree linking rule.	if "search criteria" then
		perform some action on the sequence;
		elsif "2nd search criteria" then
		perform some other action on the sequence;
		elsif "3rd search criteria" then
		perform some other action on the sequence;
		endif

always	Keyword that indicates every fault tree top event satisfies the search criteria.	if always then
		perform some action on the sequence;
		endif

init( )	Keyword used in the search criteria to indicate that the sequence logic has a particular initiating event.	if init(INITIATOR-NAME) * "other search criteria if needed" then
		perform some action on the sequence;
		endif

~	Symbol used in the search criteria to indicate that a particular top event will not be in the sequence logic that is being tested.	if (~SEARCH-CRITERIA) * "other search criteria if needed" then ... The search criteria will be satisfied for all sequences that do not contain SEARCH-CRITERIA (and also contains the optional "other search criteria"). SEARCH-CRITERIA may be an initiating event, top event, or macro.

Keyword or symbol	Definition	Example Usage
/	Symbol used to represent a complemented event (i.e., the success of a fault tree).	if (/TOP EVENT) * "other search criteria" then The search criteria will be satisfied for all sequences that contain the complement of TOP EVENT (and also contains the optional "other search criteria").
|	Symbol used to represent a comment contained in the rules. Everything on a line to the right of this symbol will be ignored by the rule compiler.	| Place your comments here! | Note that blank lines are also permissible!
;	Symbol to indicate the end of a macro line or a line that modifies the sequence logic being evaluated.	| usage for a macro command MACRO-NAME = "search criteria" ; | usage for a sequence modification line FT = FT-1;
*	Symbol to indicate the logical AND command.	if SEARCH-CRITERIA1 * SEARCH-CRITERIA2 then The search criteria will be satisfied for all top events that match SEARCH-CRITERIA1 and SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be an initiating event, macro, or top event.
+	Symbol to indicate the logical OR command.	if SEARCH-CRITERIA1 + SEARCH-CRITERIA2 then The search criteria will be satisfied for all top events that match either SEARCH-CRITERIA1 or SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be an initiating event, macro, or top event.
( )	Symbols to indicate a specific grouping of items.	if (A + B) * (C + D) then The search criteria above would return all top events that contain: [A * C], [A * D], [B * C], or [B * D].

=	Keyword to indicate the substitution of one event tree top (i.e., fault tree) for another event.	if "search criteria" then
		ET-FT = ET-FT1;
		endif

endstate	Keyword to assign a sequence (based upon sequence logic) to a particular end state.	If “search criteria” then
		eventree(ET-NAME) = endstate(ES-NAME);
		endif

Keyword or symbol	Definition	Example Usage

eventree( )	Keyword to indicate a change in the sequence transfer name.	if "search criteria" then
		  eventree(ORIG-TRAN) = eventree(NEW-TRAN);
		endif

Skip( )	Keyword to indicate that a sequence meeting the search criteria will be “skipped” (i.e., not generated and will not show up in the database).	if "search criteria" then
		  ET-FT = Skip(ET-FT);
		endif

[ ]	Keyword to indicate the number of the event tree branch for multiple-split branch points. The first branch under the top branch is designated as 1. The second is designated as 2, etc.	if "search criteria" then
		 /ET-FT = NEW-TREE-NAME1;
		  ET-FT[1] = NEW-TREE-NAME2;
		  ET-FT[2] = NEW-TREE-NAME3;
		endif

MACRO	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 24 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
		if MACRO-NAME then
		  perform some action on each sequence;
		endif
		
		|Macros are only applicable in the particular |rule set where they appear. In other words,
		|you can not define a macro in event tree
		|”A” and expect to use it in event tree “B.”

4.5. Rules for Binary and Multiple-Split Branches

- Event tree branches are normally binary (one up, indicating success, and one down, indicating failure). But, in general, there may be more than two “splits” underneath a single top event.
- SAPHIRE addresses multiple branches by way of the event tree link rules. Several important modeling conventions are provided in the following example. The nomenclature for specifying a specific event tree branch under a top event is demonstrated.
- For binary branching, the success branch for a top is denoted with the complement symbol "**/**". SAPHIRE computes the probability for /A as P(/A) = 1 – P(A).
- For multiple-split branching, the failed branches are designated with the top event name and the branch number in brackets.

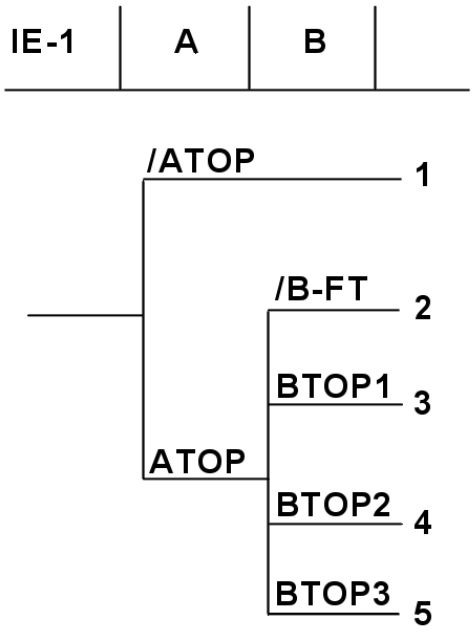
The success branch (the uppermost one) is assigned index [0], but is indicated by using the "/" nomenclature (see above). Then, the next branch (below the success branch) is assigned index [1], the next branch index [2], etc.

~RULES.TMP

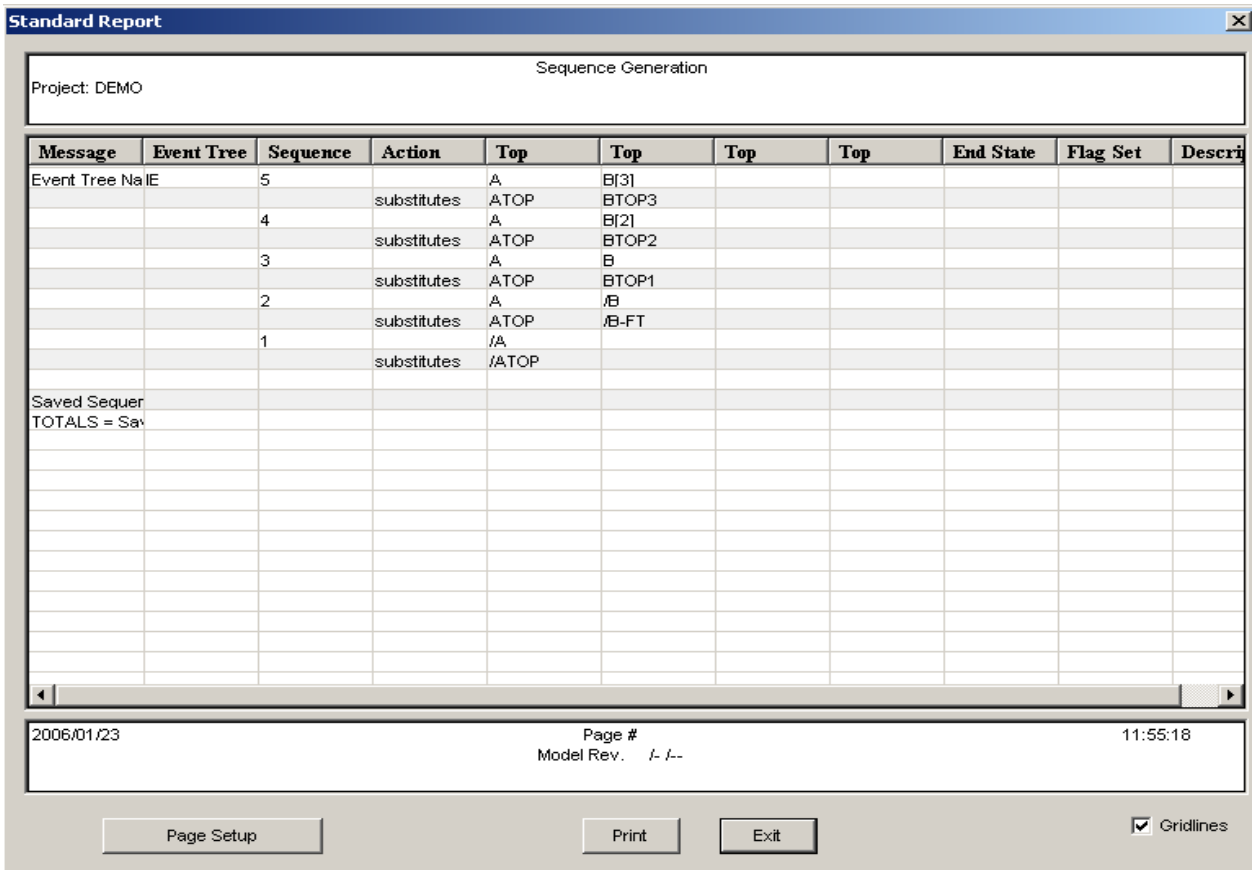
```
|RULES FOR EVENT TREE IE-1
|
|A rule for a 2-split branch
|Note: DO NOT specify "/ATOP"

if always then
 /A = ATOP;
  A = ATOP;
endif
|
|   A rule for multiple-split branch
|
if always then
 /B = B-FT;
 B[1] = BTOP1;
 B[2] = BTOP2;
 B[3] = BTOP3;
endif
<<eof>>
```

When you generate sequences using the event tree rules (**Event Tree ➔ Link Trees** option), the sequences are constructed as shown below.



The report that you will see out of SAPHIRE (if you print a logic report to the screen during the link process) will look like:



For multiple-split branches, you may want to construct a fault tree with the name that corresponds to the substituted success branch name (in our case, B-FT).

- ◊ The “success branch” fault tree would consist of the failed systems, BTOP1, BTOP2, and BTOP3 "ORed" together.
- ◊ Remember that SAPHIRE will automatically complement the fault tree when it solves the success branch (i.e., the uppermost branch).

To include a complemented event in event tree cut sets, you must specify the **Y** Process Flag (in the **Modify ➔ Basic Events** option) for the applicable top event. In this example, you would set the **Y** Process Flag for ATOP and B-FT.

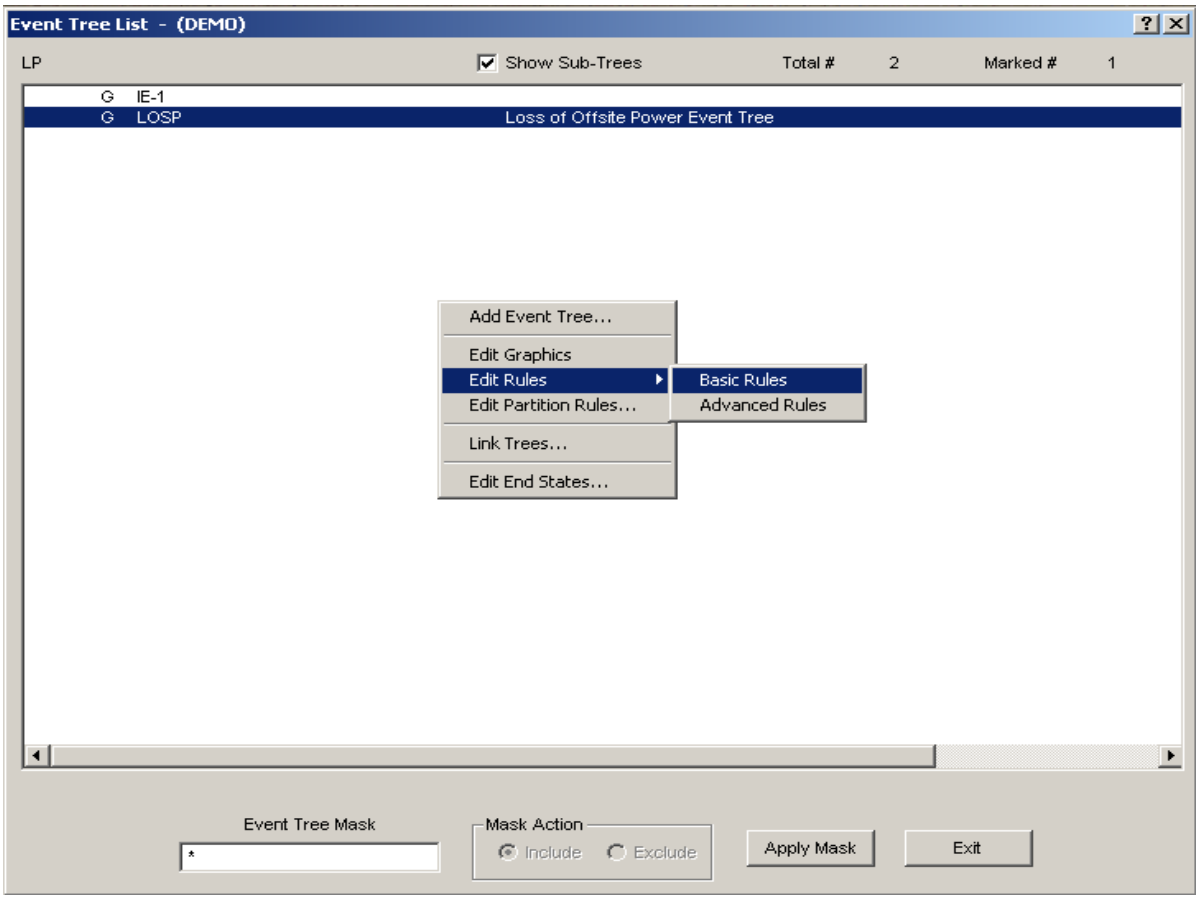
Then, to use the correct probability for this “success branch” fault tree, you will need to

- ◊ Set the B-FT event to a calculation type of “S” in order to tell SAPHIRE to use the fault tree cut sets for the event probability.
- ◊ Solve the fault tree (B-FT) prior to the sequence analysis and then generate data (**Generate ➔ Generate**).

4.6. The "Link Event Trees" Rule Editor

- Click the **Event Tree** button.

- To use the Rule Editor option, highlight an event tree, click the right mouse button, and select **Edit Rules**.



Rule Editor Options

- **Basic Rules** – This rule editor is a free formatting rule editor that is the one used to create and apply link event tree rules. This rule editor is the only one that will be discussed in the class.

- **Advanced Rules** – This rule editor is a very powerful and complex means to perform modifications to the event tree tops and other functions. This rule editor allows direct programming utilizing the same programming language that is used to program SAPHIRE (Modula).

- The link event tree rule text editor has many features in order to help the user.

  - ***File*:** The file drop down menu allows the user to save and exit the rule editor. The rules are automatically compiled when the rule editor is exited. If there is a problem in compiling the rule, SAPHIRE will not exit but go back to the rule editor and allow the user to fix the rule. The user can exit the rule editor without compiling if there are problems by selecting Exit a second time.

  - ***Edit*:** The edit drop down menu allows the user to cut, copy, and paste any portions of the existing rules.

  - ***Search*:** The search drop down menu allows the user the ability to search the rule for a specific top event to either locate the top event or replace the top event with another top event.

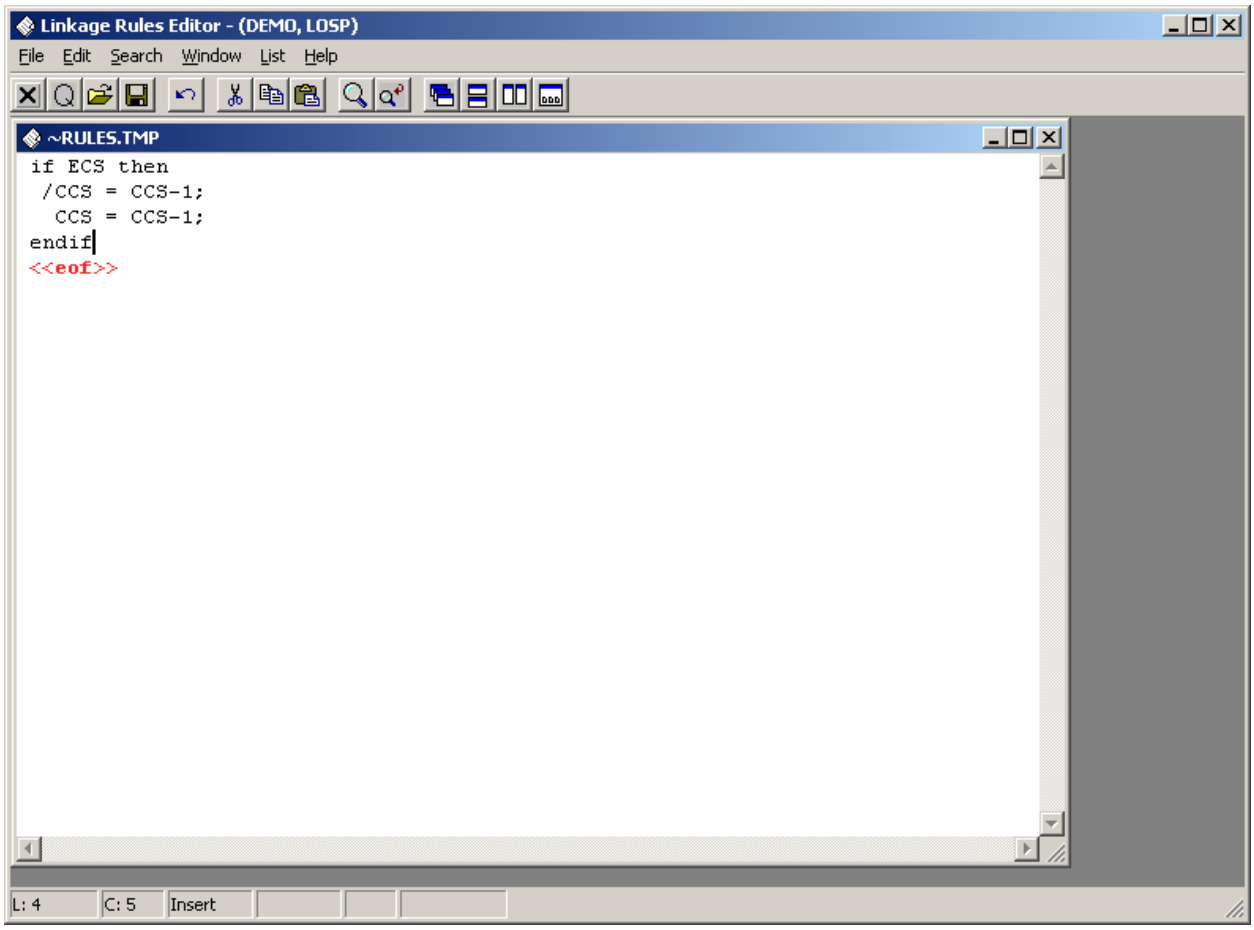
  - ***Window*:** The window drop down menu is similar to normal window features, but is not used when creating event tree linking rules since only one window is open at a time.

  - ***List*:** The list drop down menu will provide a list of existing fault trees, event trees, macros, flag sets, end states and initiators that can be selected and automatically inserted into the rule wherever the cursor is located. SAPHIRE does have an "add" feature in the editor that is accessed by right-clicking on the list and selecting the **Add** option.

  - ***Help*:** The help drop down menu will load up the help files associated with SAPHIRE.

Note: If a rule was created and the user left the rule editor without compiling the rule, the user needs to go back into the rule and edit the rule before SAPHIRE will compile the rule. This edit could be something as simple as adding and then removing a single space.

The Link Event Tree Rule Editor



NOTES

5 RECOVERY RULES

Section 5 presents the “Recovery Rule” editor. This editor allows you to create rules that affect existing cut sets in a “post-processing” fashion. The rule-based editor is available for both fault tree and sequence cut sets.

5.1. Recovery Rules Editor Introduction

The SAPHIRE Recovery Rules are "free-form" logic rules that allow for the alteration or deletion of fault tree or sequence cut sets.

> Although called "recovery rules," the recovery rules have evolved from the simple inclusion of recovery events into a powerful rule-based system for cut set manipulation.

The Recovery Rules can be used for probabilistic risk assessment techniques such as:

- ◊ The automated inclusion of sequence recovery events
- ◊ The inclusion of common-cause failure cut sets
- ◊ The elimination of mutually-exclusive events (e.g., impossible combinations of events).

The rules follow a format similar to the structure that is found in traditional programming languages (e.g., BASIC or PASCAL). As such, the ability exists to define "macros" and "if...then" type of structures.

The rules may be developed for a particular fault tree, all fault trees, a particular sequence, a single event tree, or all sequences.

Item	**Menu Button**	**Name of rule(s)**
Specific fault tree	Fault Tree	“Fault Tree” Rule Level
All fault trees	Fault Tree	“Project” Rule Level
Specific sequence	Sequence	“Sequence” Rule Level
Single event tree	Sequence	“Event Tree” Rule Level
All sequences	Sequence	“Project” Rule Level

The rules are entered in a free-form text editor within SAPHIRE. Note: The rules can be exported and loaded through MAR-D.

Use of the Recovery Rules could result in non-minimal cut sets. Thus, the typical steps in performing an analysis using the Recovery Rules are:

- ◊ Finalize logic models and data changes and then Generate changes
- ◊ Solve fault tree or sequence cut sets
- ◊ Apply Recovery Rules to applicable fault trees or sequences
- ◊ Perform a Cut Set Update to fault tree or sequence cut sets
- ◊ Perform Uncertainty analysis
- ◊ Display or report results

5.2. Recovery Rules Nomenclature and Structure

The Recovery Rule Editor looks and operates very similar to the Linking Rule Editor. However, the Recovery Rule Editor searches existing fault tree or sequence cut sets for cut sets matching the search criteria defined in the rule. The rule is used to modify the cut sets matching the search criteria.

Symbols

|	Denotes a comment line	~	Operator for "not present"
*	Logical AND operator	+	Logical OR operator
/	Complement	( )	Parentheses

Search Criteria Examples (for basic events X, Y, and Z)

Search Criteria	**Meaning of the Search Criteria**
X	Basic event X appears in the cut set
~X	Basic event X does not occur in the cut set
/X	Success of basic event X appears in the cut set
X * Y	Both basic events X and Y appear in the cut set
X + Y	Either basic event X or Y appear in the cut set
X*(Y + Z)	Either X and Y or X and Z appear in cut set in the cut set
~X*Y	Basic event Y does appear and basic event X does not appear
always	This pre-defined macro-name means the criteria is always met.
system(ECS)	Fault tree top event with name ECS

Recovery Rule Structure (Example 1 – if-then)

```
|  The "if-then" Rule Structure:
|  This rule adds a recovery action BUSREC  when electric bus B or C is failed

if  EL-BUS-B + EL-BUS-C then
  recovery = BUSREC;        |This keyword line must end with a semicolon.
endif
```

Recovery Rule Structure (Example 2 – if-then-elsif)

```
| The "if-then-elsif" Structure:
| This rule deletes the cut set if both diesel generators are out for maintenance.
| If the two DGs fail randomly, add a common cause event.

if (DG-1-MAINT * DG-2-MAINT) then
  DeleteRoot;
elsif  (DG-1-RAND * DG-2-RAND) then
  | Copy the original cut set, remove the two failure events, then add CC
  CopyRoot;
  DeleteEvent = DG-1-RAND;
  DeleteEvent = DG-2-RAND;
  AddEvent = DG-CCF-1AND2;
endif
```

Recovery Rule Structure (Example 3 – appending recovery actions)

The example below shows how the post-processing rules could be used to include recovery actions on specific cut sets via the Recovery Rules Option.

```
| The rule attaches the recovery action NRAC-12HR to every cut set for a particular
| sequence.
| This rule would probably be typed into the event tree sequence rule editor for the
| sequence of interest.
| A rule to apply NRAC-12HR recovery event to all cut sets in the sequence.
if always then
  recovery = NRAC-12HR;
endif
```

Recovery Rule Structure (Example 4 – mutually exclusive event removal)

The example below shows how the rules could be used to completely remove a particular cut set from the cut set list via the Recovery Rules Option.

- ◊ There may be instances where a cut set should be removed because the combination of basic events should not occur (e.g., two diesel generators out for maintenance at the same time).

```
| This rule could be placed in either (or both) the fault tree project rules or the event
| tree project rules.
| Define a macro to get those cut sets that have combinations of two motor driven
| pumps out for maintenance.
PUMPS-IN-MAINT = MDP-A-MAINT * MDP-B-MAINT;

| Search for the maintenance events and then delete cut set.
if PUMPS-IN-MAINT then
  | Delete the cut set
  DeleteRoot;
endif
```

Recovery Rule Structure (Example 5 – including common-cause failure events)

The example below shows how the rules could be used to add common-cause events to the cut sets via the Recovery Rules Option.

- ◊ The usefulness of the Recovery Rules for common-cause failure modeling is limited by the fact that the cut sets containing the independent random failures must exist for the search criteria to work.

- ◊ If a probability truncation is specified when generating fault tree or sequence cut sets, the independent failure cut sets may be truncated.

```
| The search criterion identifies the failure combination of two auxiliary feedwater
| pumps.
| If these two basic events are found in a cut set then a new cut set will be created that
| replaces the independent failures of the two pumps with a single common-cause basic
| event.
```

| This rule could be placed in either (or both) the fault tree project rules or the event tree
| project rules.

| Define a macro to only pick up those cut sets that
| have combinations of AFW-PUMP-A and AFW-PUMP-B.
CCF-AFW-PUMPS = AFW-PUMP-A * AFW-PUMP-B;

| Search for the AFW pump basic events and make a new
| cut set with the CCF event.
```
if CCF-AFW-PUMPS then
  | First make a copy of the original cut set
  CopyRoot;
  | Now remove the two independent failure events
  DeleteEvent = AFW-PUMP-A;
  DeleteEvent = AFW-PUMP-B;
  | Now add the CCF event
  AddEvent = AFW-PUMP-CCF;
endif
```

Recovery Rule Structure (Example 6 – use of top events)

The example below shows how the rules could be used to search top events and apply recovery basic events via the Recovery Rules Option.

```
| The search criterion identifies the failure of top event CCS and applies a
| recovery event to all cut sets in the sequence(s).

if system(CCS) then
  recovery = recover-CCS;
endif
```

5.3. Recovery Rule Keywords and Nomenclature

Each of the “rules” in SAPHIRE (linking, recovery, and partition) has their own nomenclature. The table below lists the keywords available for recovery rules.

Keyword or symbol	Definition	Usage

if then	Keyword that indicates search criteria is being specified.	if "search criteria" then
		perform some action on each cut set;
		endif

endif	Keyword that indicates the end of a particular rule.	if "search criteria" then
		perform some action on each cut set;
		endif

else	Keyword that specifies some action to be taken if all the search criteria are not met. The else should be the last condition in the recovery rule.	if "search criteria" then
		perform some action on each cut set;
		else
		perform some other action on each cut set not meeting the search criteria
		endif

elsif	Keyword that specifies an alternative search criteria. Any number of elsifs can be used within a recovery rule.	if "search criteria" then
		perform some action on each cut set;
		elsif "2nd search criteria" then
		perform some other action on each cut set;
		elsif "3rd search criteria" then
		perform some other action on each cut set;
		endif

always	Keyword that indicates that every cut set that is being evaluated satisfies the search criteria.	if always then
		perform some action on each cut set;
		endif

init( )	Keyword used in the search criteria to indicate that a sequence cut set has a particular initiating event.	if init(INITIATOR-NAME) * "other search criteria if needed" then
		perform some action on each cut set;
		endif

~	Symbol used in the search criteria to indicate that a particular event will not be in the cut set that is being evaluated.	if (~SEARCH-CRITERIA) * "other search criteria if needed" then ... The search criteria will be satisfied for all cut sets that do not contain SEARCH-CRITERIA (and also contains the optional "other search criteria"). SEARCH-CRITERIA may be an initiating event, basic event, macro, or logic expression.

Keyword or symbol	Definition	Usage
/	Symbol used to represent a complemented event (i.e., the success of a failure basic event).	if (/BASIC-EVENT) * "other search criteria" then The search criteria will be satisfied for all cut sets that contain the complement of BASIC-EVENT (and also contains the optional "other search criteria").
|	Symbol used to represent a comment contained in the rules. Everything on a line to the right of this symbol will be ignored by the rule compiler.	| Place your comments here! | Note that blank lines are also permissible!
;	Symbol to indicate the end of a macro line or a line that modifies the cut set being evaluated.	| usage for a macro command MACRO-NAME = "search criteria" ; | usage for a cut set modification line recovery = RECOVERY-EVENT ;
*	Symbol to indicate the logical AND command.	if SEARCH-CRITERIA1 * SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match SEARCH-CRITERIA1 and SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be an initiating event, basic event, macro, or logic expression.
+	Symbol to indicate the logical OR command.	if SEARCH-CRITERIA1 + SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match either SEARCH-CRITERIA1 or SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be an initiating event, basic event, macro, or logic expression.
( )	Symbols to indicate a specific grouping of items.	if (A + B) * (C + D) then The search criteria above would return all cut sets that contain: [A * C], [A * D], [B * C], or [B * D].

system( )	Keyword used in the search criteria to indicate that a fault tree contributes to the existence of the cut set that is being evaluated.	if system(ECS) then
		perform some action on each cut set
		endif

Keyword or symbol	Definition	Usage

Recovery =	Keyword that indicates that a recovery event is going to be added to the cut set being evaluated (SAPHIRE keeps record of all recovery events).	if "search criteria" then
		  recovery = NAME-OF-RECOVERY;
		endif

AddEvent =	Keyword that indicates that an event will be added to the cut set being evaluated.	if "search criteria" then
		  AddEvent = EVENT-NAME;
		endif

DeleteEvent=	Keyword that indicates that an event will be deleted from the cut set being evaluated.	if "search criteria" then
		  DeleteEvent = EVENT-NAME;
		endif

NewCutset;	Keyword that indicates that a new, empty cut set will be added to the list of cut sets. This new cut set then becomes the cut set that is being evaluated.	if "search criteria" then
		  NewCutset;
		  now make additions to the empty cut set...
		endif

DeleteRoot;	Keyword that indicates that the original cut set (i.e., that cut set that satisfied the search criteria) will be deleted.	if "search criteria" then
		  DeleteRoot;
		endif

CopyCutset;	Keyword that indicates that the cut set being evaluated will be copied and added to the list of cut sets. This copied cut set then becomes the cut set that is being evaluated.	if "search criteria" then
		  CopyCutset;
		  now make modification to a copy of the cut set...
		endif

CopyRoot;	Keyword that indicates that the original cut set (i.e., that cut set that satisfied the search criteria) will be copied. This copied cut set will then become the cut set that is being evaluated.	if "search criteria" then
		  CopyRoot;
		  now make modifications to a copy of the original cut set...
		endif

MACRO	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 16 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
		if MACRO-NAME "and other search criteria" then
		  perform some action on each cut set...;
		endif
		
		|Macros are only applicable in the |particular rule they are entered into

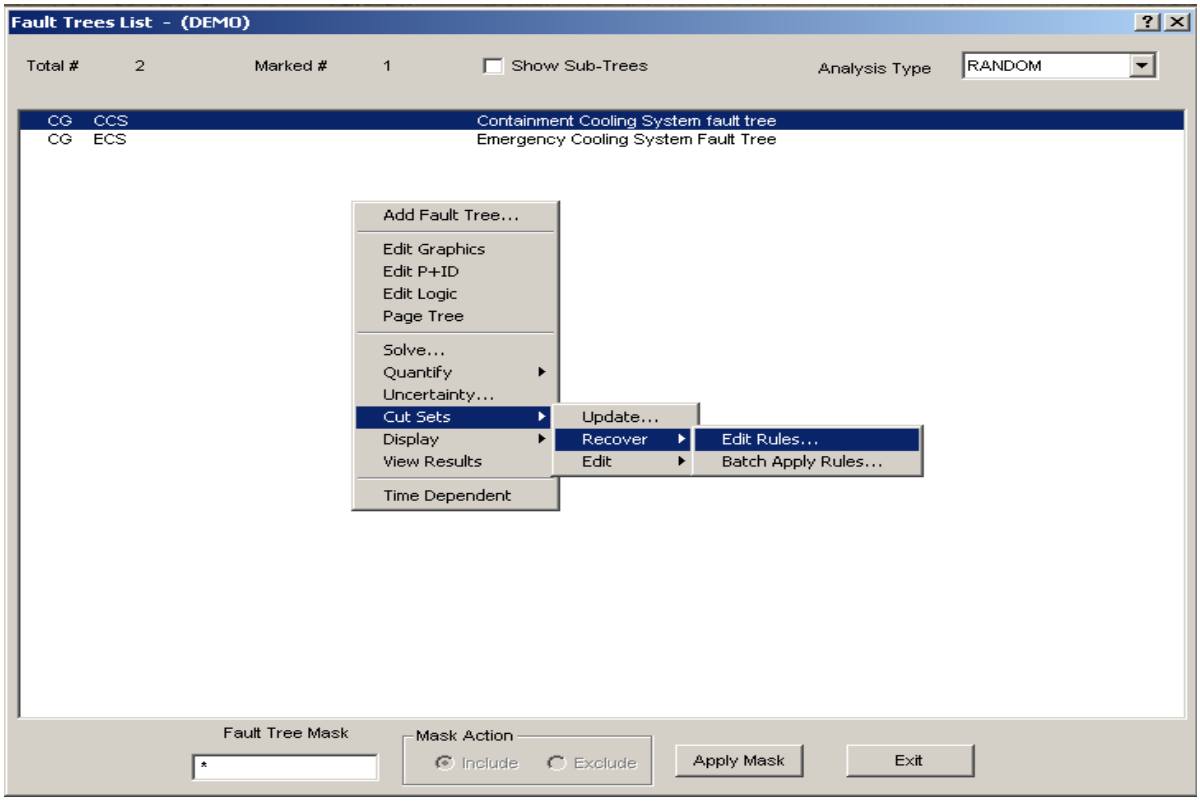
Remember: The Recovery Rule Editor is a tool used to manipulate cut sets. It does this manipulation by searching on pre-existing fault tree or sequence cut sets and looks for cut sets that meet the search criteria.

5.4. The Recovery Rules Editor

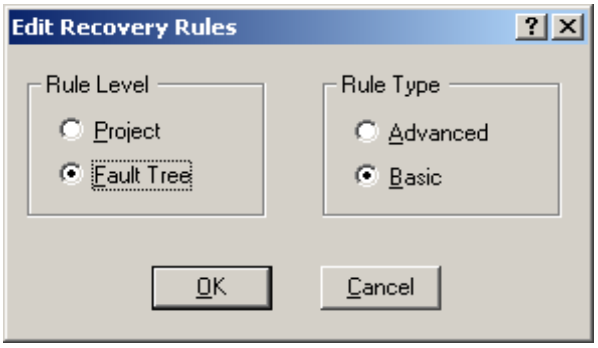
FAULT TREE RULES

♦ To create or edit the FAULT TREE Recovery Rules, click the **Fault Tree** button.

◊ To create or edit the Recovery Rules for a *specific* fault tree or all fault trees, highlight the specific fault tree and right click the mouse to get the pop up menu and select **Cut Sets → Recover → Edit Rules**.

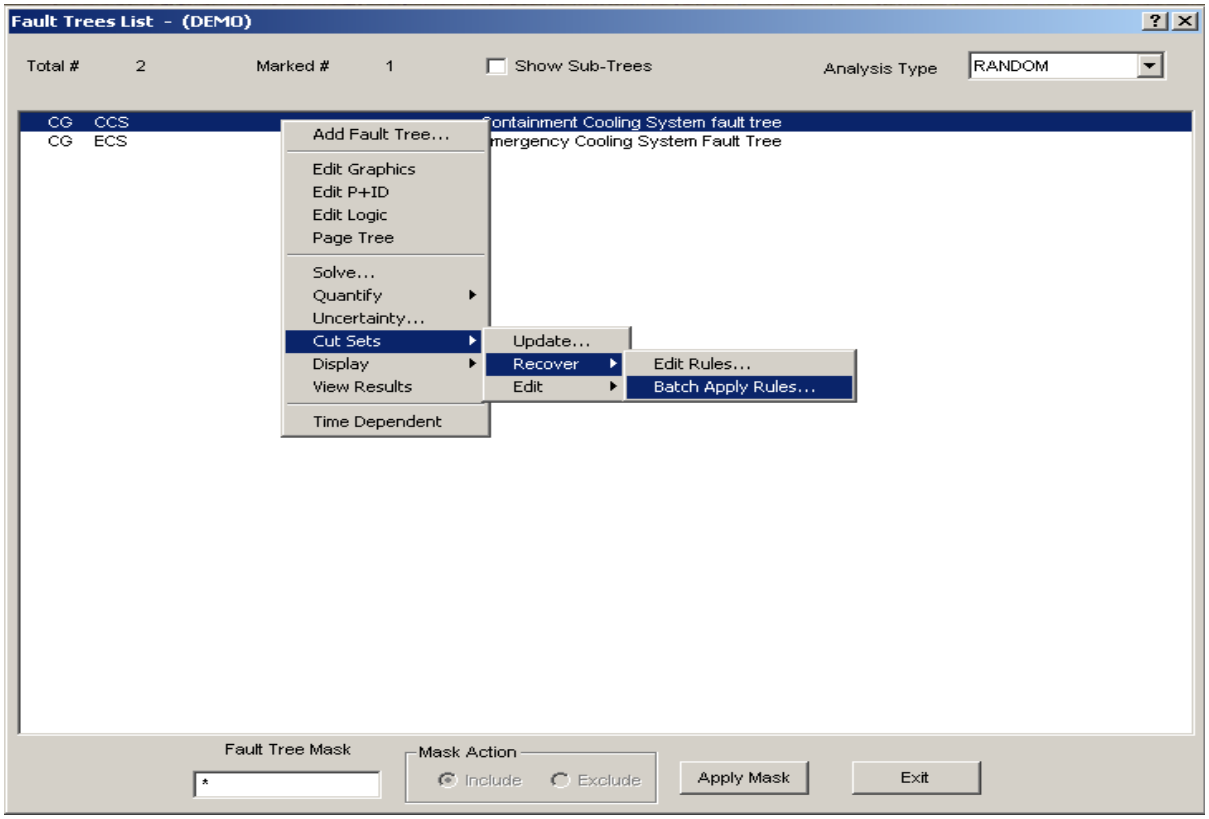


◊ A new option box will be displayed which allows the creation of Recovery Rules to either the highlighted fault tree (**Fault Tree**) or all fault trees (**Project**). (Again, use only the **Basic** Rule type)



♦ To apply the fault tree Recovery Rules, click the **Fault Tree** button.

◊ To apply the Recovery Rules for a *specific* fault tree or all fault trees, highlight the specific fault tree and right click the mouse to get the pop up menu and select **Cut Sets → Recover → Batch Apply Rules**.

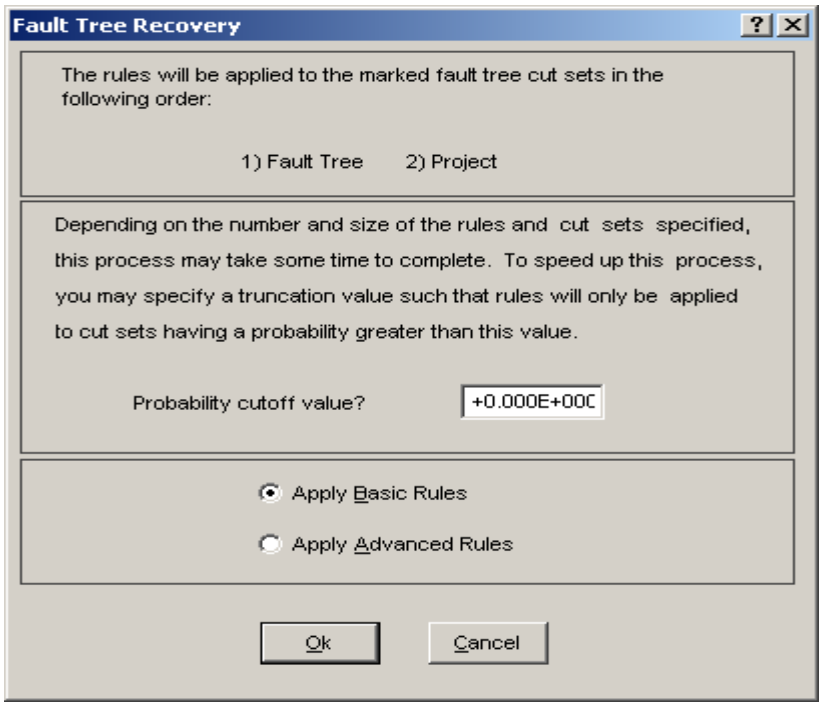


♦ The **Batch Apply Rules** option (the next option box) allows the user to apply the fault tree and project rules to the marked or highlighted fault trees.

◊ The user can specify a probability cutoff if needed
- Zero is the default value, which implies that no truncation is to be used.

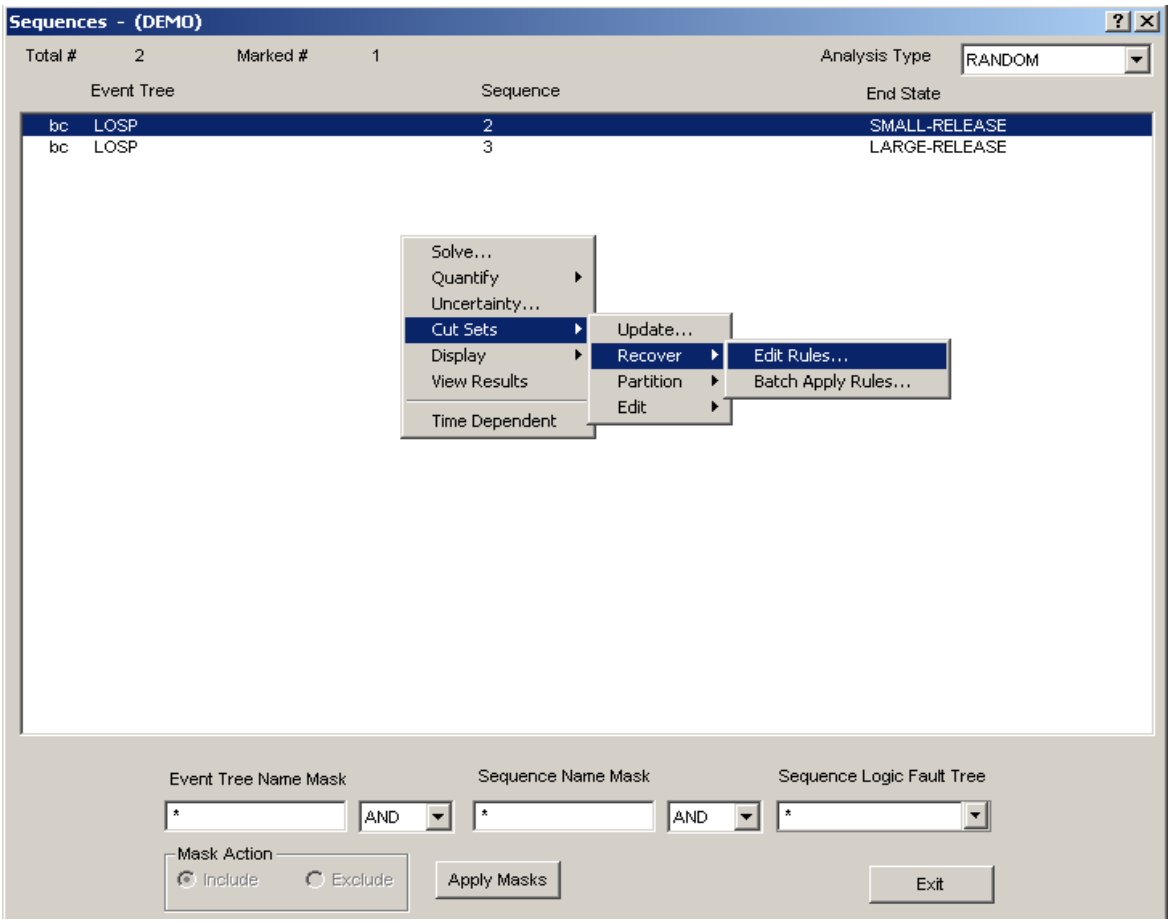
◊ SAPHIRE will apply the fault tree specific rules first then apply the project rules to the cut sets of the highlighted fault trees. Therefore, the rules need to be created appropriately (i.e., generic to specific).

*Note that when solving for cut sets, there is an option to "automatically apply recovery rules." Checking this option performs the **Batch** option shown above. If you use the "automatic" option, you do not need to reapply the recovery rules.*

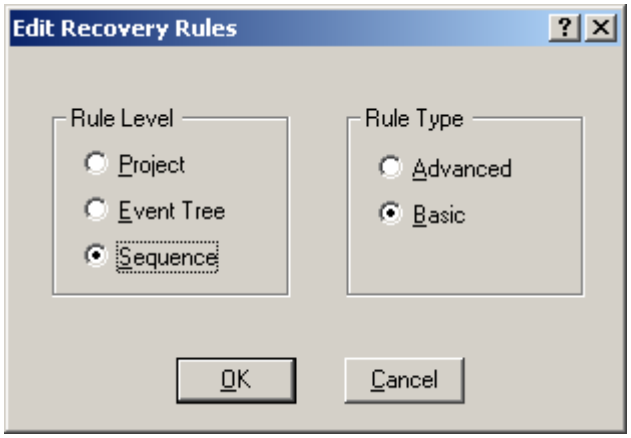


SEQUENCE RULES

- To create or edit the SEQUENCE Recovery Rules, click the **Sequence** button.
  - To create or edit the Recovery Rules for a *specific* sequence, event tree or all sequences, highlight the specific sequence and right click the mouse to get the pop up menu and select **Cut Sets → Recover → Edit Rules**.

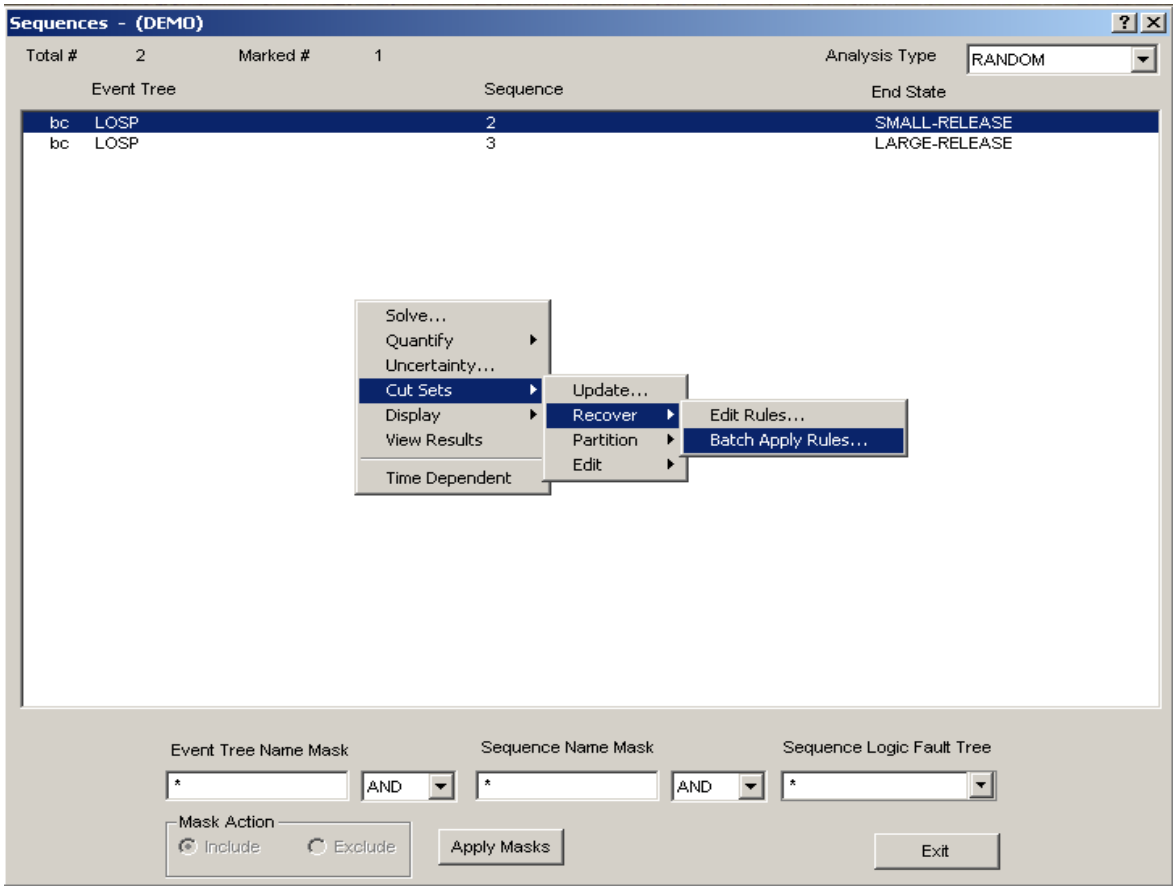


◊ A new option box will be displayed which allows the creation or editing of Recovery Rules to either the highlighted sequence (**Sequence**), event tree which is based on the highlighted sequence (**Event Tree**), or all sequences (**Project**). (Again, use only the **Basic** Rule type)



♦ To apply the sequence Recovery Rules, click the **Sequence** button.

◊ To apply the Recovery Rules for a *specific* sequence, event tree or all sequences, highlight the specific sequence and right click the mouse to get the pop up menu and select **Cut Sets → Recover → Batch Apply Rules**.

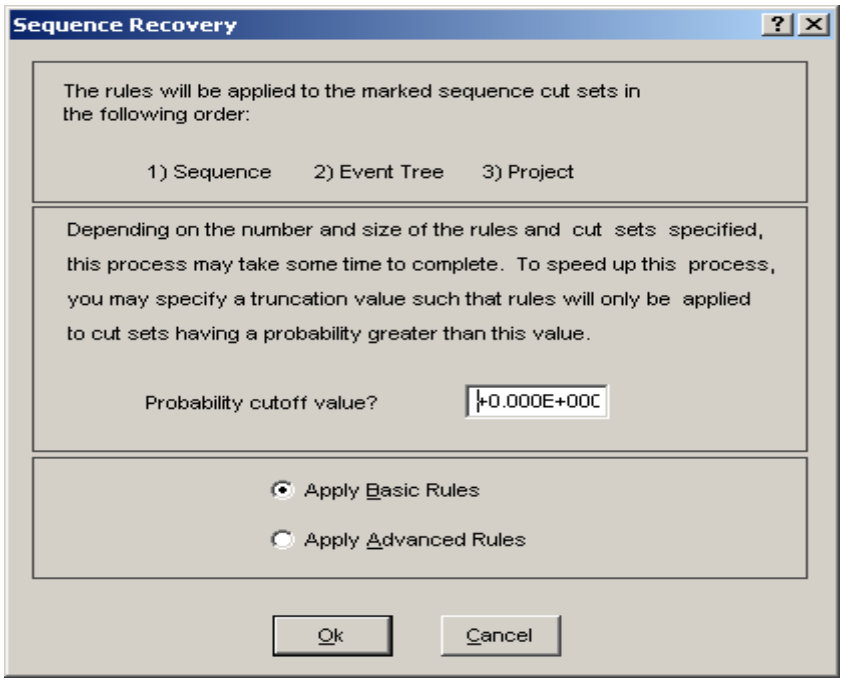


♦ The **Batch Apply Rules** option (the next option box) allows the user to apply the sequence, event tree and project rules to the marked or highlighted sequences.

◊ The user can specify a probability cutoff if needed
- Zero is the default value, which implies that no truncation is to be used.

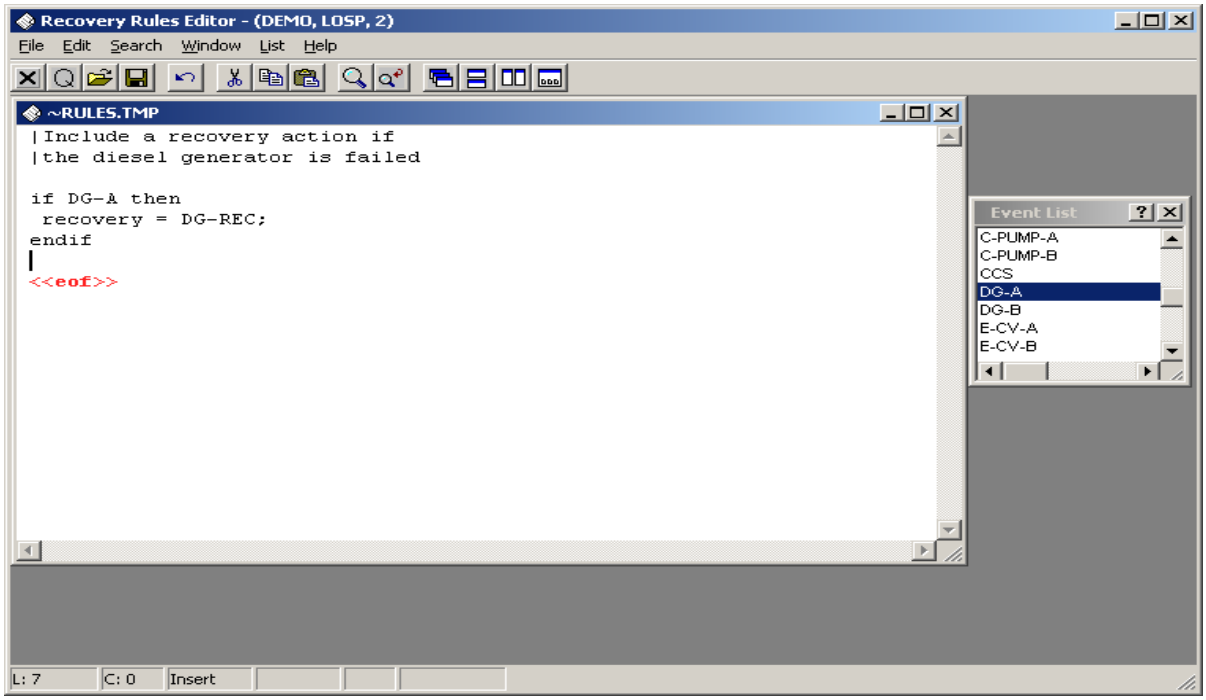
◊ SAPHIRE will apply the sequence specific rules first then apply the event tree specific rules and then finally the project rules to the cut sets of the highlighted sequences. Therefore, the rules need to be created appropriately (i.e., generic to specific).

Note that when solving for cut sets, there is an option to "automatically apply recovery rules." Checking this option performs the ***Batch*** *option shown above. If you use the "automatic" option, you do not need to reapply the recovery rules.*



Recovery Rules Editor Options

Type the rules in the editor. The Recovery Rule Editor window is shown in the figure below.



The rule editor has many features in order to help the user.

- **File**: The file drop down menu allows the user to save and exit the rule editor. The rules are automatically compiled when the rule editor is exited.

  If there is a problem in compiling the rule, SAPHIRE will not exit but go back to the rule editor and allow the user to fix the rule. The user can exit the rule editor without compiling if there are problems (e.g., the basic event is not in the database) by selecting *Exit* a second time.

- **Edit**: The edit drop down menu allows the user to cut, copy, and paste any portions of the existing rules.

- **Search**: The search drop down menu allows the user the ability to search the rule for a specific basic event to either locate the basic event or replace the basic event with another basic event.

- **Window**: The window drop down menu is similar to normal window features but is not used when creating “post-processing” rules.

- **List**: The list drop down menu will provide a list of existing basic events, initiating events, macros, and recovery events that can be selected and automatically inserted into the rule wherever the cursor is located. SAPHIRE does have an “add” feature in the editor that is accessed by right-clicking on the list and selecting the **Add** option.

- **Help**: The help drop down menu will load up the help files associated with SAPHIRE.

Note that if a user created a rule and exited the editor without compiling the rule, the user must return to the rule and make a change to the rule before SAPHIRE will compile the rule. This change could be as simple as adding and then removing a single space.

5.5. A "Complicated" Recovery Rule Example

Now, an example is presented that utilizes several of the recovery rule keywords. In this example, it is assumed that only a single cut set matches the search criteria. This cut set has a single basic event (A) and is called the *Root* cut set. The overall rule looks like:

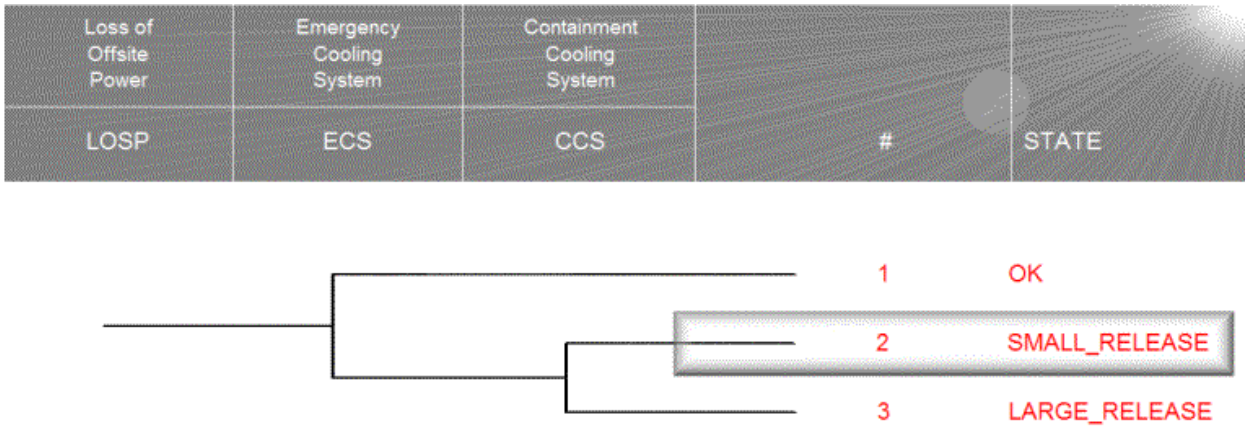
If A then
AddEvent = B;
NewCutset;
AddEvent = C;
DeleteRoot;
CopyCutset;
AddEvent = D;
endif

The outcome of applying this recovery rule is shown in the following table.

Step	Applied Keyword	Resulting cut set(s)	Comment
1	AddEvent = B;	(1) A * B	Event B is attached to the "currently-evaluated" cut set.
2	NewCutset;	(1) A * B (2) *blank*	A new blank cut set is included in the list of cut sets. This new cut set now becomes the "currently-evaluated" cut set.
3	AddEvent = C;	(1) A * B (2) C	Event C is attached to the "currently-evaluated" cut set.
4	DeleteRoot;	(1) C	The Root cut set is removed.
5	CopyCutset;	(1) C (2) C	A new cut set is included in the list of cut sets that is a duplicate of the old "currently-evaluated" cut set. Note that this is different than the "CopyRoot" command which would have included a new cut set with event A in the cut set (i.e., the starting cut set).
6	AddEvent = D;	(1) C (2) C * D	Event D is attached to the "currently-evaluated" cut set.

6 END STATE ANALYSIS

Section 6 describes the end state analysis features in SAPHIRE. Cut sets derived by analyzing event tree sequences can be grouped into end states by specifying the sequence end state on the event tree or by developing end state partitioning rules. Both approaches are described in this section.



6.1. End State Analysis Approaches

End state analysis is simply the grouping of cut sets generated from event tree sequences in ways that are useful to the analyst. Cut sets grouped by end state can be conveniently displayed and reported, and end state uncertainty analysis can be performed. There are two basic approaches provided in SAPHIRE to group cut sets into end states:

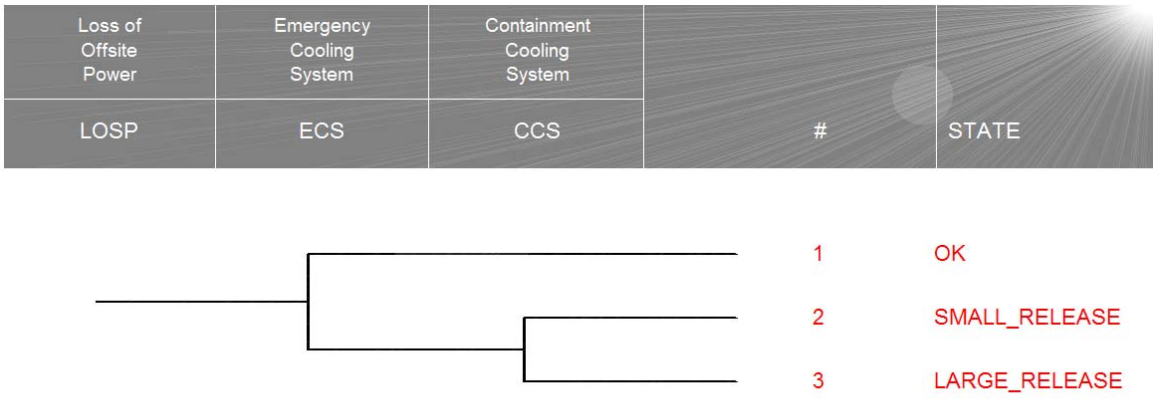
1. End state analysis by specifying sequence end states — In this approach, the end state is specified for each event tree sequence in the graphical file (or in the **Event Tree → Edit End States** option).

2. End state analysis using partitioning rules — In this approach, user-defined rules are used to assign end states. Features include:

   - Application of rules to the entire database project, event trees, and/or sequences.
   - Cut sets from the same sequence can be grouped into separate end states.
   - End state names can be creating using a "layering process" that allows character substitutions.

6.2. End States by Specifying Sequence End States

Two accident end states (SMALL_RELEASE and LARGE_RELEASE) are specified on the LOSP event tree. Note that OK end states are ignored.

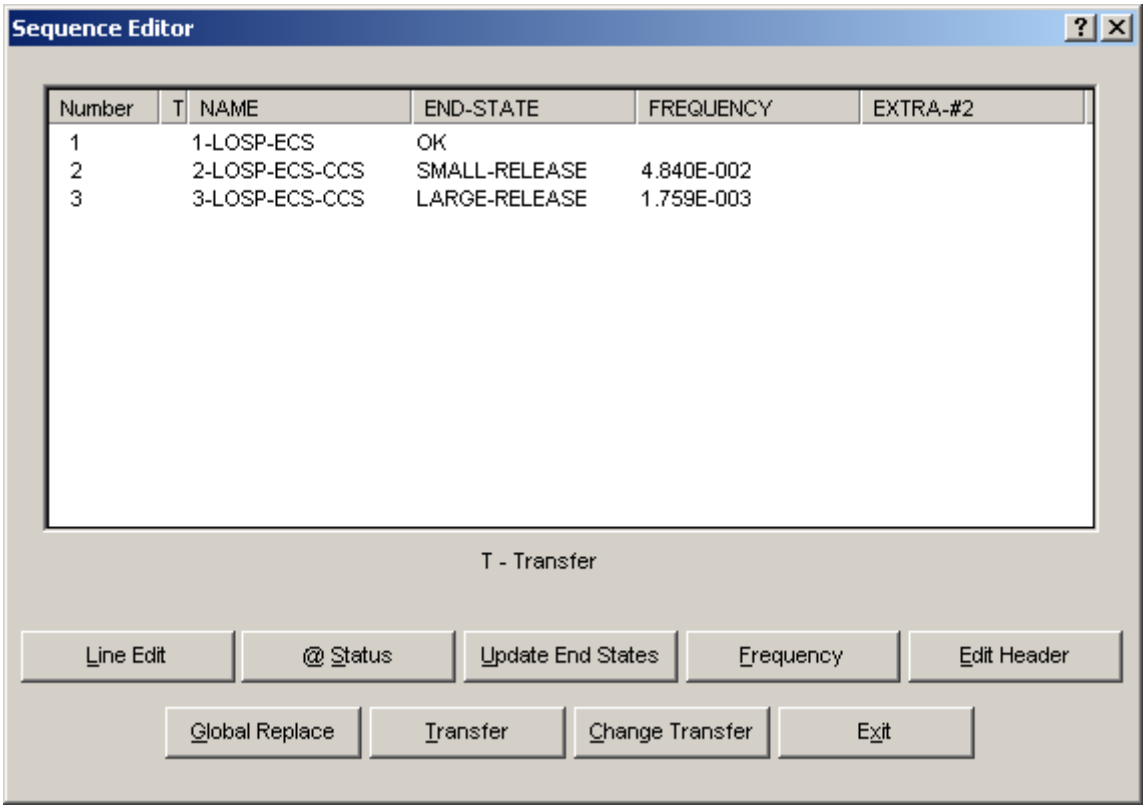


The end states specified in the "ENDSTATE" column automatically become end states in the database.

The event tree graphical editor and the sequence editor provide two ways of editing the same data file. The end state name can be edited from either editor.

◊ The graphical event tree editor was used in the Basics course.

To use the sequence editor, select the **Event Tree → Edit End States** option.



◊ The **Line Edit** option will allow for modification to the end state.

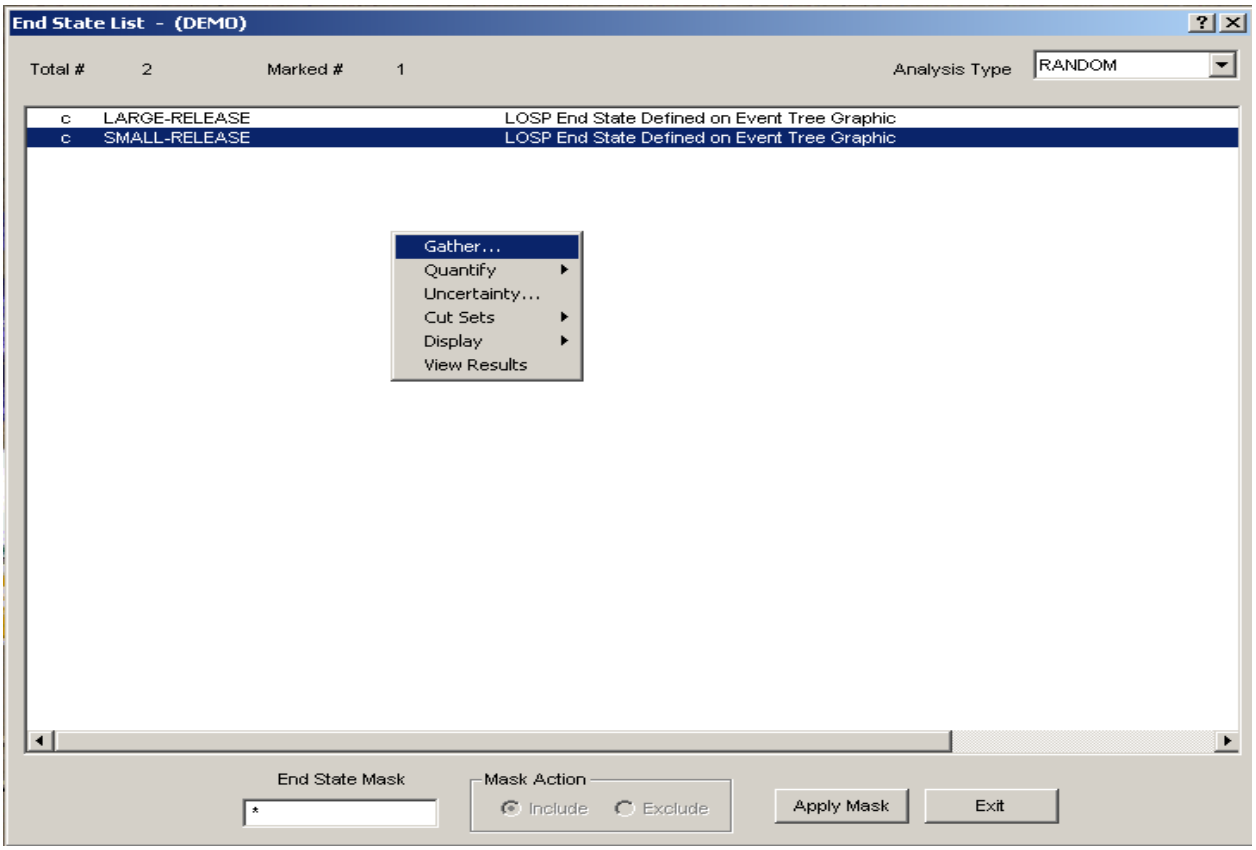
◊ The **Frequency** option updates the sequence frequency value that may be displayed on the event tree graphic.

To gather end state cut sets:

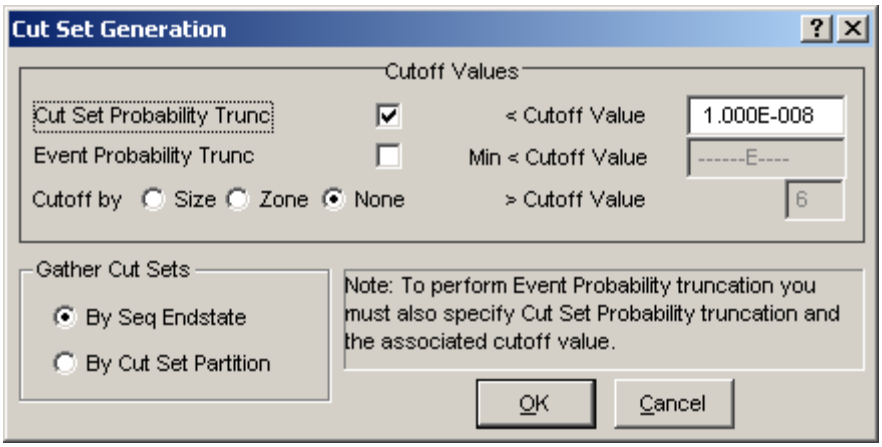
1. *Sequence* cut sets must be generated (**Sequence** → **Solve** option).

2. Gather end state cut sets by entering the end state option (click the **End State** button).

3. Highlight, the end state, right click the mouse, and select **Gather**.



When you select the Gather option, the screen shown below will appear. Enter the desired truncation on the screen, and click **OK** to begin gathering end state cut sets.



"Gather" Analysis Options:

Cut Set Probability Truncation? If you check this box, then those cut sets below the value in the "< Cutoff Value" field will not be retained.

Event Probability Truncation? If you check this box, then you must also check the box for the Cut Set Probability Truncation. This option will retain cut sets comprised of basic events that are each above the "Min < Cutoff Value" even if the entire cut set is below the Cut Set Probability Truncation Cutoff Value.

Cut Set Size Truncation? If you click the Size radio button, then cut sets having more events than specified in the "> Cutoff Value" field will not be kept. If you click the None radio button, then the number of events in a cut set will not affect whether the cut set is kept or discarded. If you click the Zone radio button, then cut sets having more events with the Process Flag = Z than specified in the "> Cutoff Value" field will not be kept.

Gather Cut Sets By Sequence Endstate? If you click the "By Seq Endstate" radio button, then the end states specified on the event tree sequences (e.g., via the graphics) are gathered. If you click the "By Cut Set Partition" radio button, then the end states created when end state partitioning rules were applied are gathered.

Description of other end state analysis options

Gather - This option gathers existing cut sets (generated from the event tree sequences). The end state frequency is quantified using the minimal cut set upper bound approximation. (Non-minimal cut sets are eliminated within each end state.)

Cut Sets → Update - This option uses the existing end state cut sets. Non-minimal cut sets are eliminated and the end state frequency is quantified using the minimal cut set upper bound approximation.

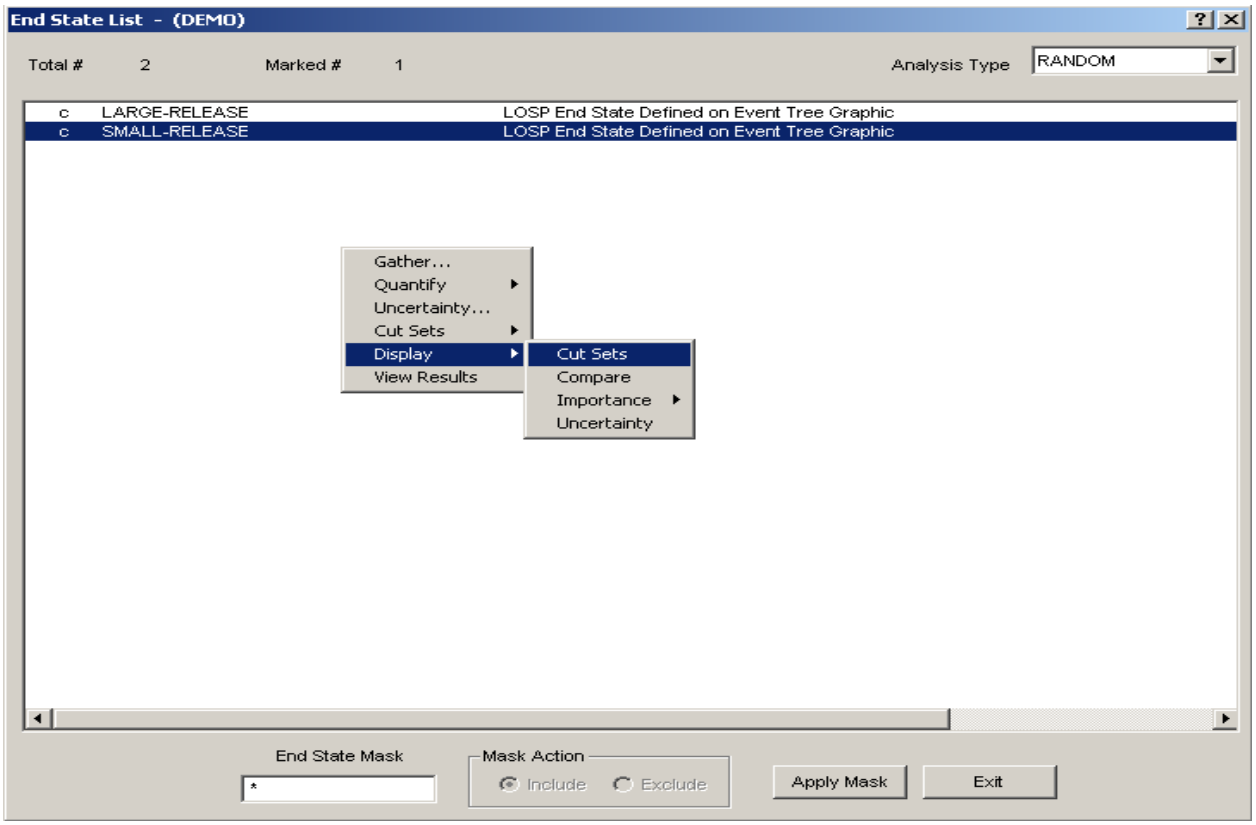
Quantify - This option uses the existing end state cut sets and requantifies the end state frequencies using the minimal cut set upper bound approximation (or rare event). This option is designed to quickly requantify the cut sets when data changes have been made.

Uncertainty Analysis - Performs Monte Carlo or Latin Hypercube uncertainty analysis for the selected end states.

Analysis Type - Select the RANDOM analysis type for the material covered in this class. The other analysis types are provided for performing fire, flood, seismic, and other hazard analyses.

To display end state cut sets, click the **End State** button.

◊ Highlight the end state(s), right click the mouse and select the **Display → Cut Sets**.



◊ The minimal cut set upper bound approximation frequency of the end state and the end state cut sets are now displayed.

◊ End state cut sets can be reported by clicking the **Reports** button.

◊ To view basic event information (descriptions and probabilities) in a cut set, highlight the cut set, click the view button or right click the mouse and select **View**.

◊ To view the path of where the cut set originated from, right click the mouse and select **Path**. The screen will expand out the information (logic required to obtain the cut set).

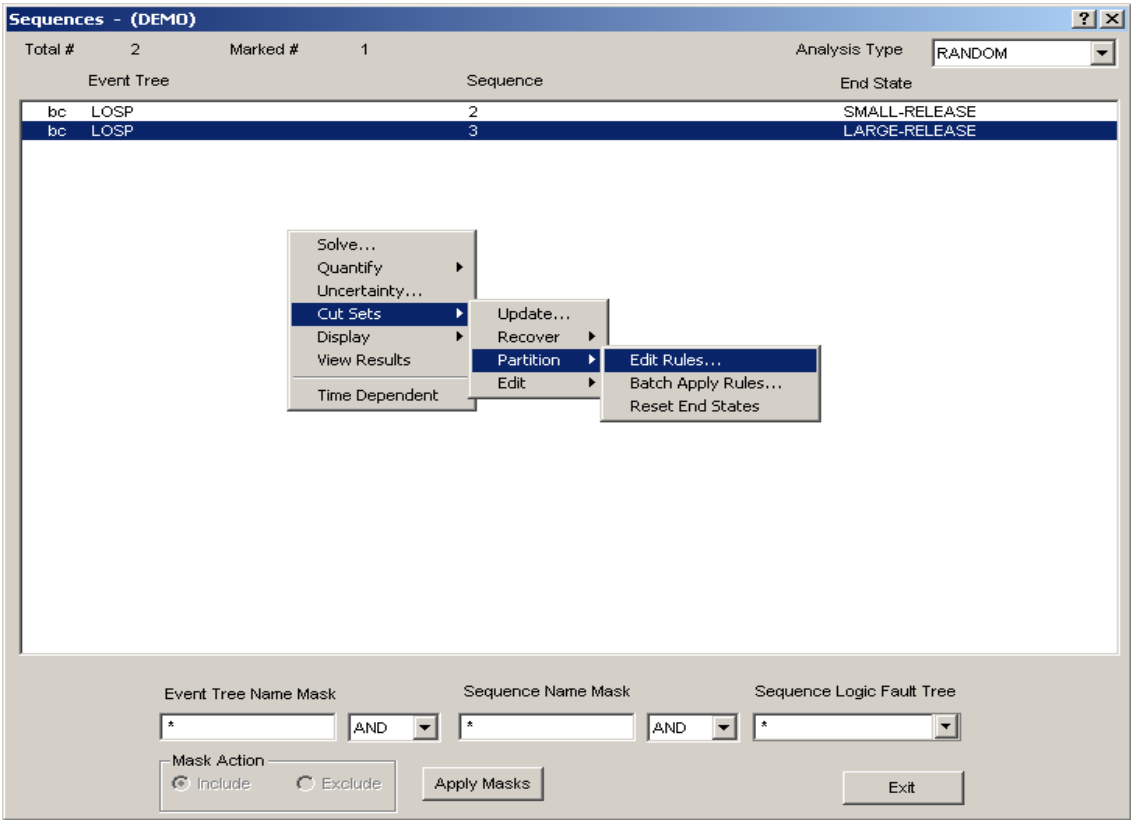
End State Functions in the Modify Menu

Several important functions are provided in the **Modify → End States** option.

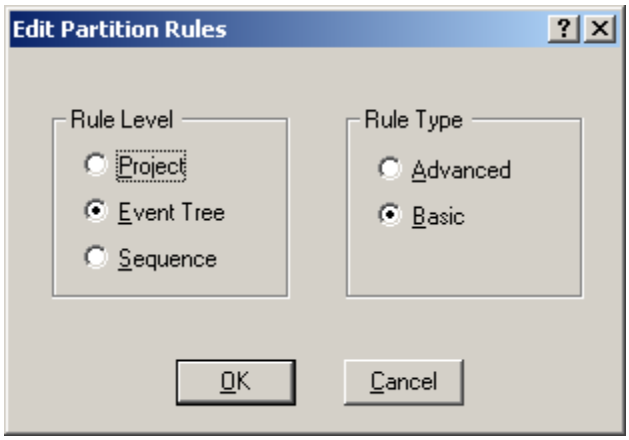
- ◊ The end state description editor (via right mouse click then **Modify**).
- ◊ The deletion of end states that are no longer used (via the "Remove Unused" option).
- ◊ The "Clear Current" option that empties the current case end states.
- ◊ The "Base Case" option that transfers the current case end state cut sets into the base case.
- ◊ The "X-Ref." Option lists the event tree sequences that are grouped into the highlighted end state.

6.3. End States via Partition Rules

The partition rule editor is located in the **Sequence → Cut Sets → Partition** option. Partitioning rules are entered and applied using the option shown below.



The “**Edit Rules**” option will execute a pop-up box shown below which will allow the creation of partition rules on a specific sequence (the highlighted sequence), specific event tree (event tree corresponding to the highlighted sequence) or all sequences.



Cut set partitioning rules can apply to the entire database project, an event tree, and/or a specific sequence.

Item	Menu Button	Name of rule(s)
Specific sequence	Sequence	Partition “Sequence” Rule Level
Single event tree	Sequence	Partition “Event Tree” Rule Level
All sequences	Sequence	Partition “Project” Rule Level

6.4. Partitioning Rule Nomenclature and Structure

The rule structure and nomenclature for the partitioning rules are similar to the "Link Event Tree" rules and “Recovery Rules” described in Sections 4 and 5.

The partitioning rule editor tests the existing sequence cut sets for the presence or absence of specific combinations of basic events or initiating events, and assigns characters in the end state name when the criteria are met. This allows end state names to be built as the rules are applied.

Symbols

|	Denotes a comment line	~	Operator for "not present"
*	Logical AND operator	+	Logical OR operator
/	Complement	( )	Grouping terms

Search Criteria Examples

Search Criteria	Meaning of the Search Criteria
DG-A	Basic event DG-A (failure)
~DG-A	Failure of DG-A is not present in the cut set
/DG-A	Complemented basic event DG-A (success)
init(LOSP)	Initiating event with the name LOSP
system(ECS)	Fault tree top event with name ECS

Partition Rule Structure (Example 1 – if-then)

```
| The "if-then" Rule Structure:
|       This rule adds -SBO as characters 4 through 7 of the end state name
|       when both DG-A and DG-B are present in the cut sets.
|       The ??? are placeholders in the end state name.  (The end state
|        name is initially blank.)

if DG-A * DG-B  then
  partition = "???-SBO";
endif

| Note that the partition statement must end with a semicolon.
| The end state name must be ≤ 24 characters.
| The end state characters are enclosed in quotation marks.
```

Partition Rule Structure (Example 2 – if-always)

```
| The "if-always" Rule Structure:
|       This rule adds END as the first 3 characters in every cut set.

if always then
  partition = "END";
endif
```

Partition Rule Structure (Example 3 – if-then-elsif)

```
| The "if-then-elsif" Structure:
|       This rule adds characters 4 through 7 to the end state name.
|       When both DG-A and DG-B are failed, -SBO is added.
|       When DG-A is failed (but not DG-B), characters -DGA are added.
|       When DG-B is failed (but not DG-A), characters -DGB are added.
|
if DG-A * DG-B then
  partition = "???-SBO";
elsif DG-A then
  partition = "???-DGA";
elsif DG-B then
  partition = "???-DGB";
endif
```

Partition Rule Structure (Example 4 – if-then-elsif-else)

```
| The "if-then-elsif-else" Structure:
if DG-A * DG-B then
   partition = "???-SBO";
elsif DG-A then
   partition = "???-DGA";
elsif DG-B then
   partition = "???-DGB";
else
   partition = "???-FLW";
endif

      | Note that the cut sets that do not contain DG-A or DG-B are assigned
      | to the ???-FLW endstate by the else statement, since they do not meet
      | the search criteria.
```

Partition Rule Structure (Example 5 – Macros)

Macros can be used to streamline complex rules. A macro is simply a user-defined keyword that specifies a search criterion that can be used in the rule instead of the individual events (i.e., search criterion). An example is provided below.

```
| Define a macro named ALL-DGS
ALL-DGS = DG-A * DG-B;

| Use the macro in a rule
if ALL-DGS then
  partition = "???-SBO";
endif
```

Partition Rule Structure (Example 6 – Macros and ~)

When creating a rule that indicates that events in the macro do not occur, use the ~ (not present) symbol. (Note, do not "complement" a macro.)

```
| Using ~macro as the search criteria:
|  The rule applies when failure of both DG-A and DG-B is not in the cut set.

if ~(ALL-DGS) then
  partition = "???-TRS;
endif
```

Partition Rule Structure (Example 7 – Current Partition)

```
|  The “Current Partition” Rule Structure:
|  The “current partition” rule structure uses the end state created by a partition
| rule to create a different end state using only those basic events currently found
| in the current end state.  (This rule makes two end states; one end state
| containing all of the basic events that meet the search criteria of the second rule
| and a second end state with those basic events that do not meet the search
| criteria of the second rule.  This rule can use wildcards as part of its search
| criteria.

| This rule creates an end state containing all cut sets with the basic event
| C-MOV-1. The rule then creates a new (second) end state using only the
| current end state cut sets, which contains only those cut sets that contains
| either E-MOV-A or E-MOV-B.
|
if C-MOV-1 then
  partition = “CMOV1”;
endif

if CurrentPart(C????) * (E-MOV-A + E-MOV-B) then
  partition = “C-E-MOVS”;
endif
```

Partition Rule Structure (Example 8 – Global Partition)

| The "GlobalPartition" Rule Structure:
| This rule globally partitions all cut sets in a sequence to an end state.
| This option is activated by using the keyword “GlobalPartition” instead
| of the normal “partition” keyword.
| This partition rule is much faster at gathering cut sets than using
| the normal “partition” rule. This rule is geared more for
| gathering cut sets based upon sequence logic than on
| individual basic events.

| The “GlobalPartition” rule structure is the same as for “partitioning” rules.
| This example “GlobalPartition” rule will gather all sequence cut sets that
| pertain to specified sequence logic.
| Cut sets will be put into an endstate called CD-SEQ2 if they are found in
| sequences that contain the following sequence logic
| LOSP * ECS * /CCS.
| Cut sets will be put into an endstate called CD-SEQ3 if they are found in
| sequences that contain the following sequence logic
| LOSP * ECS * CCS.
|

```
if  INIT(LOSP) * SYSTEM(ECS) * SYSTEM(/CCS)  then
  GlobalPartition = “CD-SEQ2";
elsif  INIT(LOSP) * SYSTEM(ECS) * SYSTEM(CCS)  then
  GlobalPartition = “CD-SEQ3";
endif
```

Note: Global Partitioning is designed for rapidly partitioning cut sets into end states based upon sequence logic. Since individual cut sets are not searched, the Global Partitioning rule is much faster at gathering cut sets than the other partitioning methods.

- The Global Partition rule loads all of the sequence cut sets into the end state in a single pass instead of evaluating each cut set. Consequently, it is recommended to “global partition” based upon initiators or system top events.

- If a “global partition” is performed on basic events, all cut sets listed after the basic event’s cut set will be partitioned into the end state.

Partition Rule Structure (Example 9 – Global Partition and Transfer)

Global Partition rules should not be mixed with normal “Partitioning” rules. Global Partition rules are geared more for Level 2 studies since the end state that is created is also an event tree with the same name. The event tree that is created uses the end state frequency as its initiating event frequency and then transfers to a Level 2 event tree. This “end state event tree” can be looked at as an event tree which transfers Level 1 information to Level 2 event trees.

```
| The “Global Partition” rule to transfer the end state frequency to be used
| by a Level 2 event tree.
|
| This rule creates an end state event tree to be used by a Level 2 event tree
| (which is already created).
|
| LEVEL2TREENAME can be viewed as a Level 2 event tree name. This tree
| will use the end state frequency gathered in the end state CD-SEQ3 as
| its initiating event frequency.

if  init(LOSP) * SYSTEM(ECS) * SYSTEM(CCS)  then
   GlobalPartition = “CD-SEQ3";
   transfer = LEVEL2TREENAME;
endif
```

6.5. Partition Rule Keywords and Nomenclature

Each of the “rules” in SAPHIRE (linking, recovery, and partition) has their own nomenclature. The table below lists the keywords available for partition rules.

Keyword or symbol	Definition	Usage

if then	Keyword that indicates search criteria is being specified.	if "search criteria" then
		perform some action on each cut set;
		endif

endif	Keyword that indicates the end of a particular rule.	if "search criteria" then
		perform some action on each cut set;
		endif

else	Keyword that specifies some action to be taken if all the search criteria are not met. The else should be the last condition in the recovery rule.	if "search criteria" then
		perform some action on each cut set;
		else
		perform action on each cut set if search criteria not met;
		endif

elsif	Keyword that specifies an alternative search criteria. Any number of elsifs can be used within a recovery rule.	if "search criteria" then
		perform some action on each cut set;
		elsif "2nd search criteria" then
		perform action on each cut set;
		elsif "3rd search criteria" then
		perform action on each cut set;
		endif

always	Keyword that indicates that every cut set that is being evaluated satisfies the search criteria.	if always then
		perform some action on each cut set;
		endif

init( )	Keyword used in the search criteria to indicate that a sequence cut set has a particular initiating event.	if init(INITIATOR-NAME) * "other search criteria if needed" then
		perform some action on each cut set;
		endif

system( )	Keyword used in the search criteria to indicate that the sequence logic contains the particular top event.	if system(TOP EVENT) * “other search criteria if needed” then
		perform action on each sequence;
		endif

Keyword or symbol	Definition	Usage
~	Symbol used in the search criteria to indicate that a particular event will not be in the cut set that is being evaluated.	if (~SEARCH-CRITERIA) * "other search criteria if needed" then ... The search criteria will be satisfied for all cut sets that do not contain SEARCH-CRITERIA (and also contains the optional "other search criteria").
/	Symbol used to represent a complemented event (i.e., the success of a system or basic event).	if (/BASIC-EVENT) * "other search criteria" then The search criteria will be satisfied for all cut sets that contain the complement of BASIC-EVENT (and also contains the optional "other search criteria").
|	Symbol used to represent a comment contained in the rules. Everything on a line to the right of this symbol will be ignored.	| Place your comments here! | Note that blank lines are also permissible!
;	Symbol to indicate the end of a macro line or a line that modifies the cut set being evaluated.	| usage for a macro command MACRO-NAME = "search criteria" ; | usage for a cut set modification line partition = ENDSTATE ;
*	Symbol to indicate the logical AND command.	if SEARCH-CRITERIA1 * SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match SEARCH-CRITERIA1 and SEARCH-CRITERIA2.
+	Symbol to indicate the logical OR command.	if SEARCH-CRITERIA1 + SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match either SEARCH-CRITERIA1 or SEARCH-CRITERIA2.
( )	Symbols to indicate a specific grouping of items.	if (A + B) * (C + D) then The search criteria above would return all cut sets that contain: [A * C], [A * D], [B * C], or [B * D].

Keyword or symbol	Definition	Usage

partition =	Keyword that indicates the end state characters for the cut sets meeting the search criteria will be modified according to the text after the equal sign.	if "search criteria" then
		  partition = “END_STATE_NAME”;
		endif

CurrentPart( )	Keyword that searches for cut sets that have already been assigned to the endstate indicated.	if CurrentPart(CORE-DAMAGE) then
		  partition = “NEW-CORE-DAMAGE”;
		endif

GlobalPartition=	Keyword to indicate that all cut sets in a particular sequence will be assigned to the end state identified after the equal sign.	if "search criteria" then
		  GlobalPartition = “MY-END-STATE”;
		endif

transfer =	Keyword to indicate the event tree to be created and transferred to for the sequence meeting the search criteria. The sequence end state frequency will be used as the initiating event frequency for the new event tree.	if "search criteria" then
		  GlobalPartition = “CORE-DAMAGE”;
		  transfer = LEVEL-2-TREE;
		endif

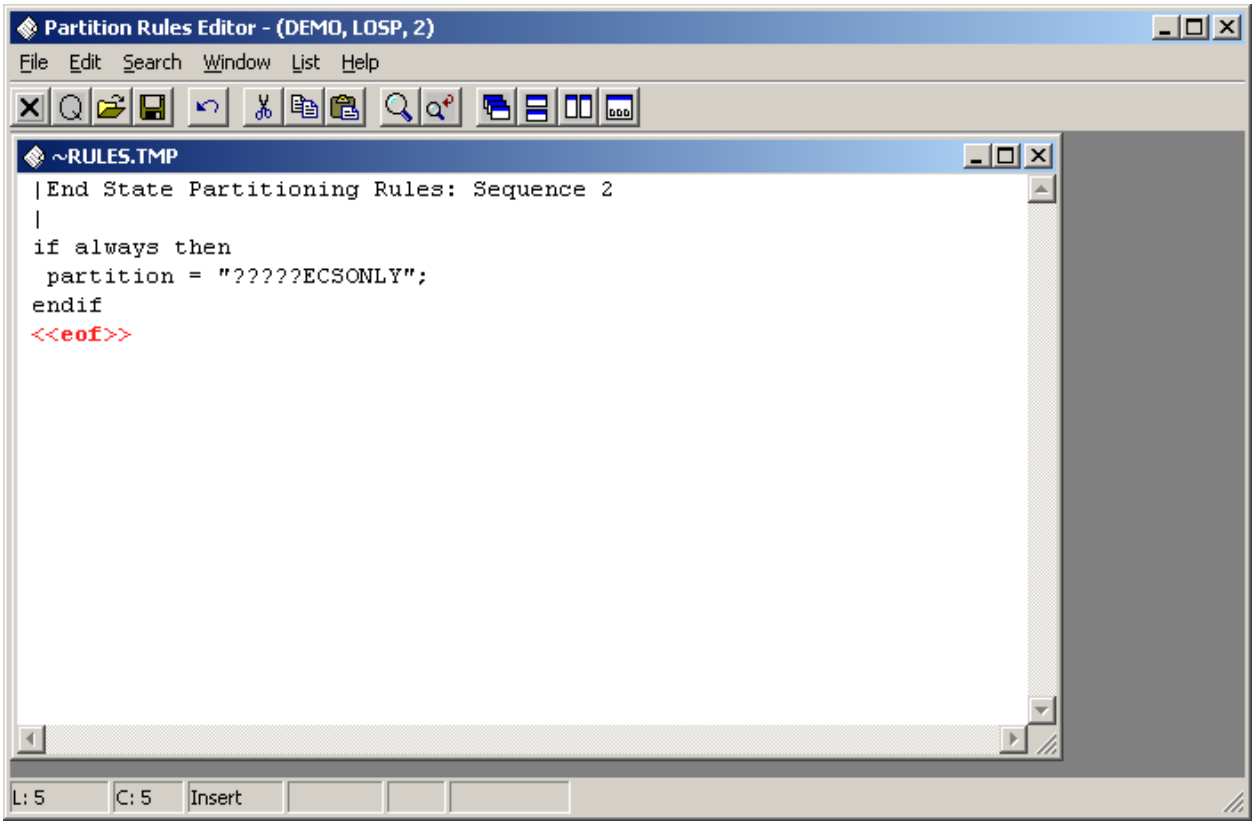
MACRO	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 16 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
		if MACRO-NAME then
		  perform some action on each cut set;
		endif
		
		|Macros are only applicable in the particular rule set where they appear

6.6. Partitioning Rule Editor and Operations

From the Sequence option, you can edit the partitioning rules. By right clicking the mouse and selecting **Cut Sets → Partition → Edit Rules**.

- ◊ The **Sequence** option lets you edit the rules for the highlighted sequence.
- ◊ The **Event Tree** option lets you edit event tree rules.
- ◊ The **Project** option lets you edit project rules.

Type the rules in the editor. The Partitioning Rule Editor window is shown below.



- ♦ **File:** The file drop down menu allows the user to save and exit the rule editor. The rules are automatically compiled when the rule editor is exited. If there is a problem in compiling the rule, SAPHIRE will not exit but go back to the rule editor and allow the user to fix the rule. The user can exit the rule editor without compiling if there are problems (e.g., the basic event is not in the database) by selecting Exit a second time.

- **Edit**: The edit drop down menu allows the user to cut, copy, and paste any portions of the existing rules.

- **Search**: The search drop down menu allows the user the ability to search the rule for a specific basic event and to either locate the basic event or replace the basic event with another basic event.

- **Window**: The window drop down menu is similar to normal window features but is not used when creating “partitioning” rules.

- **List**: The list drop down menu will provide a list of existing basic events, initiating events, systems, event trees, and macros that can be selected and automatically inserted into the rule wherever the cursor is located. SAPHIRE does have an “add” feature in the editor that is accessed by right-clicking on the list and selecting the **Add** option.

- **Help**: The help drop down menu will load up the help files associated with SAPHIRE.

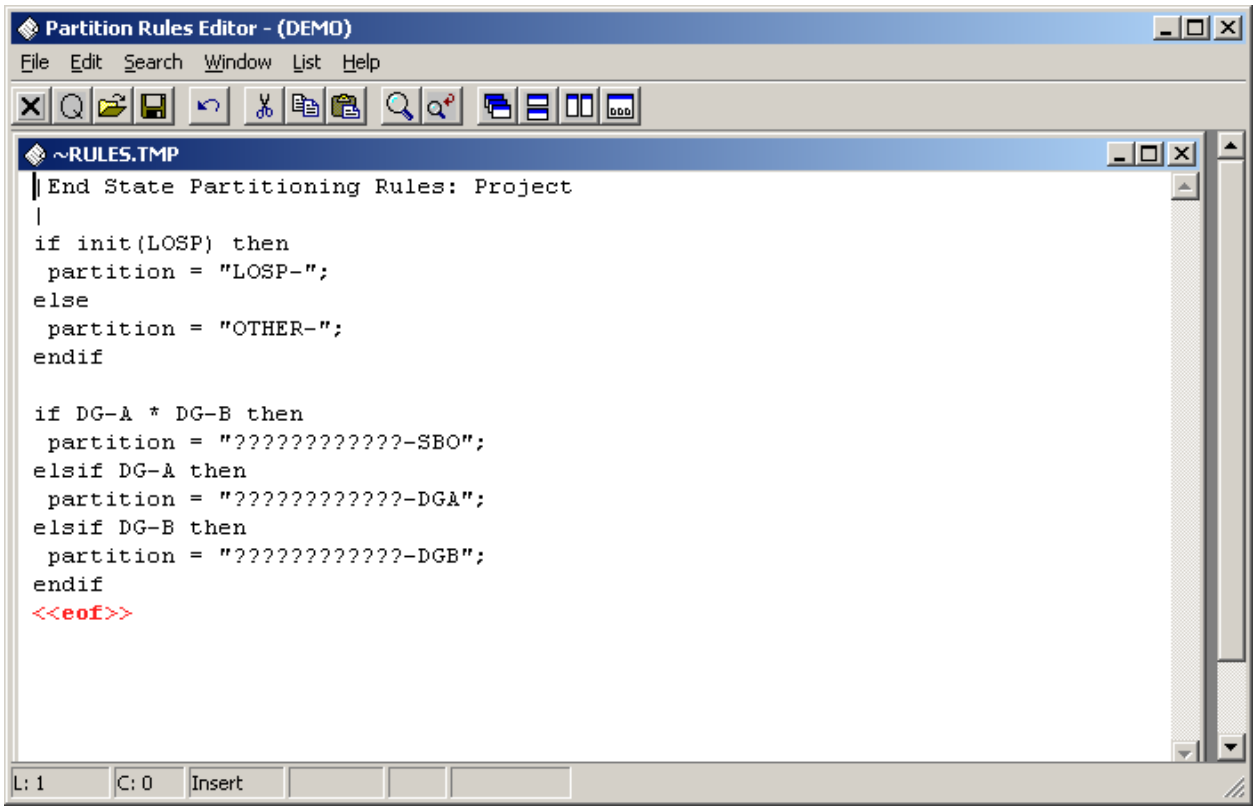
Note that if a user created a rule and exited the editor without compiling the rule, the user must return to the rule and make a change to the rule before SAPHIRE will compile the rule. This change could be as simple as adding and then removing a single space.

6.7. Partition Rule Example

A rule for the DEMO project is added by selecting the **Sequence → Cut Sets → Partition → Edit Rules → (select Project radio button)** partition rule option.

- The first project rule shown below add characters 1 through 5 as "LOSP-" when LOSP is the initiating event, and OTHER if LOSP is not the initiating event.

- The next rule adds characters 13 through 16 as -SBO when both DG-A and DG-B are failed. When DG-A is failed, DGA is added, and when DG-B is failed, DGB is added (as characters in the end state name).



Rules for LOSP sequences 2 and 3 are entered as shown below. To enter a rule that is applicable for only a specific sequence, use the **Cut Sets → Partition → Edit Rules → (select Sequence radio button)** option (highlight the specific sequence first).

- The rule in sequence 2 adds characters 6 through 12 as ECSONLY. Consequently, add the following rule to sequence 2.

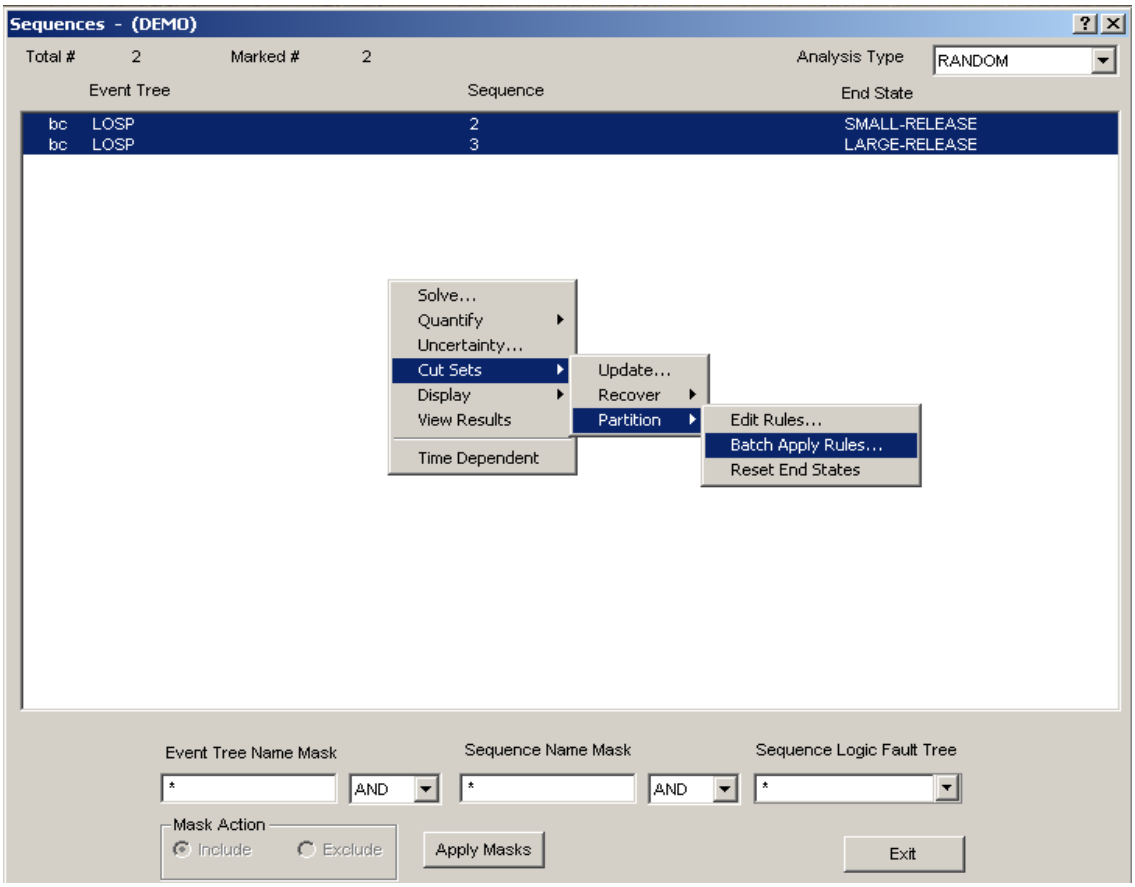
```
| End State Rule sequence 2
if  always  then
  partition = “?????ECSONLY”;
endif
```

- The rule in sequence 3 adds characters 6 through 12 as ECS&CCS. Consequently, add the following rule to sequence 3.

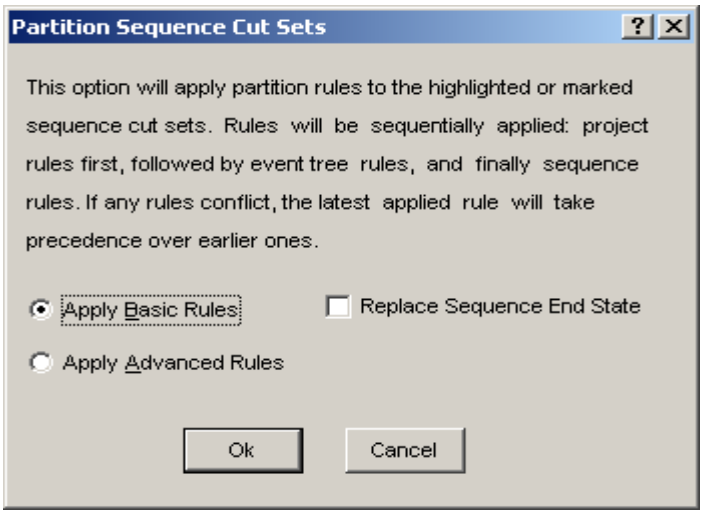
```
| End State Rule sequence 3
if  always  then
  partition = “?????ECS&CCS”;
endif
```

Applying the Partitioning Rules

To apply the partitioning rules, highlight the sequences of interest, right click the mouse, and select the **Cut Sets → Partition → Batch Apply Rules** option.



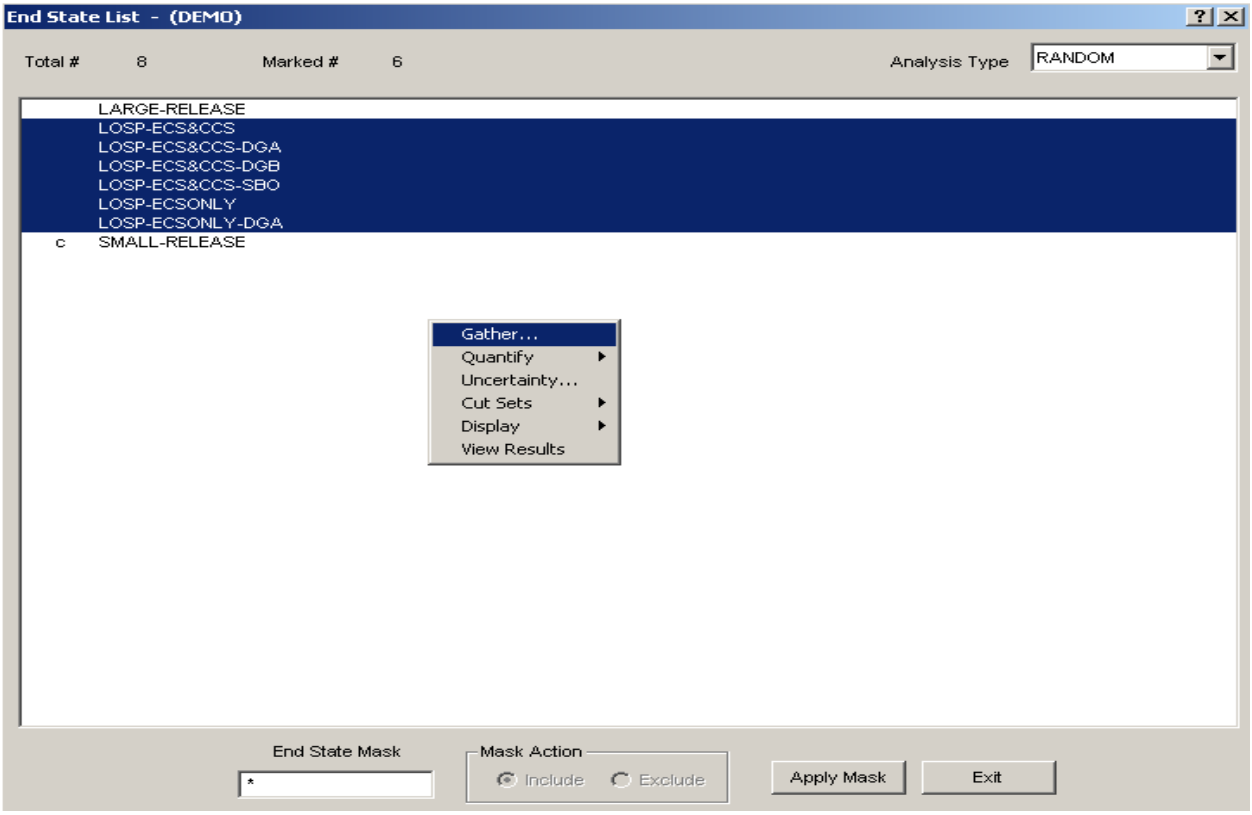
The screen shown below will appear. Click **OK** to proceed.



Using End State Analysis to Gather the Partitioned Cut Sets

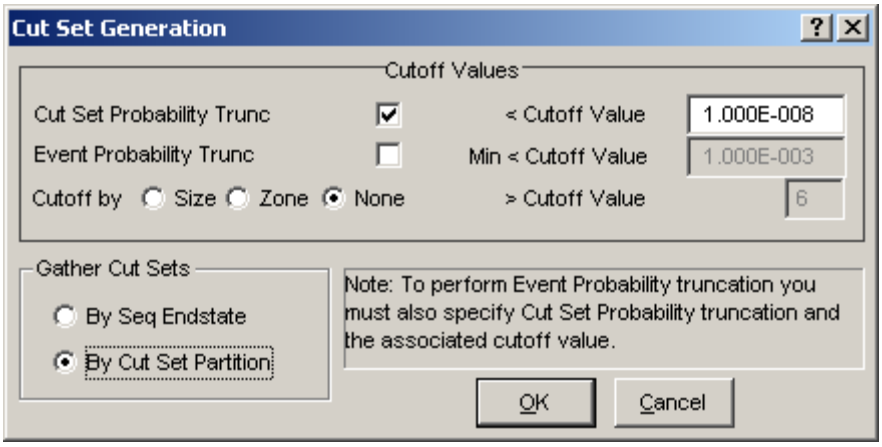
After applying the partition rules, the cut sets must be "gathered." To “gather” end states, the main **End State** option must be selected.

Highlight the end states created by partitioning rules, right click the mouse and select the Gather option.



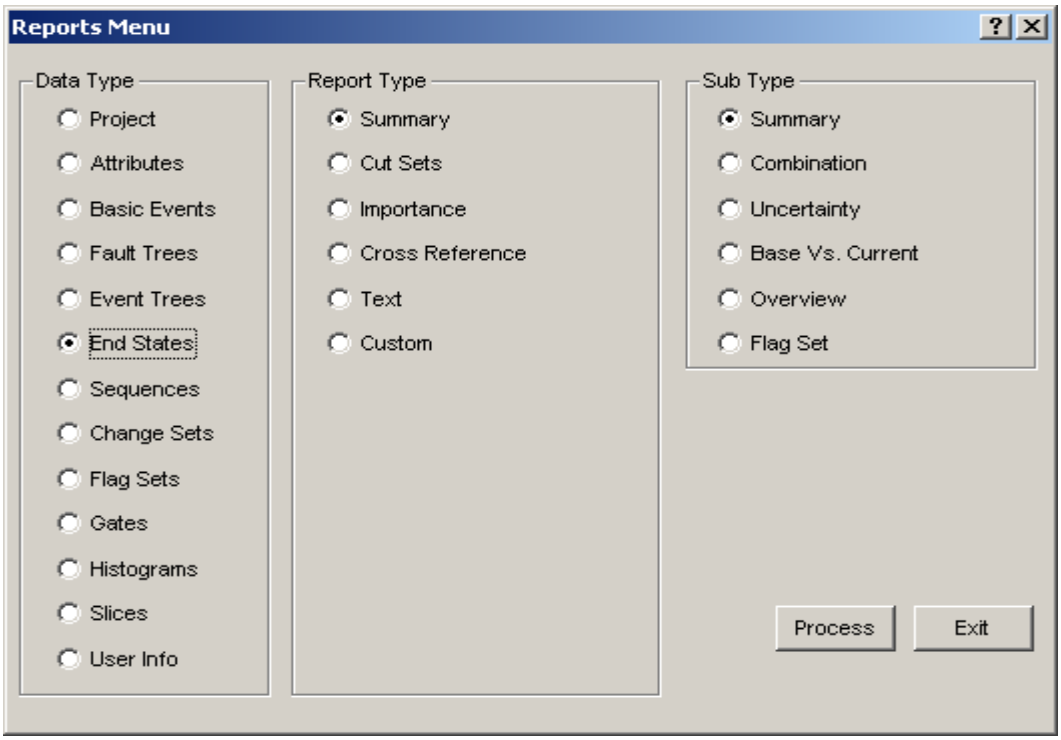
When you select the Gather option, you will be asked to specify the end state truncation options.

Make sure that the "By Cut Set Partition" option is selected.

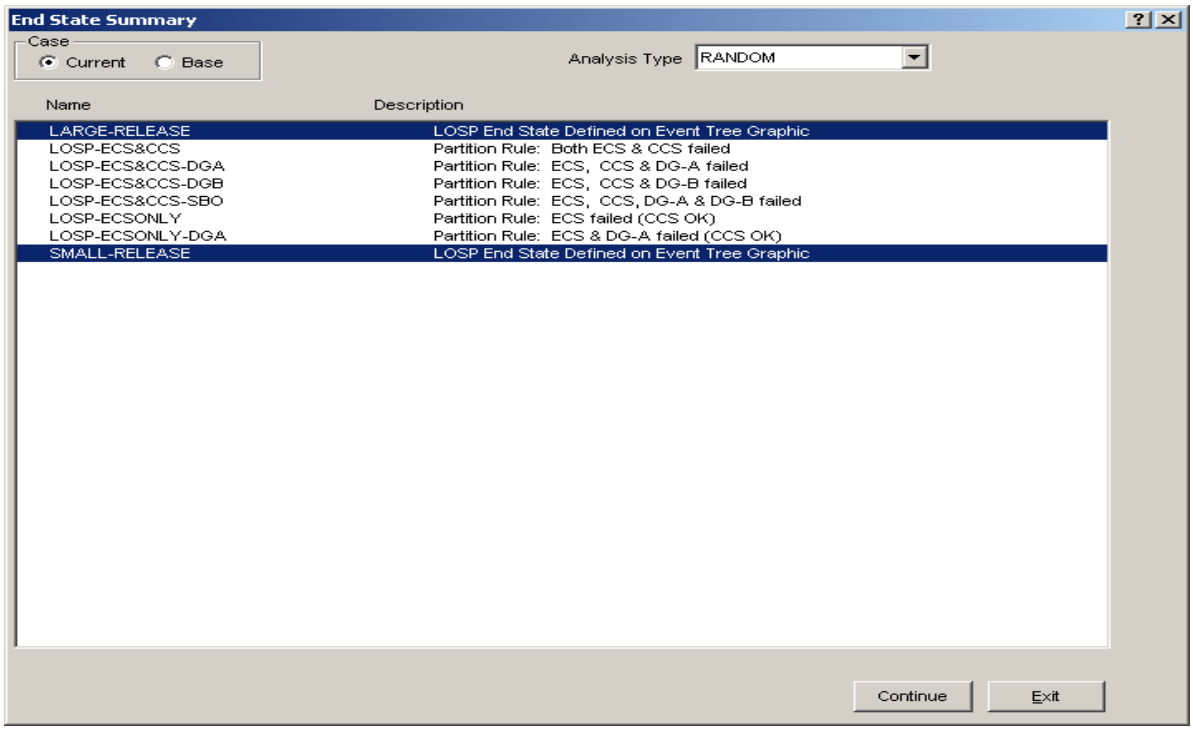


6.8. Reporting End State Results

The **Reports → End State** menu option provides various end state results including a summary of end state frequencies and uncertainty results.



Use the **Summary** "report type" option. Mark the combination of end states that are to be reported (note that both sequence and partition-based end states are listed).



With the LARGE-RELEASE and SMALL-RELEASE end states marked, the report shown below was generated.

End State	Description	Min. Cut Upper Bound
LARGE-RELEASE	LOSP End State Defined on Event Tree Graphic	1.760E-003
SMALL-RELEASE	LOSP End State Defined on Event Tree Graphic	4.840E-002
Total		5.016E-002

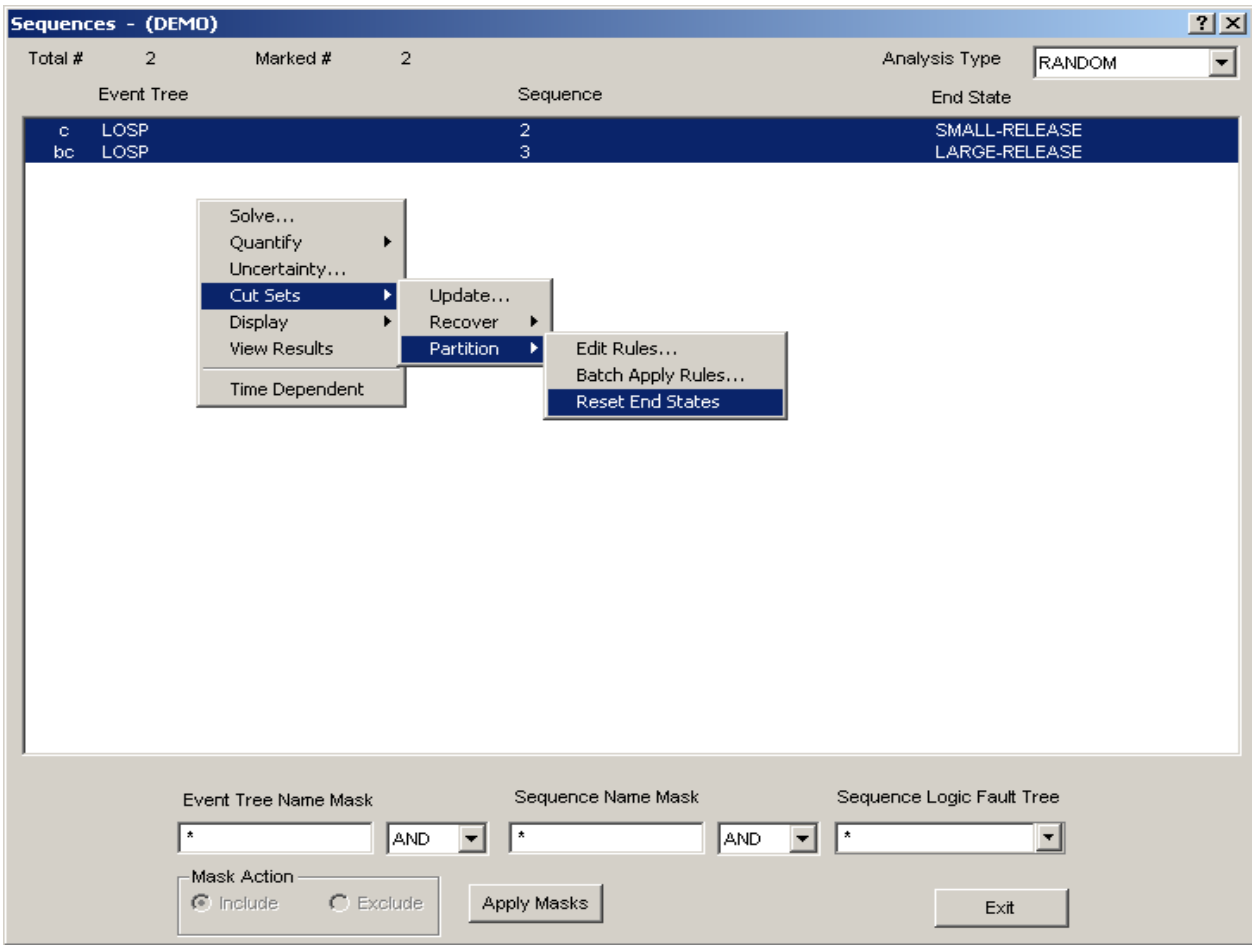
By marking the end states that were created by using the partition end state rules, the report shown below was generated.

End State	Description	Min. Cut Upper Bound
LOSP-ECS&CCS	Partition Rule: Both ECS & CCS failed	2.824E-006
LOSP-ECS&CCS-DGA	Partition Rule: ECS, CCS & DG-A failed	4.186E-004
LOSP-ECS&CCS-DGB	Partition Rule: ECS, CCS & DG-B failed	4.186E-004
LOSP-ECS&CCS-SBO	Partition Rule: ECS, CCS, DG-A & DG-B failed	9.200E-004
LOSP-ECSONLY	Partition Rule: ECS failed (CCS OK)	2.451E-003
LOSP-ECSONLY-DGA	Partition Rule: ECS & DG-A failed (CCS OK)	4.600E-002
Total		5.021E-002

Note: There is a minor difference between the overall total of the results shown above. This difference is due to round-off error. However, keep in mind that when comparing end state results to sequence results, differences can occur due to the removal of non-minimal cut sets when cut sets are gathered into the end state.

Reset Partition Rule End States

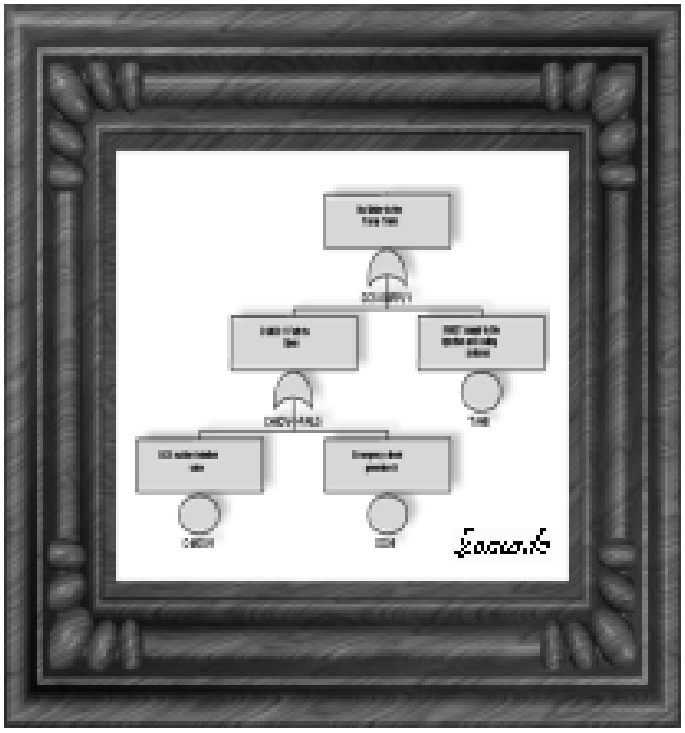
- To reset the end states created using the partitioning rules, highlight the sequence(s), right click, and select the **Cut Sets → Partition → Reset EndState** option.
  - The Reset End State option clears the sequence end state created by the partitioning rule.
  - When performing this option, the sequence end state in the sequence partition module used in SAPHIRE is reset to blank. Only blank end states can be removed (if desired) from the end states listed in the **Modify → End States** list. However, the cut sets will still appear in the end state until the end state is removed.



NOTES

7 SOLVING FAULT TREE CUT SETS

Section 7 describes the truncation options for analyzing fault tree cut sets. Model preparation prior to generating cut sets is discussed, and the various analysis and truncation options are described. Evaluating "subtrees," flag sets, and using process flags to prune fault tree logic is also described.



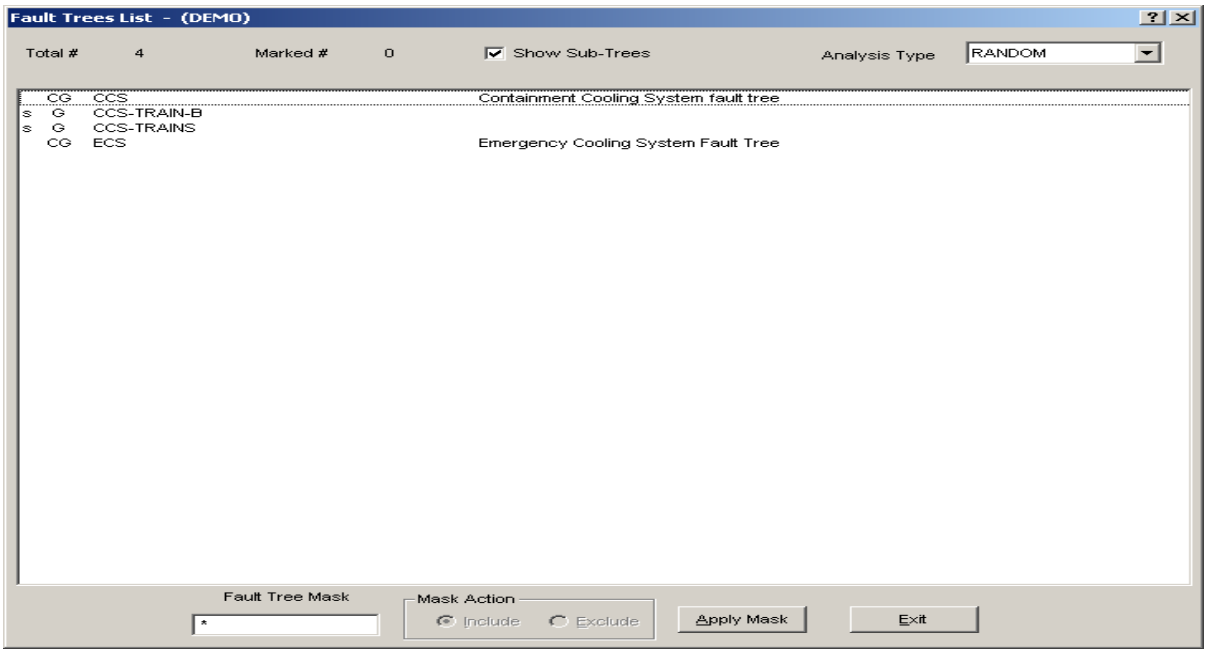
7.1. Solving Fault Tree Cut Sets

Fault tree cut sets are derived from the fault tree logic. Prerequisites that are required prior to solving for minimal cut sets are:

1. Fault tree logic was created by using the fault tree graphics editor, fault tree logic editor, or loaded into the database via the MAR-D interface.
2. Basic event data were added through the **Modify ➔ Basic Events** menu.
3. Basic event data was prepared by using the **Generate** option.
4. Fault tree transfers are properly modeled so that there are no logic loops in the fault trees, there is only one top gate in each fault tree, and the naming of transfer gates and fault tree filenames is consistent.

Menus and options for fault tree cut set solving.

- Select the **Fault Tree** option from the menu.



- Mark the Fault Trees using the mask feature, or using the mouse.
- Right-click to invoke the pop-up menu.
- Select the **Solve** option.

  C – flag indicates the fault tree has base case and current case cut sets

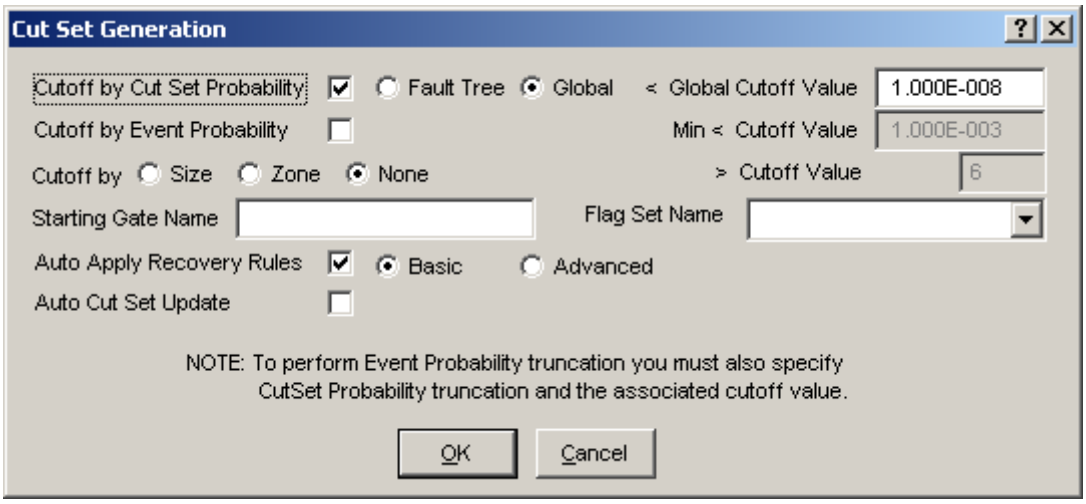
  c – flag indicates the fault tree has current case cut sets only

  b – flag indicates the fault tree has base case cut sets only

  G – flag indicates that the fault tree has graphics.

The **Solve** option uses the fault tree logic from all fault trees that link to the top gate in the fault tree. The fault tree probability is quantified using the user-specified quantification option (via the **Modify** option). The default quantification is the "minimal cut set upper bound."

Truncation Parameters



Select the desired truncation parameters on the dialog, and choose **OK** to begin generating cut sets.

Cutoff by Cut Set Probability - If you select this check box, then cut sets below the cutoff value will not be retained. Choose one of the radio buttons:

> ***Global*** – uses the cutoff value in the ***"< Global Cutoff Value"*** field.
>
> ***Fault Tree*** – uses the cutoff value stored in the fault tree record (via the **Modify ➔ Fault Tree** option).

Cutoff by Event Probability - If you select this check box, then you must also select the ***Cutoff by Cut Set Probability*** check box. This option will retain cut sets comprised of basic events that are above the ***Min < Cutoff Value*** even if the cut set is below the ***Global Cutoff Value***. This option is generally not used.

Cut Set Size - If you select this check box, then cut sets having more events than specified in the ***> Cutoff Value*** field will not be retained. If you select the ***Zone*** check box, then cut sets having more Zone Flagged Events than specified in the ***> Cutoff Value*** field will not be retained. If neither check box is selected, then the number of events in a cut set will not affect whether the cut set is retained or discarded. This option is generally not used.

Starting Gate Name - If you leave the field blank, the top gate in the system will be used. If you specify a gate, that gate will be used as though it were the top gate. This option is generally not used.

Flag Set Name - If you leave the field blank, the system-specific flag set, if any, will be used. If you specify a flag set, that flag set will be used during processing. This option is generally not used.

Auto Apply Recovery Rules – If you check this box, any recovery rules associated with this fault tree will automatically be applied after the fault tree cut sets have been generated. (The default is the **Basic Rules** which are the rules discussed in this workbook and the rules predominantly used.)

Auto Cut Set Update – If this box is checked, then the existing current case cut sets are reevaluated to remove all non-minimal cut sets and the fault tree probability is requantified.

Selecting the Appropriate Fault tree Analysis Option

Solve — This option uses the fault tree logic from all fault trees that link to the top gate in the fault tree. The fault tree probability is quantified using the minimal cut set upper bound approximation (or the default method defined in the "Define Constants"). This option is appropriate for all sensitivity studies where fault tree logic is available; however, it will take longer than the Cut Set Update or Quantification options.

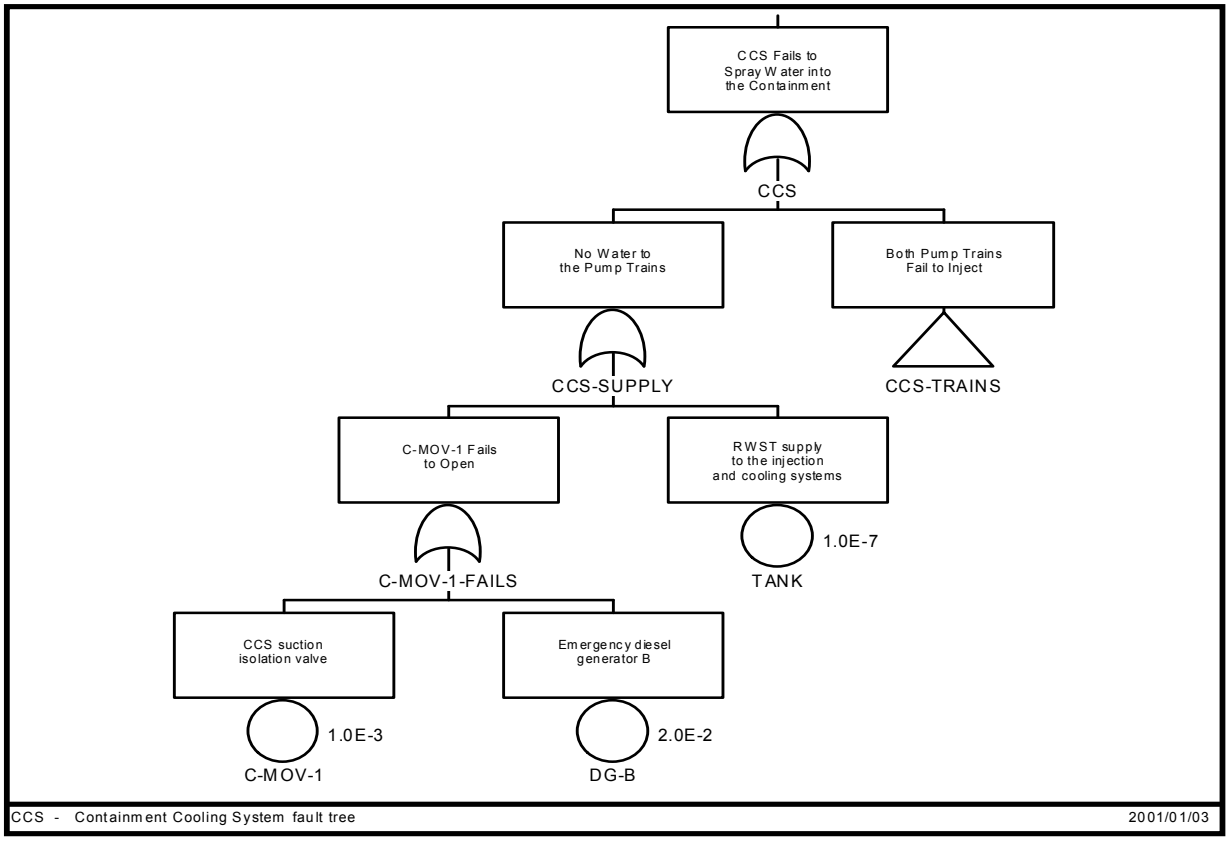
Cut Set Update (under **Cut Sets → Update**) — This option uses the existing current case cut sets (unless the user specifies that base case cut sets are to be used instead). Non-minimal cut sets are eliminated and the fault tree probability is quantified using the minimal cut set upper bound approximation (or the default method specified for the tree). Do not use this option if the fault tree logic is modified, event probabilities are increased, or the truncation is lowered from when the existing cut sets were solved.

Quantification (under **Quantify → *method***) — This option uses the existing current case cut sets and requantifies the fault tree probabilities using the minimal cut set upper bound approximation, rare event approximation, or min-max equation. This option is designed to quickly requantify the cut sets when data changes have been

made. (Note: if data changes increase the failure probability of an event, the Solve option should be used instead.) This option must be used if the fault tree has cut sets, but does not have fault tree logic.

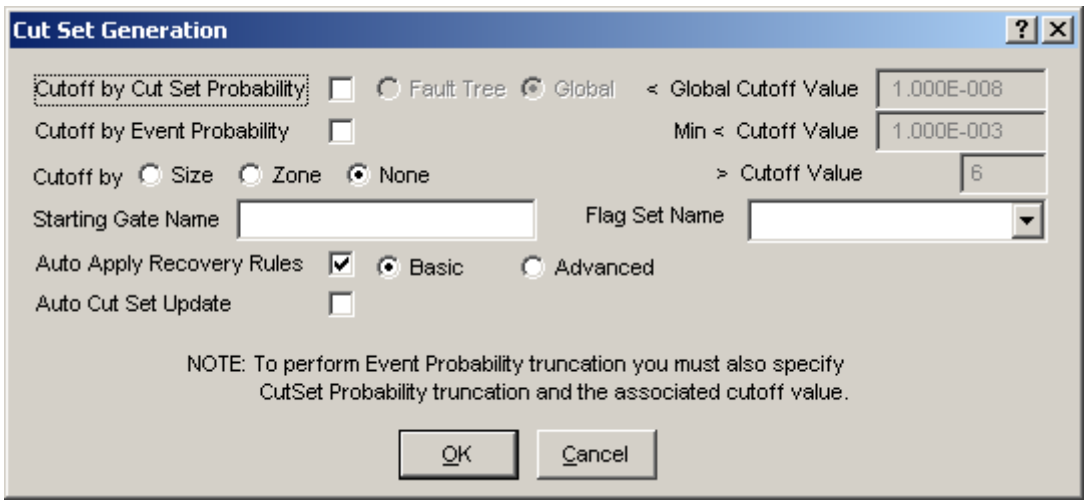
7.2. Examples of Fault Tree Solve Options

The CCS Fault Tree shown below will be used to demonstrate the various solve options. Notice the gate CCS-TRAINS is a transfer gate. The fault tree was paged (via the **Fault Tree → Page Tree** option) at the CCS-TRAINS gate to create the two fault trees (the main CCS tree and a CCS-TRAINS subtree).



Fault Tree Cut Sets With No Truncation

To generate cut sets without truncation, uncheck the Cutoff by Cut Set Probability box.



Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

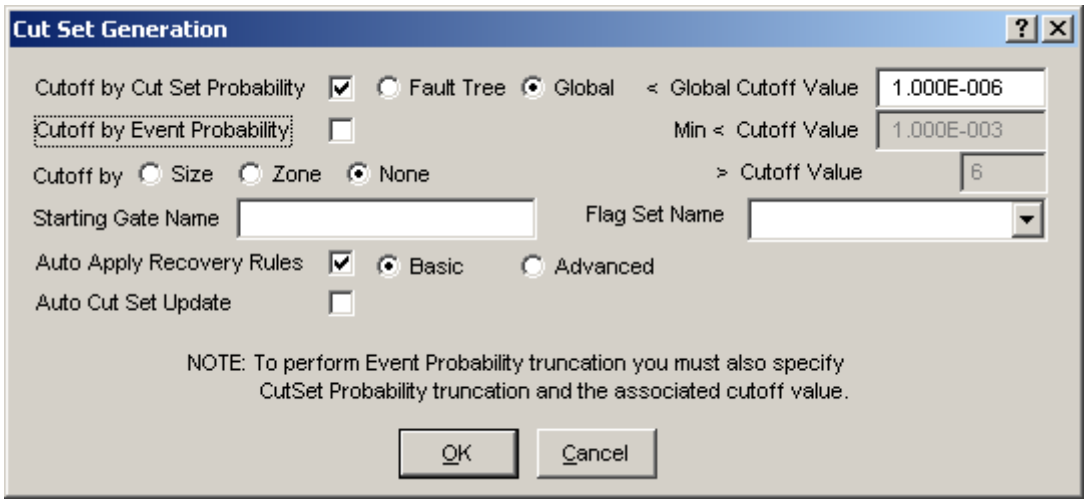
Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 2.120E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	94.33	94.33	2.000E-002	DG-B
2	99.05	4.72	1.000E-003	C-MOV-1
3	99.53	0.48	1.000E-004	C-MOV-B, DG-A
4	99.82	0.29	6.000E-005	C-PUMP-B, DG-A
5	99.94	0.12	2.500E-005	C-MOV-A, C-MOV-B
6	100.00	0.08	1.500E-005	C-MOV-A, C-PUMP-B
7	100.00	0.08	1.500E-005	C-MOV-B, C-PUMP-A
8	100.00	0.05	9.000E-006	C-PUMP-A, C-PUMP-B
9	100.00	0.01	2.000E-006	C-CV-B, DG-A
10	100.00	0.01	5.000E-007	C-CV-A, C-MOV-B
11	100.00	0.01	5.000E-007	C-CV-B, C-MOV-A
12	100.00	0.01	3.000E-007	C-CV-A, C-PUMP-B
13	100.00	0.01	3.000E-007	C-CV-B, C-PUMP-A
14	100.00	0.01	1.000E-007	TANK
15	100.00	0.01	1.000E-008	C-CV-A, C-CV-B

Fault Tree Cut Sets With Probability Truncation

To generate cut sets with truncation, check the Cutoff by Cut Set Probability box and specify a truncation value of 1.0E-6.



Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

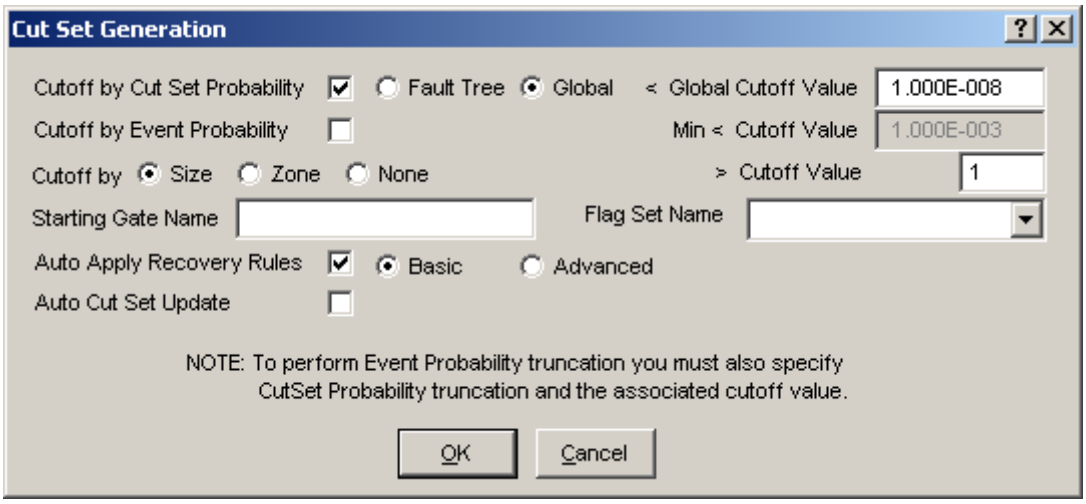
Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 2.120E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	94.34	94.34	2.000E-002	DG-B
2	99.06	4.72	1.000E-003	C-MOV-1
3	99.54	0.48	1.000E-004	C-MOV-B, DG-A
4	99.83	0.29	6.000E-005	C-PUMP-B, DG-A
5	99.95	0.12	2.500E-005	C-MOV-A, C-MOV-B
6	100.00	0.08	1.500E-005	C-MOV-A, C-PUMP-B
7	100.00	0.08	1.500E-005	C-MOV-B, C-PUMP-A
8	100.00	0.05	9.000E-006	C-PUMP-A, C-PUMP-B
9	100.00	0.01	2.000E-006	C-CV-B, DG-A

Fault Tree Cut Sets with Size Truncation

To generate cut sets with truncation on the number events in a cut set, click the Cutoff by Size and enter the size cutoff value (a 1 in our example). Also, specify a Cutoff by Cut Set Probability value of 1.0E-8.



Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

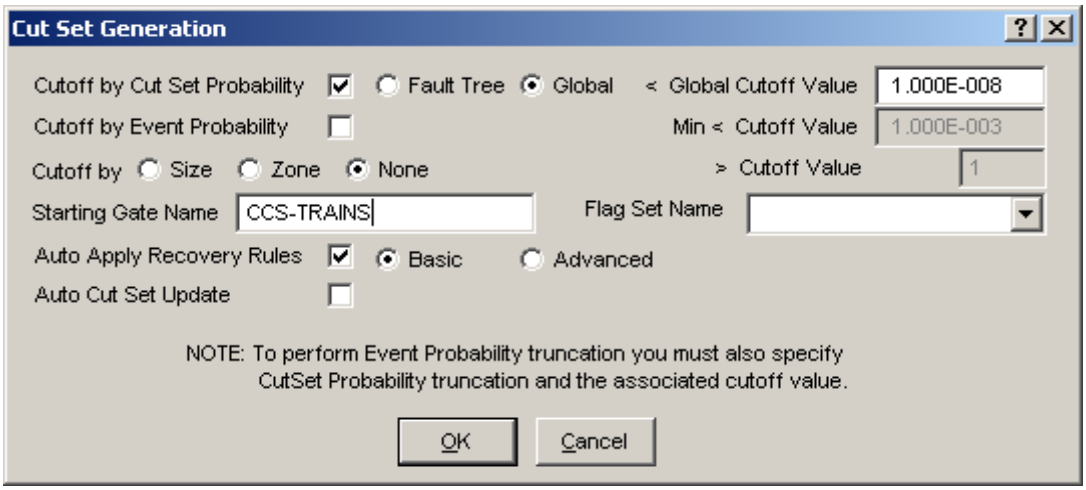
Min Cut Upper Bound: 2.098E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	95.33	95.33	2.000E-002	DG-B
2	100.00	4.77	1.000E-003	C-MOV-1
3	100.00	0.01	1.000E-007	TANK

Note that the “Cutoff by Event Probability” option is rarely used and will not be discussed.

Analyzing Fault Tree "Sub-trees"

To generate cut sets beginning with a gate below the top gate, enter the gate name in the "Starting Gate Name" field. In this example, cutoff by cut set probability was checked and the starting gate specified was CCS-TRAINS.



When you specify a starting gate name, you must remember that only the cut sets for the subtree are stored for the CCS fault tree.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 7.894E-004

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	50.68	50.68	4.000E-004	DG-A, DG-B
2	63.35	12.67	1.000E-004	C-MOV-A, DG-B
3	76.02	12.67	1.000E-004	C-MOV-B, DG-A
4	83.63	7.61	6.000E-005	C-PUMP-A, DG-B
5	91.23	7.61	6.000E-005	C-PUMP-B, DG-A
6	94.41	3.17	2.500E-005	C-MOV-A, C-MOV-B
7	96.31	1.91	1.500E-005	C-MOV-A, C-PUMP-B
8	98.22	1.91	1.500E-005	C-MOV-B, C-PUMP-A
9	99.36	1.15	9.000E-006	C-PUMP-A, C-PUMP-B
10	99.62	0.26	2.000E-006	C-CV-A, DG-B
11	99.88	0.26	2.000E-006	C-CV-B, DG-A
12	99.95	0.07	5.000E-007	C-CV-A, C-MOV-B
13	100.00	0.07	5.000E-007	C-CV-B, C-MOV-A
14	100.00	0.04	3.000E-007	C-CV-A, C-PUMP-B
15	100.00	0.04	3.000E-007	C-CV-B, C-PUMP-A
16	100.00	0.01	1.000E-008	C-CV-A, C-CV-B

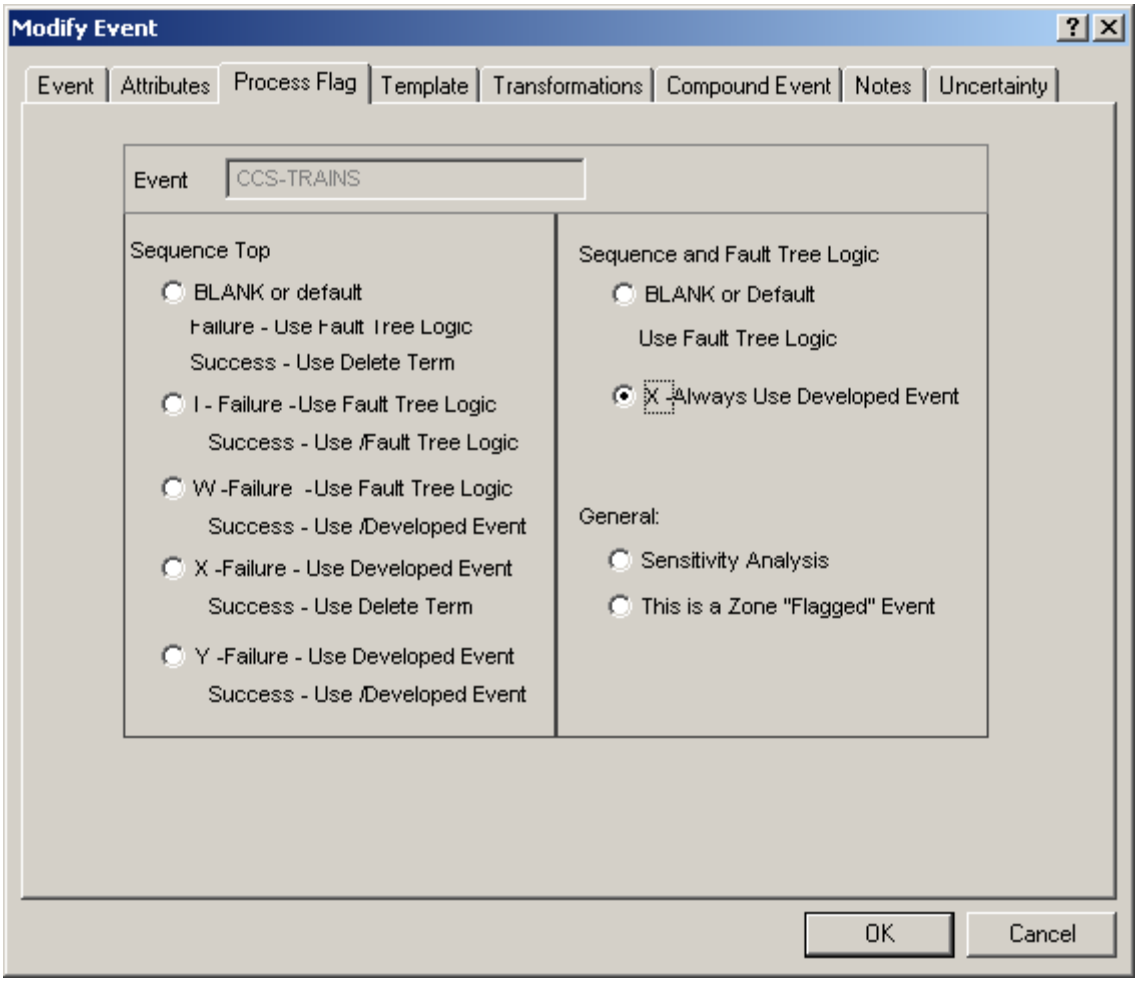
Note that DG-B appears in several cut sets when only the CCS-TRAINS subtree is analyzed. When the CCS gate is the starting gate, DG-B appears as a "single" in the cut sets; therefore, all combinations of DG-B with other basic events are non-minimal cut sets and they are eliminated.

Treating a Fault Tree Gate as a Basic Event

Now, we will solve the CCS tree while treating the CCS-TRAINS gate as a basic event (rather than its subtree logic)

To treat the CCS-TRAINS gate as though it were a basic event, either:

1. Set its *Process Flag* to the "X" type in **Modify → Basic Events**
2. Make a Change Set in the Generate option, where the CCS-TRAINS event's process flag is set to "X."



- Any probability can be specified for CCS-TRAINS; however, it was left as 1.0 in this example.

- Remember: All fault tree top gates and event tree top events are automatically defined as "basic events" in SAPHIRE. As such, they can be edited with Change Sets or via the **Modify → Basic Events** menu.

Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 1.000E+000

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	100.00	100.00	1.000E+000	CCS-TRAINS
2	100.00	2.00	2.000E-002	DG-B
3	100.00	0.10	1.000E-003	C-MOV-1
4	100.00	0.00	1.000E-007	TANK

Treating a Fault Tree Gate as a Basic Event with an Appropriate Probability

In this example, we will treat the CCS-TRAINS gate as though it were a basic event with a probability equal to its calculated minimal cut set upper bound value (the sub tree value).

To have SAPHIRE automatically use the sub-tree cut sets, we must:

1. Generate cut sets for CCS-TRAINS (**Fault Tree → Solve**).
2. Modify the CCS-TRAINS Calculation Type to “S” and the Process Flag to “X” (**Modify → Basic Events or use a Change Set, Generate → Add**)
3. Generate event data (under **Generate → Generate**).

Now, solve the CCS fault tree without truncation. Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 2.175E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	91.95	91.95	2.000E-002	DG-B
2	96.55	4.60	1.000E-003	C-MOV-1
3	100.00	3.63	7.894E-004	CCS-TRAINS
4	100.00	0.01	1.000E-007	TANK

Treating a Fault Tree Gate as Failed

To model failure of the entire CCS-TRAINS sub-tree (for example, if the subsystem is not functional), we need to specify that CCS-TRAINS was failed (a House Event TRUE).

> Set the CCS-TRAINS Process Flag equal to "X" (as discussed earlier) and set the Calculation Type equal to "T" (**Modify → Basic Events or Change Set**).

Now, solve the CCS fault tree without truncation. Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 1.000E+000

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	100.00	100.00	1.000E+000	<TRUE>

Treating a Fault Tree Gate as Working

To model success of the entire CCS-TRAINS sub-tree (for example, if the subsystem is working), we need to specify that CCS-TRAINS was functional (a House Event FALSE).

> Set the CCS-TRAINS Process Flag equal to "X" (as discussed earlier) and set the Calculation Type equal to "F" (**Modify → Basic Events or Change Set**).

Now, solve the CCS fault tree without truncation. Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

Min Cut Upper Bound: 2.098E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	95.33	95.33	2.000E-002	DG-B
2	100.00	4.77	1.000E-003	C-MOV-1
3	100.00	0.00	1.000E-007	TANK

Bonus Question: *What would the result be if the CCS top gate was an AND gate (instead of an OR gate)?*

Ignoring a Fault Tree Gate

Occasionally, one would like to see the output of fault tree logic with a portion of the fault tree removed. Rather than having to physically delete portions of the tree, SAPHIRE will allow a gate (or an event) to be ignored.

To remove an event or gate from a fault tree, set its Calculation Type equal to "I" (for Ignore).

> Set the CCS-TRAINS Process Flag equal to "X" (as discussed earlier) and set the Calculation Type equal to "I" (**Modify → Basic Events or Change Set**).

Now, solve the CCS fault tree without truncation. Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows.

Sort/Slice Cut Set Report

Project : DEMO Fault Tree: CCS

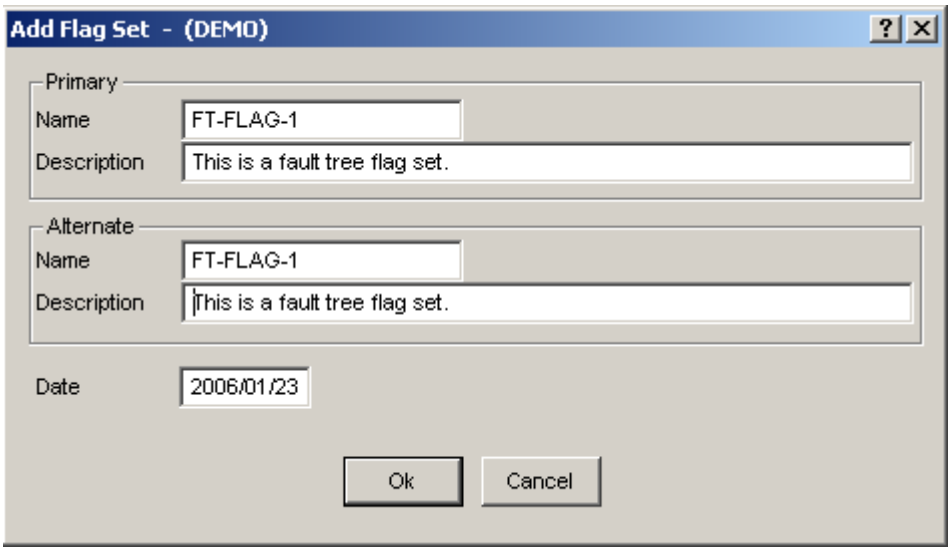
Min Cut Upper Bound: 2.098E-002

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	95.33	95.33	2.000E-002	DG-B
2	100.00	4.77	1.000E-003	C-MOV-1
3	100.00	0.01	1.000E-007	TANK

7.3. Using Flag Sets During Fault Tree Cut Set Solving

First, a brief review of flag sets.

Flag Sets are a special type of change set. SAPHIRE will keep flag sets separate from change sets by specifying it as a flag set. Fault tree flag sets are created using the **Modify → Flags** menu.



- Flag Sets can only contain individually selected changes. No "Class Changes" are allowed.
- Flag Sets are used to indicate modifications to particular events or gates on individual fault trees.
- A basic event probability of failure may *not* be changed in Flag Sets.

When generating fault tree cut sets, Flag Sets are used for setting house events or basic events to either TRUE, FALSE, or IGNORE.

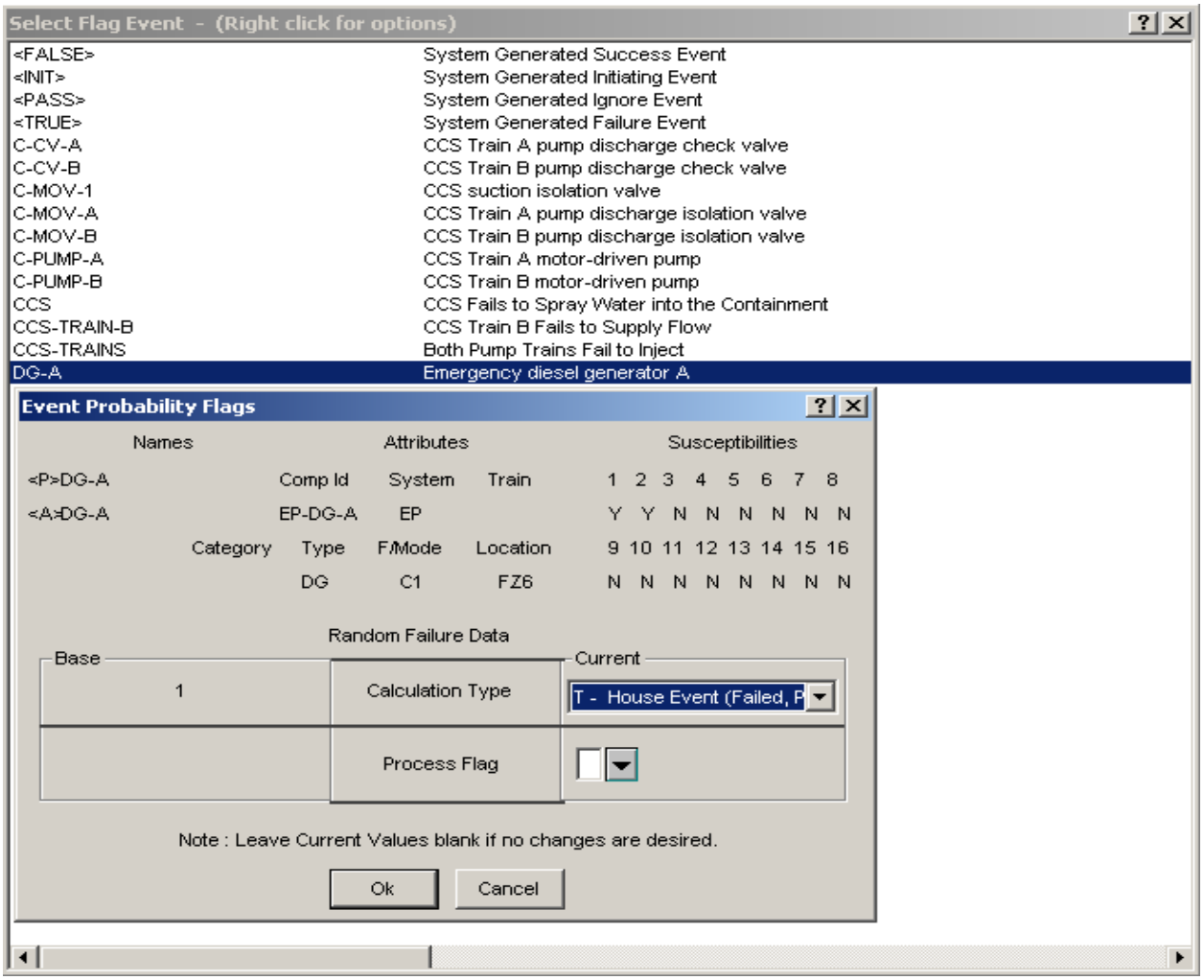
To make the Flag Set

1. Enter the **Modify → Flags** menu
2. Right click the mouse and select Add, and enter the Flag Set name (maximum of 24 characters) and description. Click OK.

3. Highlight the Flag Set then click the **Flags** button on the bottom of page. Right click the mouse and select **Add** to add events to the Flag Set.

4. Highlight the event to be modified, right click the mouse and select **Add**.

5. Modify the calculation type to either TRUE, FALSE or IGNORE.



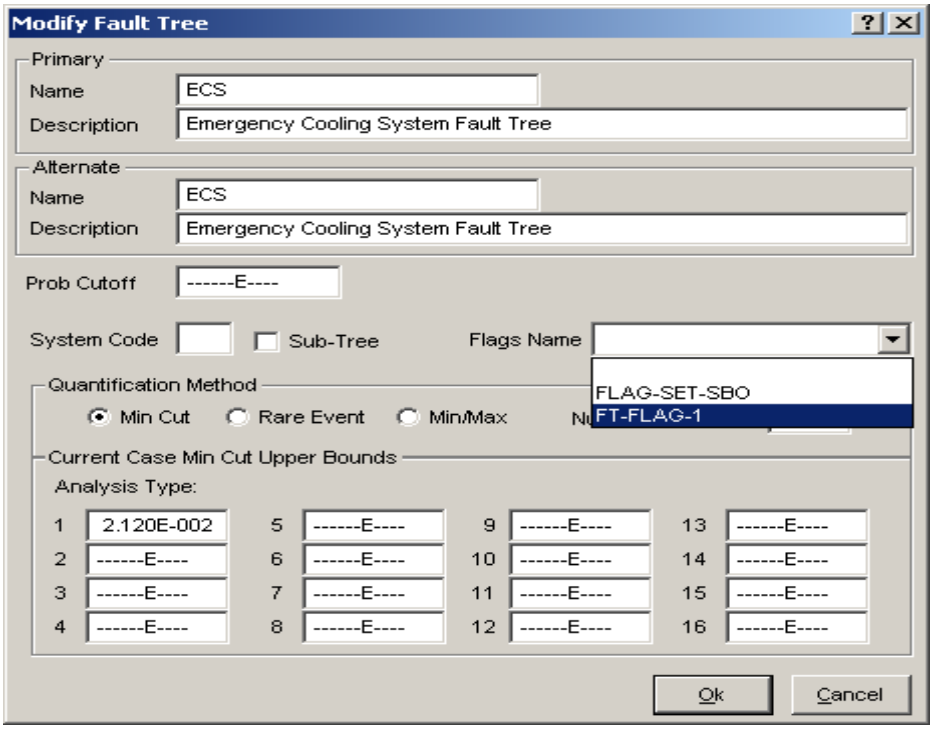
6. Continue adding as many events as necessary to the Flag Set.

To use a Flag Set

♦ After the Flag Set has been created, the Flag Set name needs to be assigned to one or more fault trees.

◊ Select the **Modify** → **Fault Tree** menu.

◊ Highlight the fault tree that will use the Flag Set, right click the mouse and select **Modify**. The Flag Set name is entered in the field labeled "Flags Name."



To illustrate the use of Fault Tree Flag Sets, the Demo project is used.

♦ The Flag Set named "FT-FLAG-1" was created

◊ In this Flag Set, DG-A basic event was set to FALSE.

♦ The Flag Set "FT-FLAG-1" was assigned to fault tree ECS.

♦ Now, solve the ECS fault tree without truncation. Reporting the solve results (**Fault Tree → Display → Cut sets → Report**) shows (note that the cut sets do not include basic event DG-A).

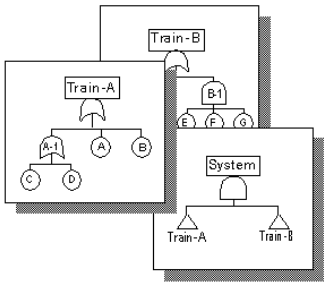
Sort/Slice Cut Set Report

Project : DEMO Fault Tree: ECS

Min Cut Upper Bound: 1.227E-003

Cut No.	% Total	% Cut Set	Prob./Frequency	Cut Sets
1	81.47	81.47	1.000E-003	E-MOV-1
2	89.63	8.15	1.000E-004	DG-B, E-MOV-A
3	94.52	4.89	6.000E-005	DG-B, E-PUMP-A
4	96.56	2.04	2.500E-005	E-MOV-A, E-MOV-B
5	97.79	1.23	1.500E-005	E-MOV-B, E-PUMP-A
6	99.02	1.23	1.500E-005	E-MOV-A, E-PUMP-B
7	99.75	0.74	9.000E-006	E-PUMP-A, E-PUMP-B
8	99.92	0.17	2.000E-006	DG-B, E-CV-A
9	99.97	0.05	5.000E-007	E-CV-B, E-MOV-A
10	100.00	0.05	5.000E-007	E-CV-A, E-MOV-B
11	100.00	0.03	3.000E-007	E-CV-B, E-PUMP-A
12	100.00	0.03	3.000E-007	E-CV-A, E-PUMP-B
13	100.00	0.01	1.000E-007	TANK
14	100.00	0.01	1.000E-008	E-CV-A, E-CV-B

7.4. Steps Performed During Fault Tree Solving



RESTRUCTURE

- Change gate to a basic event?
  - "X" process flag.
- Consistency check.
- Unresolved transfers changed to basic events.

System	AND	Train-A	Train-B
Train-A	OR	A-1	A B
A-1	OR	C	D
Train-B	OR	H	B-1
B-1	AND	E	F G

EXPAND

- Expand any N/M gates into AND and OR gates.
- Determine the logic top gate.

top gate is bolded in example below...

System	AND	Train-A	Train-B
Train-A	OR	A-1	A B
A-1	OR	C	D
Train-B	OR	H	B-1
B-1	AND	E	F G

CHECK/CONVERT

- Check for logic loop errors.
- Convert complemented gates. (see page 5-9, SAPHIRE Basics)

Logic OK...no logic loops.

System	AND	Train-A	Train-B
Train-A	OR	A-1	A B
A-1	OR	C	D
Train-B	OR	H	B-1
B-1	AND	E	F G

PRUNE/CONDENSE

- Perform House event pruning (see page 6-9, SAPHIRE Basics).
- Coalesce like gates...
  - OR gates input to OR gates
  - AND gates into AND gates

If event "F" set to a FALSE house event...

System	AND	Tain-A	Train-B
Train-A	OR	A-1	A B
A-1	OR	C	D
Train-B	OR	H	

OPTIMIZE

- Determine if any modules exist in logic.
- Determine if any independent subtrees exist in logic.

Train-A, Train-B, and A-1 are independent subtrees.

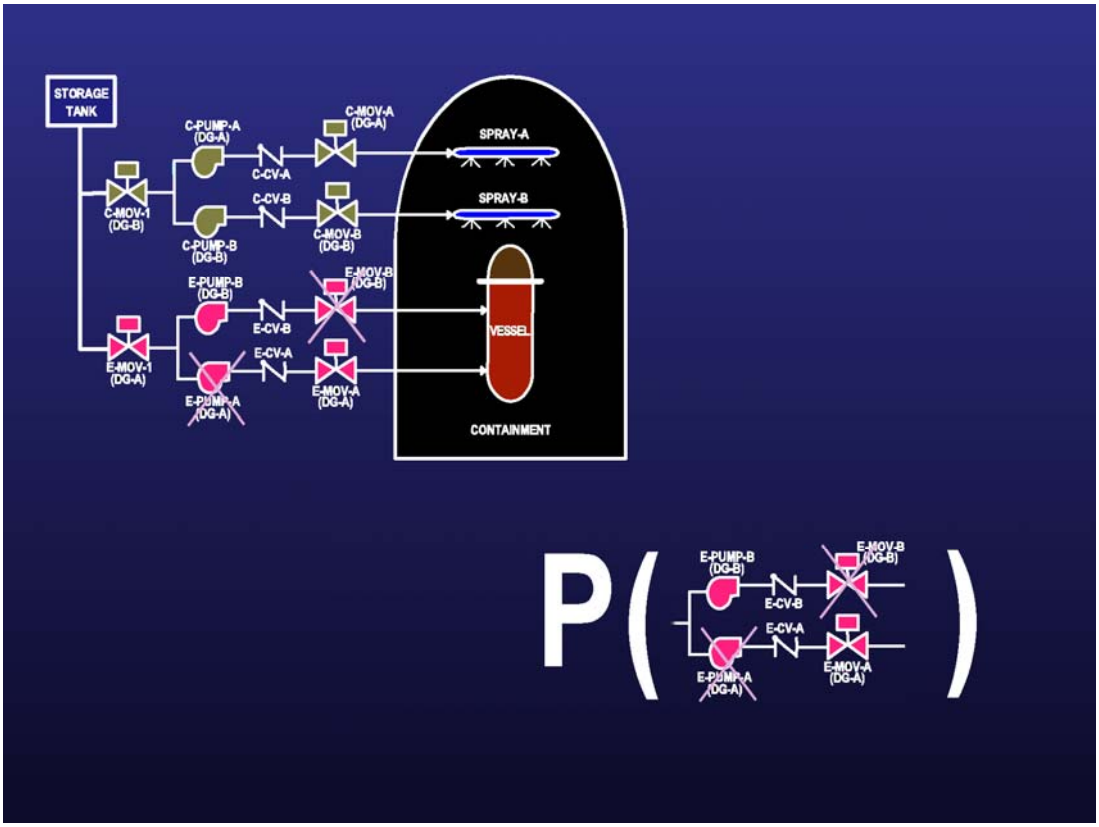
System	AND	Train-A	Train-B
Train-A	OR	A-1	A B
A-1	OR	C	D
Train-B	OR	H	

SOLVE TREE

- Fault tree reduction (see App. A, NUREG/CR-6116, Vol. 1)
  1. Gate expansion.
  2. Boolean absorption.
- Perform cut set truncation.

8 QUANTIFYING FAULT TREE CUT SETS

Section 8 describes the process of quantifying fault tree cut sets. Included in the discussion is a review of the minimal cut set upper bound approximation and details of the Min/Max option. The Min/Max option quantifies existing cut sets using an "exact" calculation for the union of the cut sets.



8.1. Cut Set Quantification Approaches

In general, there are many ways to quantify minimal cut sets. But, it is standard to use one of three methods, which include

1. Rare event approximation. - This calculation approximates the probability of the union of minimal cut sets. The equation for the rare event approximation is:

$$P = \sum_{i=1}^{m} C_i$$

where P is the probability of interest, C_i is the probability of the i'th cut set, and *m* is the total number of cut sets.

2. Minimal cut set upper bound - This calculation approximates the probability of the union of minimal cut sets. The equation for the minimal cut set upper bound is

$$P = 1 - \prod_{i=1}^{m} (1 - C_i)$$

where P is the probability of interest, C_i is the probability of the i'th cut set, and m is the total number of cut sets. Note (1) that the capital pi symbol implies multiplication and (2) most analysis tools utilize this equation as the default method of quantification.

3. Exact - There are various methods of determining the exact probability given a set of cut sets. One approach, referred to as the "inclusion-exclusion rule," goes by the name “Min/Max” within SAPHIRE.

8.2. The Min/Max Approach to Quantifying Cut Sets

The Min/Max quantification option quantifies the current case cut sets using the "exact" probability quantification algorithm.

To quantify the union of events, the first pass consists of adding the events, the second pass consists of subtracting pairs of events, the third pass consists of adding "triples", and so on.

For a simple example, assume that a fault tree X has only three cut sets which are the union of A, B, and C; which can be expressed as $A \cup B \cup C$.

In this example, each cut set consists of only one term; however, the approach is not limited to one term per cut set. For 3 passes, the exact solution is

$$X = (A + B + C) - (A * B + A * C + B * C) + (A * B * C).$$

Note that the Min/Max algorithm applies the Boolean idempotent law ($A * A = A$) to reduce identical terms during the multiplication of cut sets.

To obtain the probability of X, one simply evaluates the expression above with the individual event probabilities.

It is useful to compare the Min/Max algorithm to the Minimal Cut Set Upper Bound algorithm. The results are usually quite close; however, the Minimal Cut Set Upper Bound will be the more conservative estimate when the cut set probabilities are high (e.g., greater than 0.1) or when complemented events appear in the cut sets.

For our example above, the minimal cut set upper bound approximation for fault tree X is

$$Prob(X) = 1 - [1 - Prob(A)][1 - Prob(B)][1 - Prob(C)] .$$

Lastly, the rare event approximation for X is simply

$$Prob(X) = Prob(A) + Prob(B) + Prob(C) .$$

The equation used for the Min/Max quantification depends on the number of passes (which is user defined). To get the exact answer, the number of passes must be equal to the number of cut sets, but depending on the number of cut sets, this calculation may be intractable.

As an example of the numerical results, we return to our example where Prob(A) = Prob(B) = Prob(C) = 0.8. The minimal cut set upper bound approximation would be $1 - (1 - 0.8)^3 = 0.992$. The Min/Max calculation is shown below, where the results are displayed for each pass. Note that the Min/Max and the minimal cut set upper bound will be equal when the cut sets do not contain common events.

# of Passes	Min/Max Equation	Min/Max Probability
1	A + B + C	2.4
2	A + B + C - (A * B + A * C + B * C)	2.4 - 1.92 = 0.48
3	A + B + C - (A * B + A * C + B * C) + (A * B * C)	2.4 - 1.92 + 0.512 = 0.992
4	A + B + C - (A * B + A * C + B * C) + (A * B * C)	2.4 - 1.92 + 0.512 = 0.992

8.3. Using the Min/Max Quantification Option

Cut sets for the fault tree or sequence selected must have already been generated.

The number of passes must be selected by the user.

- The number of passes required for convergence is a function of the number of cut sets for the selected fault tree or sequence and the value of the basic events included in the cut sets.

- Setting the number of passes equal to the number of cut sets for the selected fault tree or sequence will obtain the exact probability.

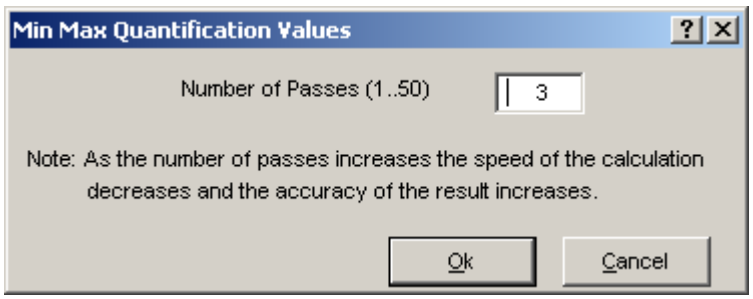
The computer run-time needs to be compatible with the user's needs. The Min/Max run-time is a function of the number of cut sets and the number of passes.

To analyze fault tree cut sets, select the **Fault Tree** menu. (Similarly, to analyze sequence cut sets, select the **Sequence** menu.)

Highlight the desired fault trees (or sequences).

Select the **Quantify → Min/Max** option.

Enter the number of passes and select Ok.

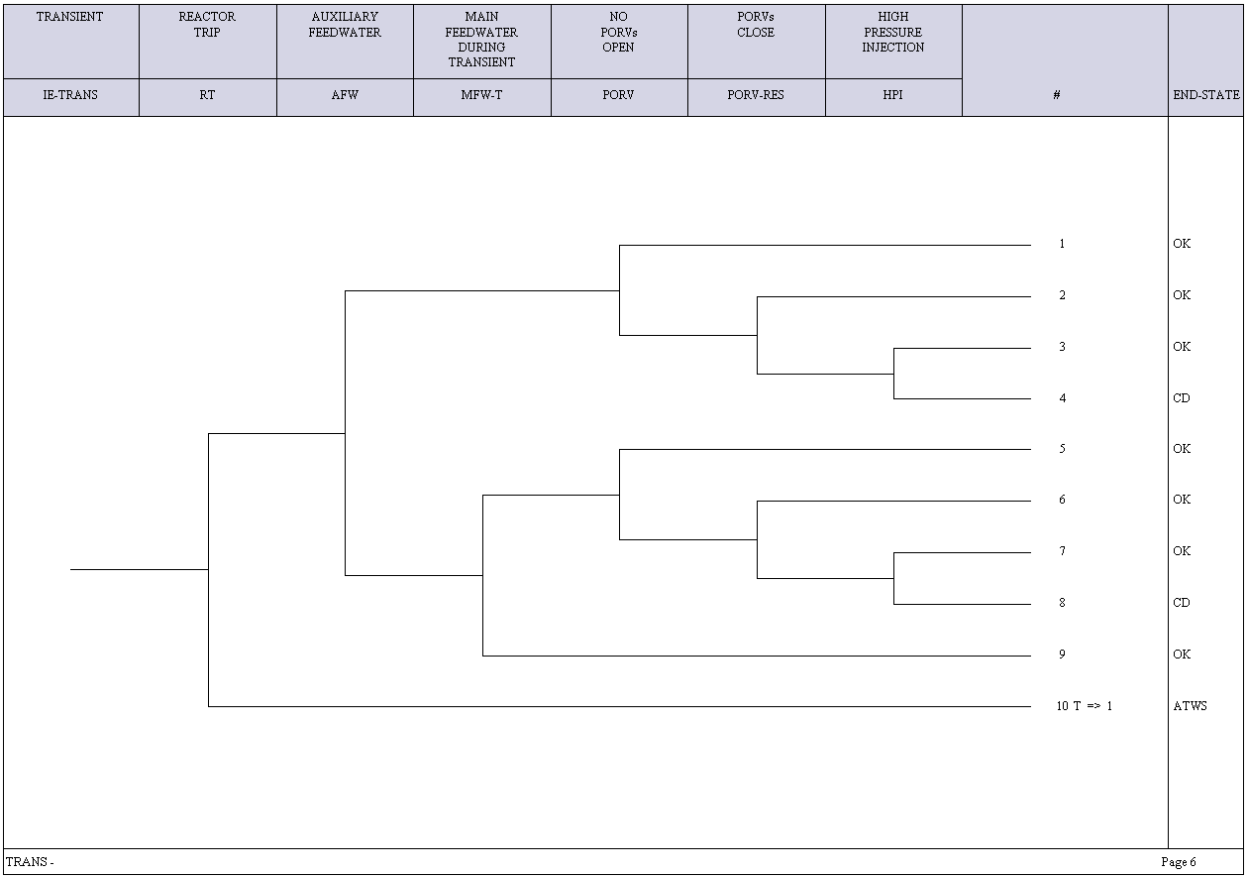


◊ Min/Max Quantification results are only displayed on the screen.

◊ To change the default quantification, use the **Modify → Fault Tree** option. If the Min/Max is selected as the default quantification method for a fault tree, this routine will be used each time minimal cut sets are solved for that fault tree.

9 SOLVING EVENT TREE CUT SETS

Section 9 describes how to solve for event tree cut sets. Model preparation prior to generating cut sets is discussed. Also, the fault tree linking approach is addressed. Uses of process flags, “dynamic” flag sets, and traditional flag sets are also presented.



9.1. Solving Sequence Cut Sets

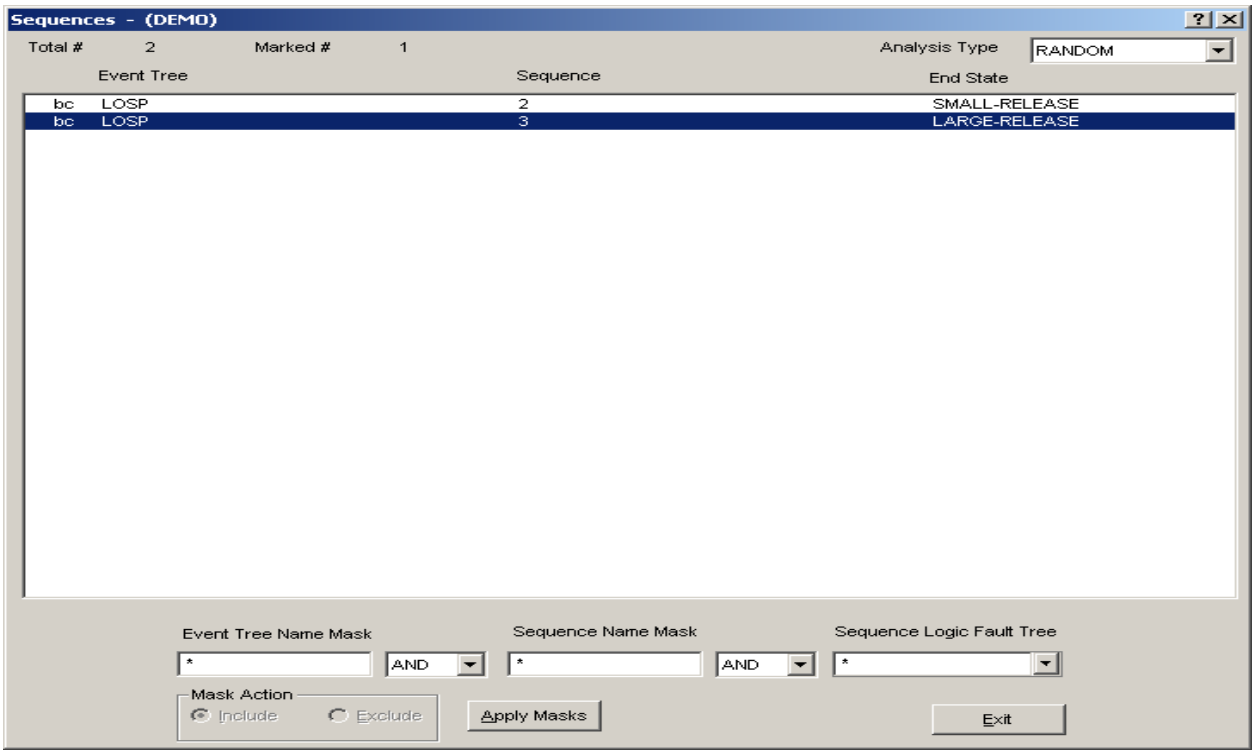
Sequence cut sets are derived from both the fault tree and event tree logic.

Prerequisites that are required prior to solving for sequence minimal cut sets are:

1. Fault tree and event tree logic was created by using the graphics or logic editors (or loaded into the database via the MAR-D interface).
2. Basic event data were added through the **Modify** → **Basic Events** menu.
3. Current case data were prepared for by using the **Generate** option.
4. Sequence logic was generated via the **Event Tree** → **Link Trees** option.

Menus and options for sequence cut set solving.

- Select the **Sequence** option from the menu.

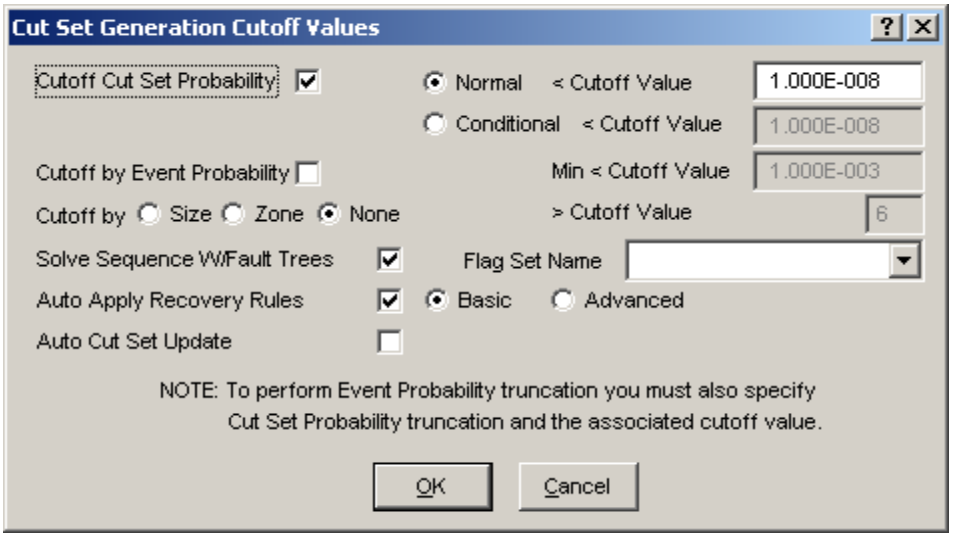


- Mark the Sequences using the mask feature, or using the mouse.
- Right-click to invoke the pop-up menu.
- Select the **Solve** option.

b – flag indicates the sequence has base case cut sets
c – flag indicates the sequence has current case cut sets

Truncation Parameters

Select the desired truncation parameters on the dialog, and choose **OK** to begin generating cut sets.



Cutoff by Cut Set Probability - If you select this check box, then cut sets below the cutoff value will not be retained. Choose one of the radio buttons:

> ***Conditional*** – uses the cutoff value in the "***"< Cutoff Value"*** field, but assumes that each initiating event has a value of one (just to solve cut sets). Note that the correct initiating event frequency will be used to quantify the cut sets.
>
> ***Normal*** – uses the cutoff value in the ***"< Cutoff Value"*** field divided by the initiating event frequency.

Cutoff by Event Probability - If you select this check box, then you must also select the ***Cutoff by Cut Set Probability*** check box. This option will retain cut sets comprised of basic events that are above the ***Min < Cutoff Value*** even if the cut set is below the ***Global Cutoff Value***. This option is generally not used.

Cut Set Size - If you select this check box, then cut sets having more events than specified in the ***> Cutoff Value*** field will not be retained. If you select the ***Zone*** check box, then cut sets having more Zone Flagged Events than specified in the ***> Cutoff Value*** field will not be retained. If neither check box is selected, then the number of events in a cut set will not affect whether the cut set is retained or discarded. This option is generally not used.

Solve Sequences w/ Fault Trees – This box is normally checked, indicating that the fault tree logic associated with the sequence will be used to determine the resulting cut sets. If this option is unchecked, then SAPHIRE will use existing fault tree cut sets (instead of the logic) to determine the sequence cut sets (note that this approach generally takes a much longer time than using fault tree logic).

Flag Set Name – If you leave the field blank, the sequence-specific flag set, if any, will be used. If you specify a flag set, that flag set will be used during processing. This option is generally not used.

Auto Apply Recovery Rules – If you check box, any recovery rules associated with this sequence will automatically be applied after the sequence cut sets have been generated. (The default is the **Basic Rules** which are the rules discussed in this workbook and the rules predominantly used.)

Auto Cut Set Update – If this box is checked, then the existing current case cut sets are reevaluated to remove all non-minimal cut sets and the fault tree probability is requantified.

Selecting the Appropriate Sequence Analysis Option

Solve — This option uses the sequence logic, including the associated fault tree logic. The sequence cut sets are quantified using the minimal cut set upper bound approximation (or the default method specified for the sequence). This option is appropriate for all sensitivity studies; however, it will take longer than the Cut Set Update or Quantification options.

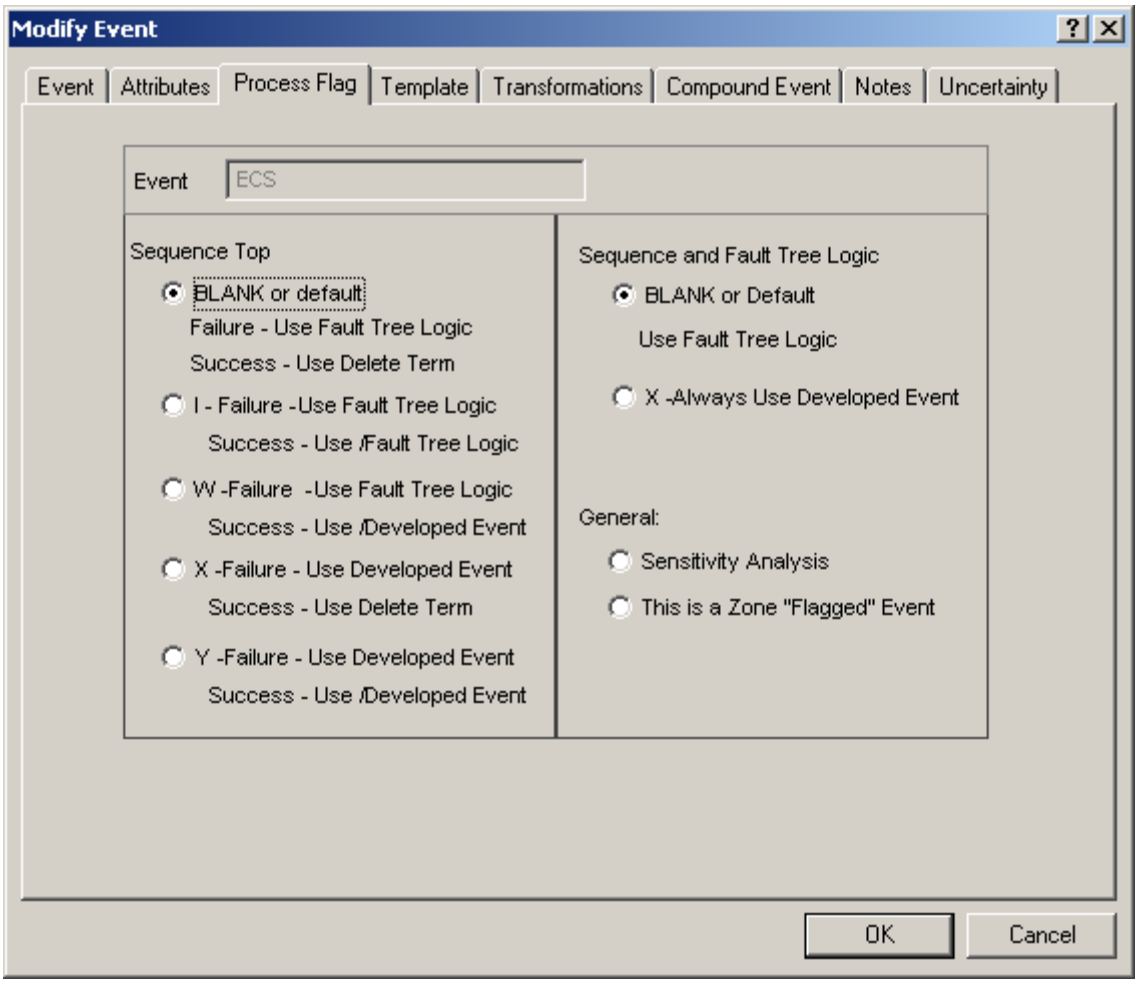
Cut Set Update (under **Cut Set → Update**) — This option uses existing sequence cut sets. Non-minimal cut sets are eliminated and the sequence probability is quantified using the minimal cut set upper bound approximation (or the default method). Do not use this option if fault tree logic is modified, event probabilities are increased, or the truncation is lowered from when the existing cut sets were solved

Quantification (under **Quantify → *method***) — This option uses the existing sequence cut sets and requantifies the probabilities using the minimal cut set upper bound approximation, rare event approximation, or min-max equation. (Note: if data changes increase the failure probability of an event, the Solve option should be used instead.)

9.2. Process Flags and Sequence Cut Set Generation

In event trees, Process Flags are special identifiers that tell SAPHIRE how to treat top events in various ways. For example, SAPHIRE has one Process Flag that uses a top event as a split-fraction probability rather than as a link to its fault tree logic.

The process flag is entered in the **Modify → Basic Event** option. Once in that option, highlight the fault tree top event to be modified, click the right mouse button, select **Modify**, and then click the **Process Flag** tab.



- When evaluating event tree accident sequences, you would modify the process flags for the event tree top events. Recall that both fault tree and event tree top events show up in the list of basic events.

The process flag field is one character long (I, W, X, or Y) and is indicated via a radio button. The process flag has different characteristics depending on the sequence branch path (recall that an up branch is success while a down branch is failure).

"Sequence" Process Flags

Flag	Use on failure branches	Use on success branches
" " (a space) This is the default process flag.	**Failure** - Use system logic Use fault tree logic (if available) for the top event. If fault tree logic is not present, then use the developed event probability.	**Success** - Use the "delete term" Use the "delete term" process to eliminate failure cut sets based on the event tree success event(s). The "delete term" process looks for, and removes, impossible cut sets from the analysis.
I	**Failure** - Use system logic Use fault tree logic (if available) for the top event. If fault tree logic is not present, then use the developed event probability.	**Success** - Use the complement of the logic Use the complement of the system logic for the successful branch. SAPHIRE will then treat the success tree as part of the sequence cut set solving process. Note that (1) this calculation may take a long time and (2) SAPHIRE does not perform the Boolean operation A*B + A*/B = A.
W	**Failure** - Use system logic Use fault tree logic (if available) for the top event. If fault tree logic is not present, then use the developed event probability.	**Success** - Use the complement of the developed event Use the complement of the developed event (i.e., one minus the probability specified for the top event).
X	**Failure** - Use a developed event Use a basic event (named the same as the top event) instead of fault tree logic. The user must specify the failure probability of the top event.	**Success** - Use the "delete term" Use the "delete term" process to eliminate failure cut sets based on the event tree success event(s). The "delete term" process looks for, and removes, impossible cut sets from the analysis.
Y	**Failure** - Use a developed event Use a basic event (named the same as the top event) instead of fault tree logic. The user must specify the failure probability of the top event.	**Success** - Use the complement of the developed event Use the complement of the developed event (i.e., one minus the probability specified for the top event).

- Any combination of top events with process flags could be used as needed. But, care should be taken since some combinations of process flags could result in questionable results.

- Example: If an event tree top event is treated as a basic event (via the **Y** process flag) but is not independent of other top events, it is possible to obtain non-conservative results due to double counting of basic events.

- The " " (space) process flag is generally the most commonly used flag since this is the default flag and meets the needs of most applications.

- The **I** process flag is used when the analyst wants to see the success basic events in the cut sets.

- The **Y** process flag is used when the analyst only wants to use a split fraction for the top event. Note that in the next section, the "large event tree methodology," we will demonstrate a technique for using split-fractions for each top event in the event tree.

- The **W** and **X** process flags are not used that often when solving sequence cut sets.

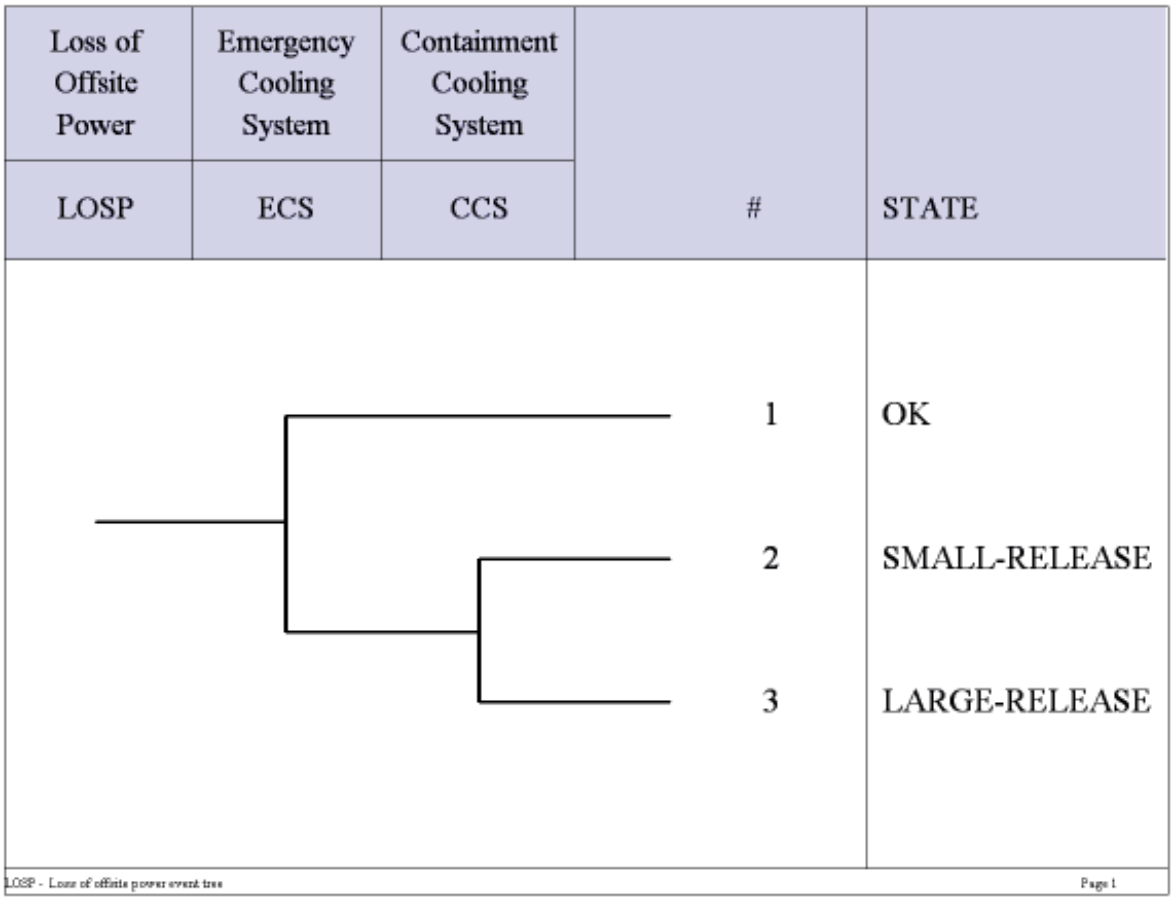
9.3. Process Flag Example

Once the process flags have been defined for the top events and the changes generated (via the **Generate** option), sequence cut sets can be solved by using the **Sequences** → **Solve** option.

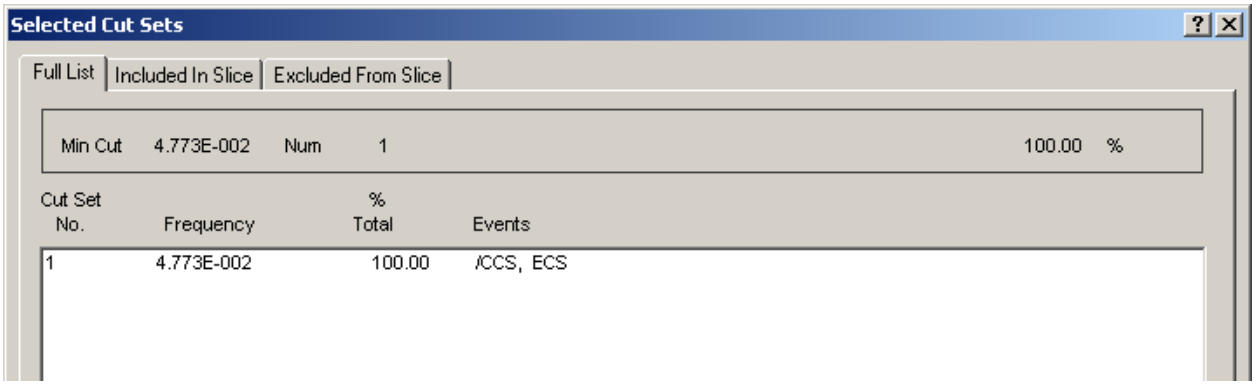
- The LOSP event tree will be used to demonstrate how process flags operate.

  - Modify the process flag (via a change set) for both CCS and ECS to **Y**. Also, set the CCS and ECS developed events to a probability of 2.12E-2 (representing the individual system failure probabilities).

  - Generate the data changes using the **Generate** option.

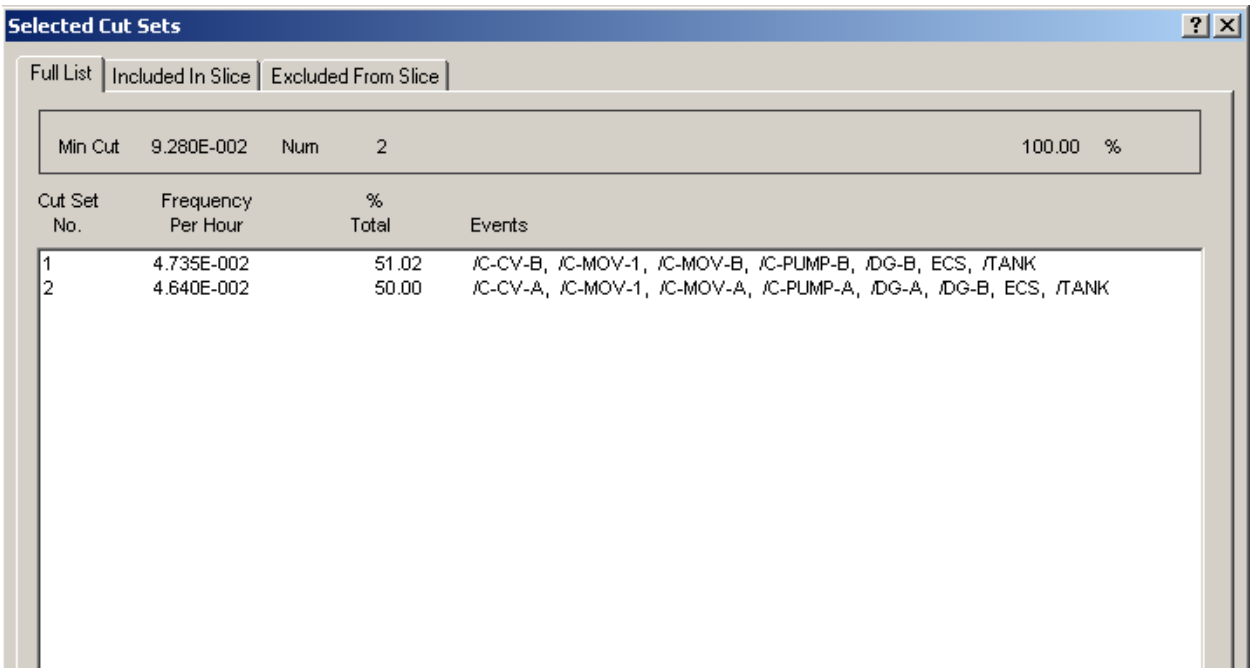


When both top events CCS and ECS have their process flags set to **Y**, the sequence cut set solve option will yield the cut set below for sequence 2.



◊ Now, modify the process flag (via a second change set) for both CCS and ECS. First, set CCS to a process flag of **I** and then set ECS to a process flag of **Y**. Also, set the ECS developed event to a probability of 2.12E-2.

◊ Generate the data changes using the **Generate** option. The sequence cut set solve process will yield the cut sets below for sequence 2.



Note: The original frequency (i.e., calculated without process flags) for sequence 2 was found to be 4.84E-2

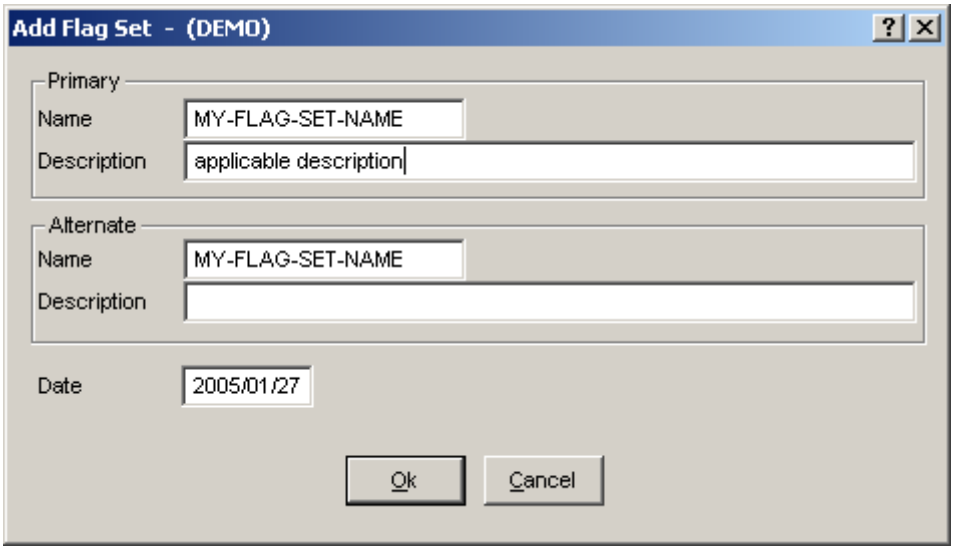
◊ There is a large difference between the sequence 2 frequency calculated with process flags and the original sequence frequency.

◊ Not accounting for the dependencies between ECS and CCS results in a non-conservative sequence frequency.

9.4. Flag Sets and Sequence Cut Set Generation

First, let us present a brief review of Flag Sets.

◊ Flag Sets are a special type of change set. SAPHIRE will keep flag sets separate from change sets by specifying it as a flag set. Flag sets can be created under **Modify** → **Flags** menu.



◊ Flag Sets can only contain individually selected changes. No "Class Changes" are allowed in a Flag Set.

◊ Flag Sets are used to indicate modifications to particular events on a sequence-by-sequence basis (or to events in specific fault trees).

◊ The *probability* of failure may *not* be changed in a Flag Set.

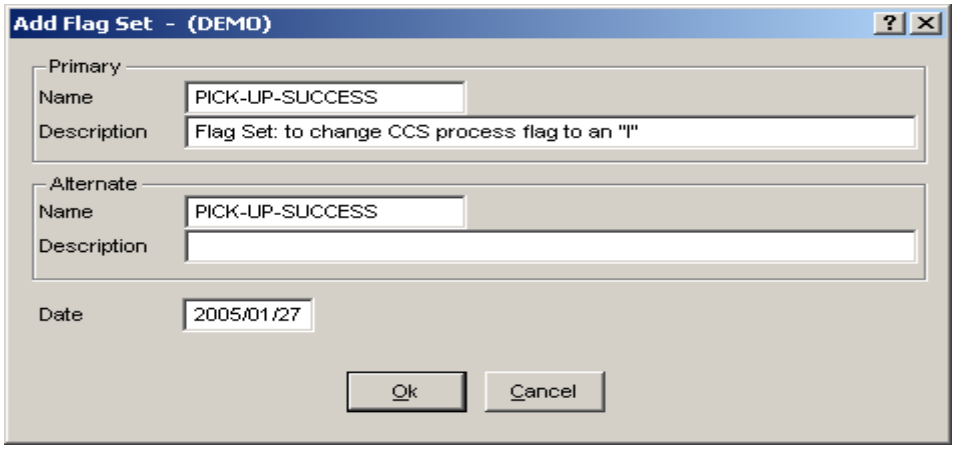
When generating sequence (or fault tree) cut sets, Flag Sets are used for one of two purposes.

◊ Setting house events, basic event, or top events to TRUE, FALSE, or IGNORED.

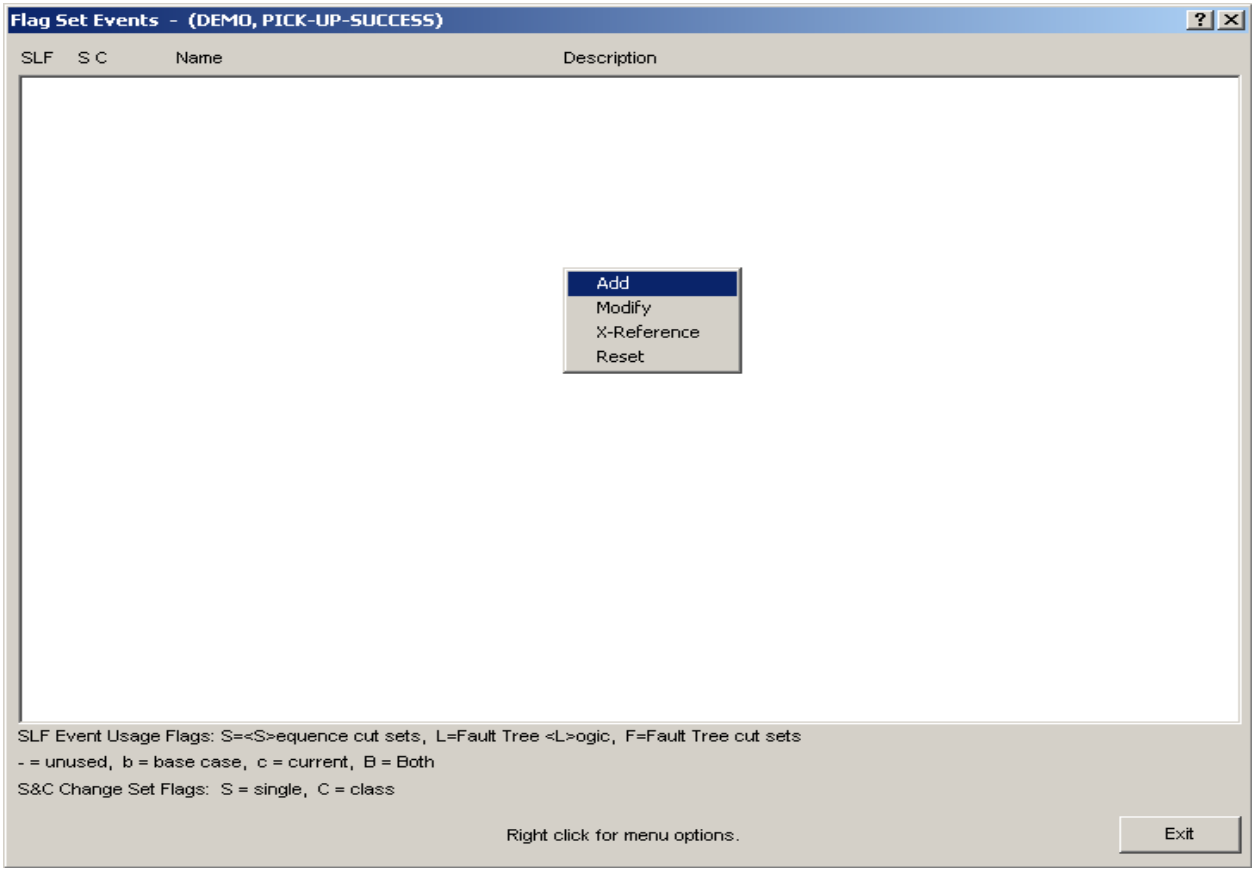
◊ Modifying the top event Process Flags from its default condition.

Therefore, Flag Sets can only contain house event changes (T, F, or I) to the calculation type or changes to the Process Flag (space, **I**, **W**, **X**, or **Y**).

- To make a Flag Set
  - ◊ Enter the **Modify** → **Flags** menu.
  - ◊ Right click the mouse and select **Add**, and enter the Flag Set name and description.

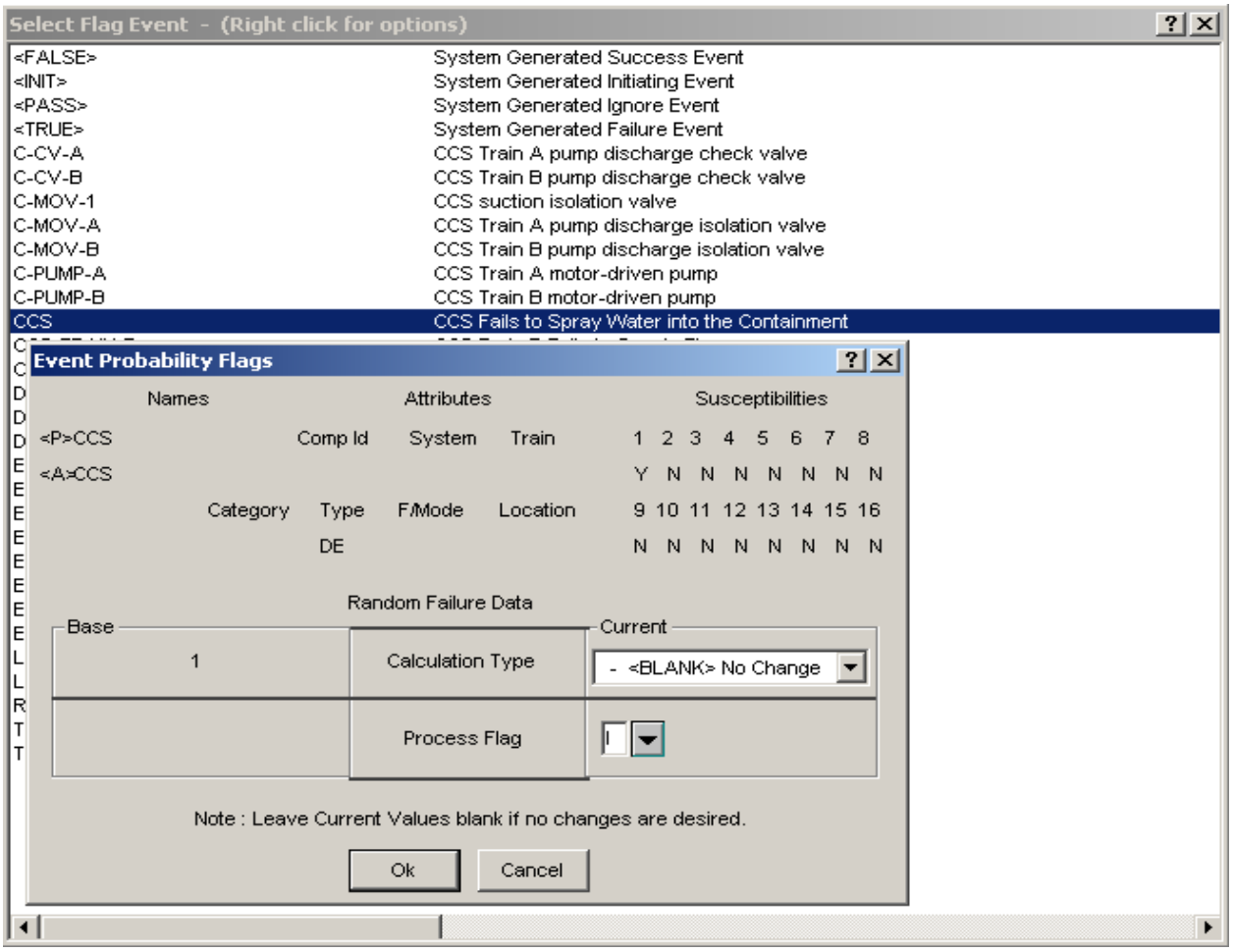


  - ◊ Highlight the Flag Set, select the Flags button on the bottom of page and then right click the mouse and select **Add**.



◊ Highlight the event to be modified, right click the mouse and select **Add**.

◊ Modify either the calculation type or the process flag.



♦ To use a Flag Set after it has been created the Flag Set name must be assigned to a sequence or sequences.

♦ To assign the Flag Set to a sequence

◊ Use the **Modify → Event Trees** option, then highlight the event tree and click the **Sequence** button on the bottom of page.

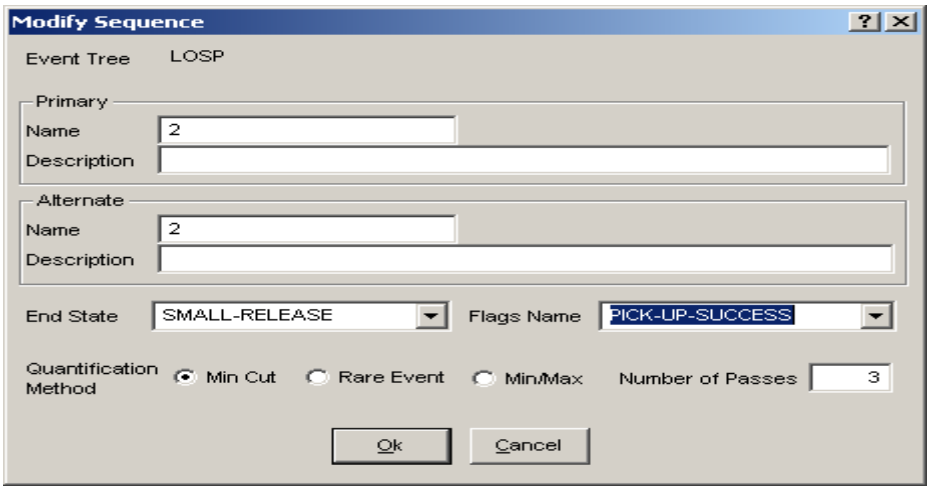
◊ Highlight the sequence, right click the mouse and select **Modify**. The Flag Set name is entered in the field labeled "Flags Name."

Note: The Flag Set name is limited to 24 characters in length.

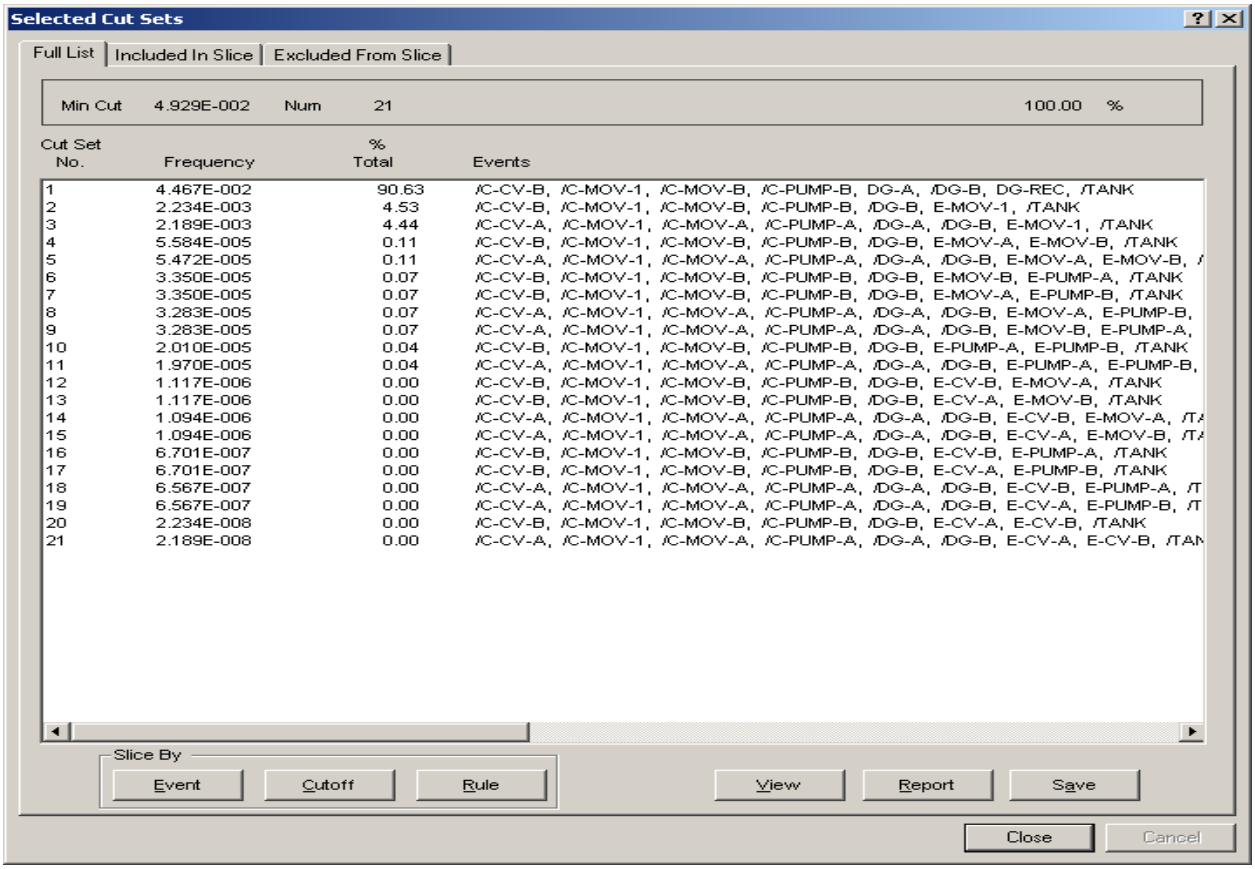
To illustrate the use of Flag Sets, the DEMO project will be used.

- A Flag Set named "PICK-UP-SUCCESS" was created.
  - In this Flag Set, the process flag for top event CCS was changed to an **I**.

- The Flag Set "PICK-UP-SUCCESS" was assigned to sequence 2.



- The resulting cut sets for LOSP sequence 2 are shown below. Notice that success cut sets from the CCS logic now appears in the list of cut sets.



9.5. "Dynamic" Flag Sets and Sequence Cut Set Generation

"Dynamic" Flag Sets are a special type of Flag Set that is assigned to sequences by the use of event tree rules. "Dynamic" Flag Sets are named such since the flag set is created "on-the-fly" based upon a special type of Linking Rule.

- A Dynamic Flag Set is assigned to a sequence(s) if the search criteria in the rule are met.

- The advantages of using Dynamic Flag Sets are:

  - If event tree logic changes are made (e.g., sequences are added or deleted), then the proper flag sets will be applied to new sequences automatically. Otherwise, the analyst would have to manually assign flag sets to the applicable accident sequences.

  - The flag set does not first need to be created. Instead, a rule can be used to change a calculation type to TRUE, FALSE, or IGNORE.

- Dynamic Flag Sets are treated the same as Flag Sets when solving cut sets. For example, changes can only be specified to individual basic events (i.e., no class changes).

- No probability changes can be made with Dynamic Flag Set.

- Dynamic Flag Sets can only contain house event changes to the calculation type for an event.

Type of Change	Allowable Values
Calculation type	T (TRUE) F (FALSE) I (IGNORE)

- Dynamic Flag Sets will appear in the list of Flag Sets after the flag set rule is applied. However, the name given to a Dynamic Flag Set is in a form such as ET-000001-000001. This name is based upon the event tree, sequence name, and number of Dynamic Flag Sets already created.

"Dynamic Flag Set" Rule Nomenclature and Structure

Dynamic Flag Set rules are created by using the Linking Rule editor (see Section 4).

If linking rules are written for Dynamic Flag Sets, SAPHIRE searches the event tree logic for the search criteria specified in the rule and assigns the Dynamic Flag Set to sequences as dictated by the rule. This process takes place only during the "link" step (see Section 4.1).

Dynamic Flag Set Rule Structure (Example 1 – Setting an event to TRUE)

| The "if-then" rule structure for creating Dynamic Flag Sets:
| This rule sets E-MOV-A and E-PUMP-A to TRUE only if top event ECS fails in
| the LOSP event tree sequence.

```
if  ECS  then
  eventree(LOSP) = True(E-MOV-A, E-PUMP-A);
endif
```

| The rule above could have set the basic events in parenthesis to house events
| FALSE or IGNORE by replacing True with either False or Ignore, respectively.

Dynamic Flag Set Rule Structure (Example 2 – Using an existing flag set)

| The "if-then" rule structure can be used to assign an existing Flag Set to a sequence.
| Note that the Flag Set must be created prior to solving by using the **Modify → Flags**
| option.
|
| This rule adds the Flag Set "FLAG-SET-1" to the sequence(s) that meets
| the criteria specified (failure of ECS).
|

```
if  ECS  then
  eventree(LOSP) = flag(FLAG-SET-1);
endif
```

Dynamic Flag Set Rule Structure (Example 3 – Assigning an end state)

| The rule structure can be used to add an already created end state for the sequence cut sets that meet the specified search criteria.

| This rule adds the end state “ECS-END” to the sequence(s) meeting the criteria specified (failure of ECS).

|

```
if  ECS  then
  eventree(LOSP) = endstate(ECS-END);
endif
```

Note: The Dynamic Flag Set is designed to assign a Flag Set to the sequence meeting the search criteria even if the specified event tree transfers to a subtree.

Therefore, either rule below will append a Flag Set to the same sequence, which in this case is any sequence that has an initiator named “IE-NAME.”

```
if  init(IE-NAME)  then
   eventree(main event tree) = True(event1);
endif
```

or

```
if  init(IE-NAME)  then
   eventree(subtree) = True(event1);
endif
```

9.6. Dynamic Flag Set Keywords and Nomenclature

Each of the “rules” in SAPHIRE (e.g., linking, recovery, and partition) has their own nomenclature. The table below lists the keywords available for Dynamic Flag Set rules.

Keyword or symbol	Definition	Usage

if then	Keyword that indicates search criteria is being specified.	if "search criteria" then
		perform some action on the sequence;
		endif

endif	Keyword that indicates the end of a particular rule.	if "search criteria" then
		perform some action on the sequence;
		endif

else	Keyword that specifies some action to be taken if all the search criteria are not met. The else should be the last condition in the recovery rule.	if "search criteria" then
		perform some action on the sequence;
		else
		perform some other action on the sequence if the search criteria not met;
		endif

elsif	Keyword that specifies an alternative search criteria. Any number of elsifs can be used within a recovery rule.	if "search criteria" then
		perform some action on the sequence;
		elsif "2nd search criteria" then
		perform some other action on the sequence;
		elsif "3rd search criteria" then
		perform some other action on the sequence;
		endif

always	Keyword that indicates that every sequence that is being evaluated satisfies the search criteria.	if always then
		perform some action on the sequence;
		endif

init( )	Keyword used in the search criteria to indicate that a sequence has a particular initiating event.	if init(INITIATOR-NAME) * "other search criteria if needed" then
		perform some action on the sequence;
		endif

~	Symbol used in the search criteria to indicate that a particular system will not be in the sequence that is being evaluated.	if (~SEARCH-CRITERIA) * "other search criteria if needed" then ... The search criteria will be satisfied for all sequences that do not contain SEARCH-CRITERIA (and also contains the optional "other search criteria"). SEARCH-CRITERIA may be an initiating event, fault tree, or macro.

Keyword or symbol	Definition	Usage
/	Symbol used to represent a complemented event (i.e., the success of a system).	if (/SYSTEM) * "other search criteria" then The search criteria will be satisfied for all sequences that contain the complement of SYSTEM (and also contains the optional "other search criteria").
|	Symbol used to represent a comment contained in the rules. Everything on a line to the right of this symbol will be ignored by the rule compiler.	| Place your comments here! | Note that blank lines are also permissible!
;	Symbol to indicate the end of a macro line or a line that modifies the cut set being evaluated.	| usage for a macro command MACRO-NAME = "search criteria" ;
*	Symbol to indicate the logical AND command.	if SEARCH-CRITERIA1 * SEARCH-CRITERIA2 then The search criteria will be satisfied for all sequences that match SEARCH-CRITERIA1 and SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be an initiating event, fault tree, or macro.
+	Symbol to indicate the logical OR command.	if CRITERIA1 + CRITERIA2 then The search criteria will be satisfied for all sequences that match either CRITERIA1 or CRITERIA2. The CRITERIA# may be an initiating event, fault tree, or macro.
( )	Symbols to indicate a specific grouping of items.	if (A + B) * (C + D) then The search criteria above would return all sequences that contain: [A * C], [A * D], [B * C], or [B * D].

True( )	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to TRUE for the applicable sequence. Multiple basic events should be separated using commas.	if "search criteria" then
		eventree(ET-NAME) = True (EVENT1,
		EVENT2, EVENT3, …);
		endif

Keyword or symbol	Definition	Usage

False( )	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to FALSE for the applicable sequence. Multiple basic events should be separated using commas.	if "search criteria" then
		eventree(ET-NAME) = False (EVENT1,
		EVENT2, EVENT3, …);
		endif

Ignore( )	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to IGNORE for the applicable sequence. Multiple basic events should be separated using commas.	if "search criteria" then
		eventree(ET-NAME) = Ignore (EVENT1,
		EVENT2, EVENT3, …);
		endif

Flag( )	Keyword to assign an existing Flag Set to sequences meeting the search criteria.	if "search criteria" then
		eventree(ET-NAME) = Flag (FS-NAME);
		endif

Endstate( )	Keyword to assign a sequence meeting the search criteria to a particular end state.	If "search criteria" then
		eventree(ET-NAME) = endstate(ES-NAME);
		endif

MACRO	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 16 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
		if MACRO-NAME "and other search criteria" then
		perform some action...;
		endif
		
		|Macros are only applicable in the |particular rule they are entered into

To make a Dynamic Flag Set

- Enter the **Event Tree** option. Highlight the event tree, right click the mouse, and select **Edit Rules**.
- Using the rule structures discussed above, construct a rule that will modify a basic event's calculation type.
- Generate event tree sequences by highlighting the event tree, right click the mouse, and select **Link Trees**. When asked, select the applicable output option and click **OK**.

(Note: Event tree sequences must be generated for the Dynamic Flag Set to be appended to the sequence.) The Dynamic Flag Set will automatically be assigned to the sequence without having to manually modify the sequence.

♦ The event tree sequences are now ready to be analyzed in the Sequence menu option.

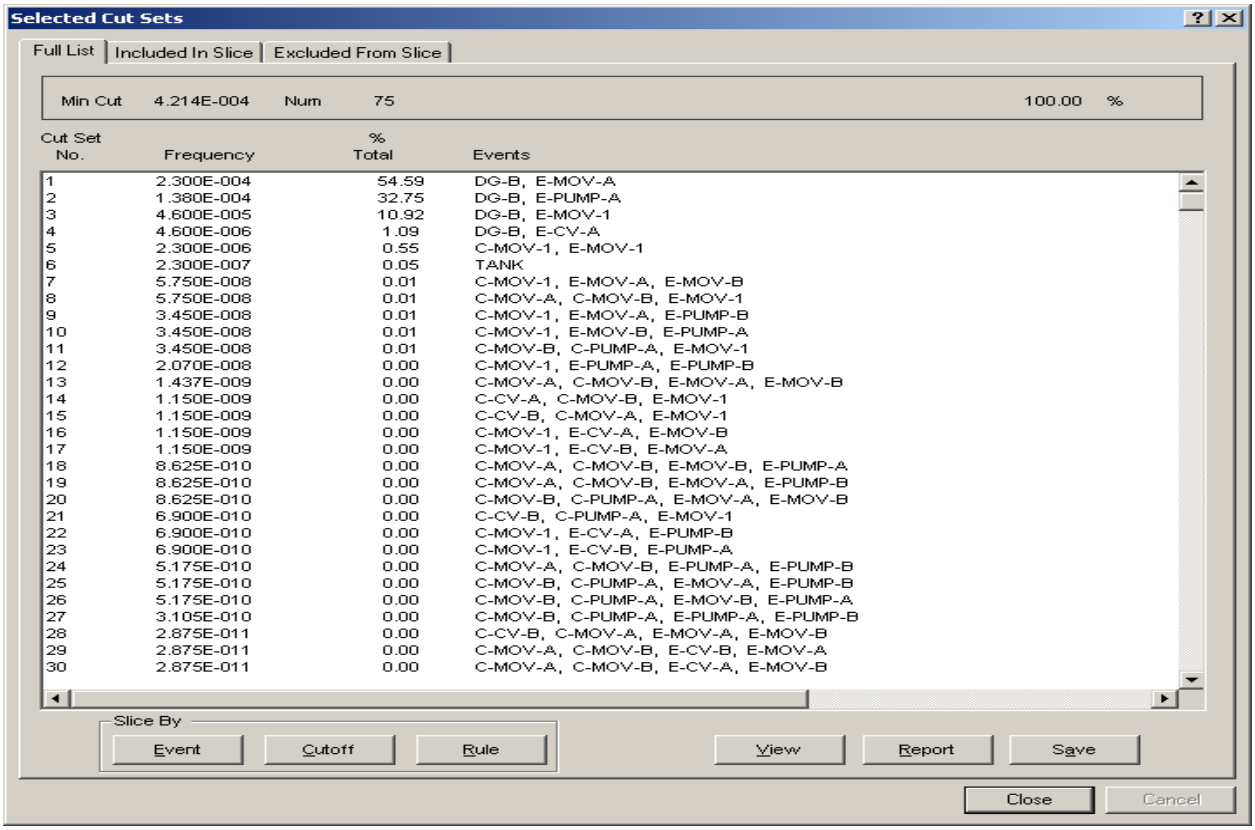
To illustrate the use of Dynamic Flag Sets, the DEMO project will be used

◊ A rule was entered to set DG-A and C-PUMP-B to a house event FALSE only if CCS fails in the LOSP event tree. The Link rule looks like:

if CCS then
eventree(LOSP) = False(DG-A, C-PUMP-B);
endif

◊ The Dynamic Flag Set will append itself to sequences meeting the rule search criteria. For this rule, only Sequence 3 will have the Flag Set associated with it since CCS fails only within this sequence.

◊ The resulting cut sets for Sequence 3 are shown below. Notice that basic events DG-A and C-PUMP-B *do not* show up in the cut sets.



Cut Set No.	Frequency	% Total	Events
1	2.300E-004	54.59	DG-B, E-MOV-A
2	1.380E-004	32.75	DG-B, E-PUMP-A
3	4.600E-005	10.92	DG-B, E-MOV-1
4	4.600E-006	1.09	DG-B, E-CV-A
5	2.300E-006	0.55	C-MOV-1, E-MOV-1
6	2.300E-007	0.05	TANK
7	5.750E-008	0.01	C-MOV-1, E-MOV-A, E-MOV-B
8	5.750E-008	0.01	C-MOV-A, C-MOV-B, E-MOV-1
9	3.450E-008	0.01	C-MOV-1, E-MOV-A, E-PUMP-B
10	3.450E-008	0.01	C-MOV-1, E-MOV-B, E-PUMP-A
11	3.450E-008	0.01	C-MOV-B, C-PUMP-A, E-MOV-1
12	2.070E-008	0.00	C-MOV-1, E-PUMP-A, E-PUMP-B
13	1.437E-009	0.00	C-MOV-A, C-MOV-B, E-MOV-A, E-MOV-B
14	1.150E-009	0.00	C-CV-A, C-MOV-B, E-MOV-1
15	1.150E-009	0.00	C-CV-B, C-MOV-A, E-MOV-1
16	1.150E-009	0.00	C-MOV-1, E-CV-A, E-MOV-B
17	1.150E-009	0.00	C-MOV-1, E-CV-B, E-MOV-A
18	8.625E-010	0.00	C-MOV-A, C-MOV-B, E-MOV-B, E-PUMP-A
19	8.625E-010	0.00	C-MOV-A, C-MOV-B, E-MOV-A, E-PUMP-B
20	8.625E-010	0.00	C-MOV-B, C-PUMP-A, E-MOV-A, E-MOV-B
21	6.900E-010	0.00	C-CV-B, C-PUMP-A, E-MOV-1
22	6.900E-010	0.00	C-MOV-1, E-CV-A, E-PUMP-B
23	6.900E-010	0.00	C-MOV-1, E-CV-B, E-PUMP-A
24	5.175E-010	0.00	C-MOV-A, C-MOV-B, E-PUMP-A, E-PUMP-B
25	5.175E-010	0.00	C-MOV-B, C-PUMP-A, E-MOV-A, E-PUMP-B
26	5.175E-010	0.00	C-MOV-B, C-PUMP-A, E-MOV-B, E-PUMP-A
27	3.105E-010	0.00	C-MOV-B, C-PUMP-A, E-PUMP-A, E-PUMP-B
28	2.875E-011	0.00	C-CV-B, C-MOV-A, E-MOV-A, E-MOV-B
29	2.875E-011	0.00	C-MOV-A, C-MOV-B, E-CV-B, E-MOV-A
30	2.875E-011	0.00	C-MOV-A, C-MOV-B, E-CV-A, E-MOV-B

9.7. Steps Used by SAPHIRE to Solve Sequences

SAPHIRE is designed to solve for minimal cut sets. The cut set solve process can occur for both sequences and fault trees. For sequences, several different methods exist.

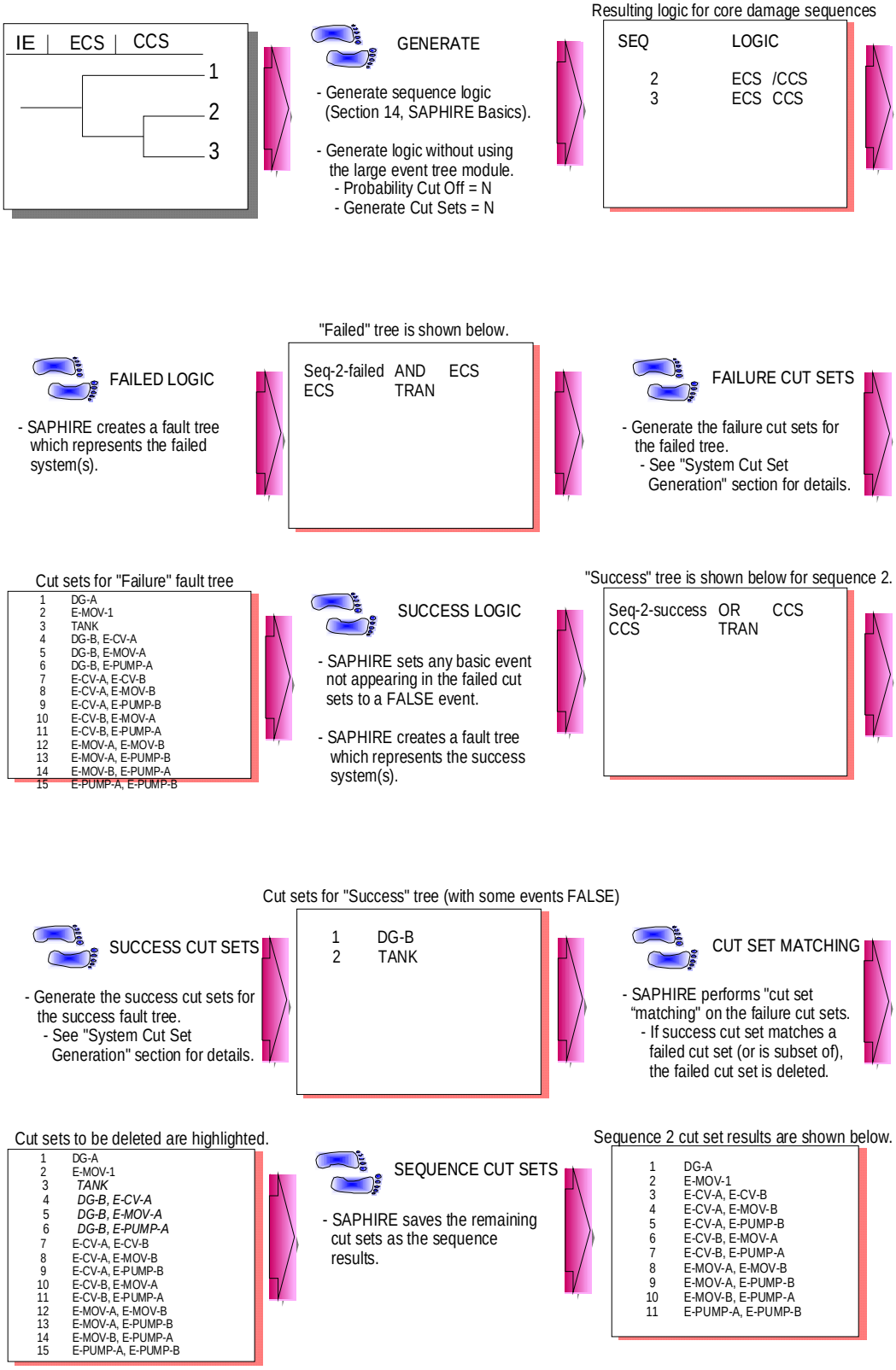
◊ Sequence cut set solving for the “fault tree linking” approach.

- Cut set generated using fault tree logic and "cut set matching" (i.e., delete term) method. This is the standard technique.
- Cut set generated using fault tree logic while solving sequence Boolean logic.
- Cut set generated using existing fault tree cut sets. This method is not used very often.

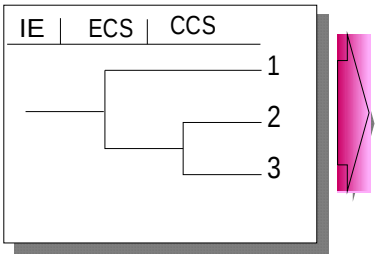
◊ Sequence cut set solving for the “large event tree” approach.

Additional technical details on cut set generation are contained in NUREG/CR-6116, Volume 1, Technical Reference Manual.

Sequence Cut Set Solving Using Fault Tree Linking ("Cut Set Matching")



Sequence Cut Set Solving Using Fault Tree Linking (Solve Full Logic)



GENERATE

- Generate sequence logic (Section 14, SAPHIRE Basics).
- Generate logic without using the large event tree module.
  - Probability Cut Off = N
  - Generate Cut Sets = N

Resulting logic for core damage sequences

SEQ	LOGIC	
2	ECS	/CCS
3	ECS	CCS

SEQUENCE LOGIC

- When using the "I" process flag, SAPHIRE creates a fault tree which represents both the failed and success system(s).

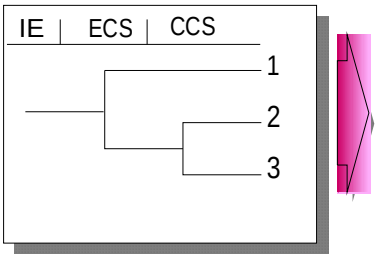
Sequence 2 fault tree is shown below.

Seq-2	AND	Failed	Success
Failed	AND	ECS	
Success	NOR	CCS	
ECS	TRAN		
CCS	TRAN		

SEQUENCE CUT SETS

- Generate the sequence cut sets for the fault tree.
  - See "System Cut Set Generation" section for details.
- Note: SAPHIRE does not perform A*/B + A*B = A Boolean operation.

Sequence Cut Set Solving Using Large Event Tree Method



GENERATE

- Generate sequence logic (Section 14, SAPHIRE Basics).
- To use the large event tree module, use
  - Probability Cut Off = Y
  - Generate Cut Sets = Y

Resulting cut sets for core damage sequences

SEQ	LOGIC	
2	ECS	/CCS
3	ECS	CCS

Note: Cut sets are generated for each sequence above the probability truncation level that was specified. Each sequence will contain one cut set. This cut set will be the combination of success and failure top events.

9.8. Example of Sequence and Fault Tree Flag Sets for Cut Set Solving

The example in this subsection will illustrate how to model changes in logic dependencies when analyzing event tree accident sequences. Two potential ways to handle a change in logic dependency include:

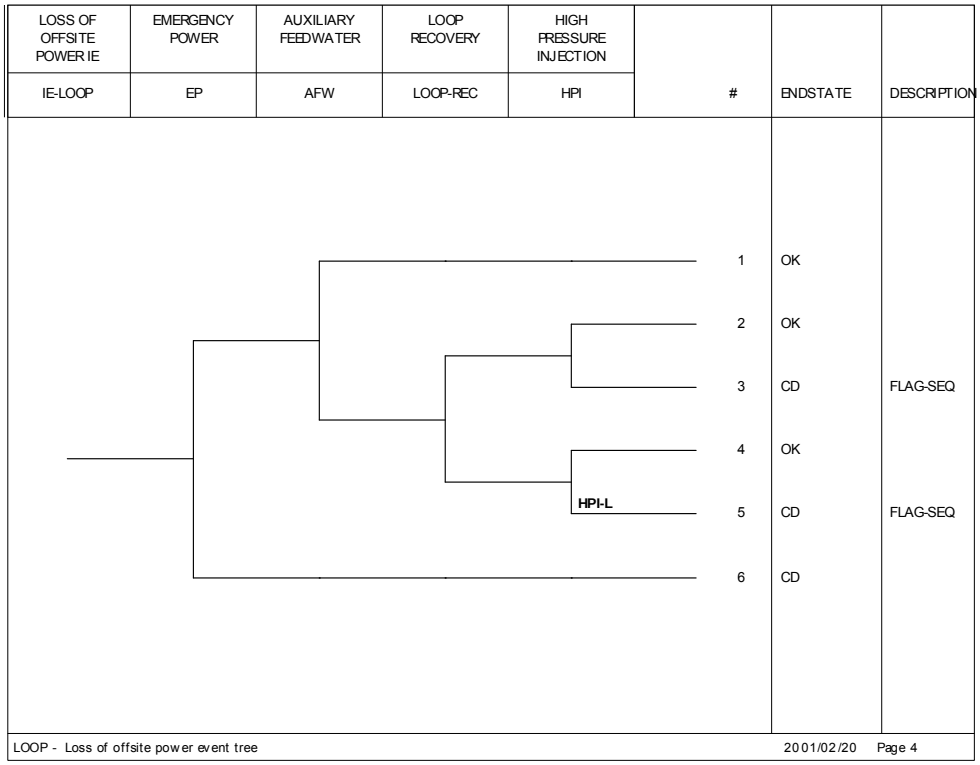
1. Use multiple fault trees
2. Use fault tree and sequence flag sets.

Both methods are discussed in this section with an emphasis on how SAPHIRE handles logic changes based upon fault tree and sequence flag settings.

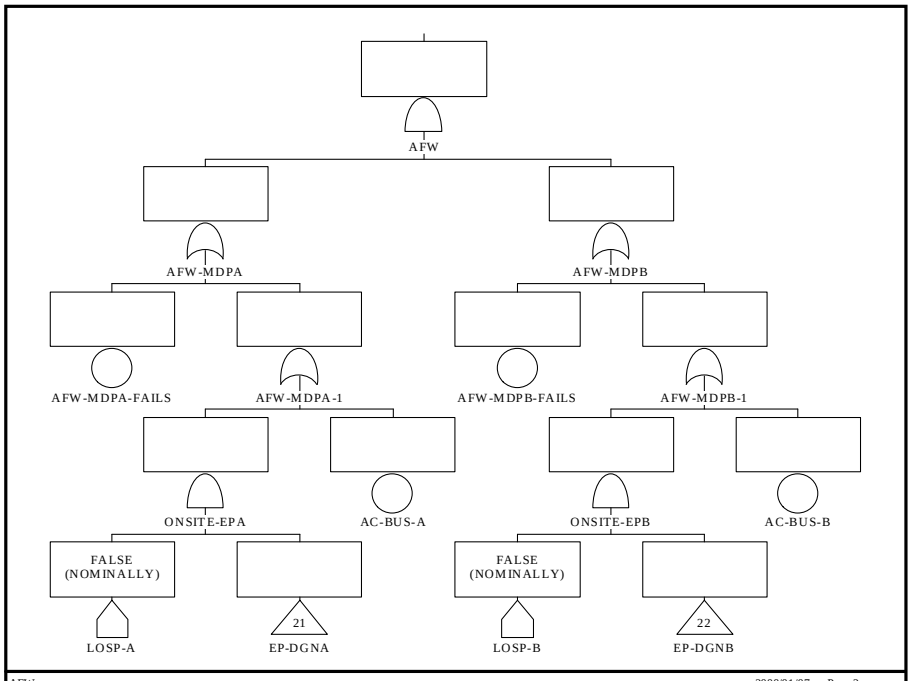
Note that both fault tree and sequence flag sets can be used simultaneously when analyzing event tree accident sequences.

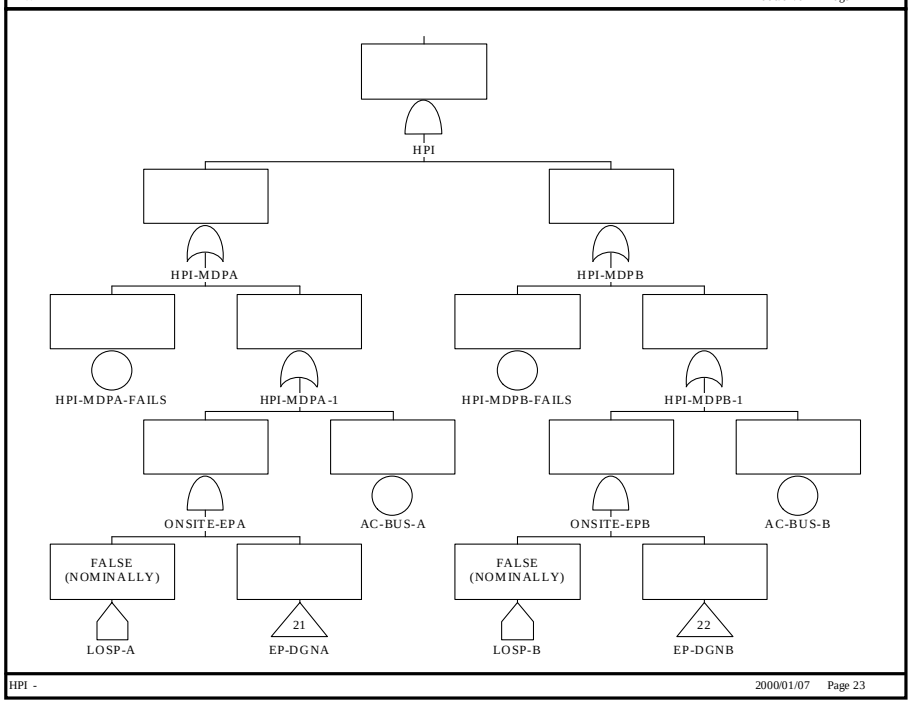
- If a given sequence contains a flag set and a fault tree in that sequence has a flag set, the *fault tree flag set* takes precedence over the sequence flag set.

The following event tree will be used to illustrate this example.



Sequences 3 and 5 contain a sequence flag set (FLAG-SEQ). This flag set changes events LOSP-A and LOSP-B to TRUE. These house events force the emergency diesel generators (EDG) to supply ac power to the auxiliary feedwater (AFW) and high pressure injection (HPI) pumps (in the fault tree logic).





From the event tree structure, we can see that sequences 3 and 5 are different because offsite power has been recovered (i.e., top event LOOP-REC) in sequence 3. The success or failure of this top event requires different logic for the HPI system.

After failure of LOOP-REC (i.e., no recovery), the HPI pumps depend on the EDGs for ac power. After success of LOOP-REC, the HPI pumps no longer depend on the EDGs for ac power.

- Analysis of sequence 5 is straightforward because there is no logic dependency change for the HPI pumps. Specifically, sequence 5 requires LOSP-A and LOSP-B to be TRUE (EDGs are required since offsite ac power was not recovered).

  No special treatment is required to analyze this sequence, because the flag set "FLAG-SEQ" already sets LOSP-A and LOSP-B to TRUE.

- Analysis of sequence 3 is more complicated because the logic dependency on ac power for the AFW and HPI pumps varies.

  The AFW pumps depend on the EDGs to supply ac power. However, the HPI pumps do not depend on the EDGs because offsite ac power was recovered.

  - The difficulty related to sequence 3 is due to the changing house event settings for LOSP-A and LOSP-B.

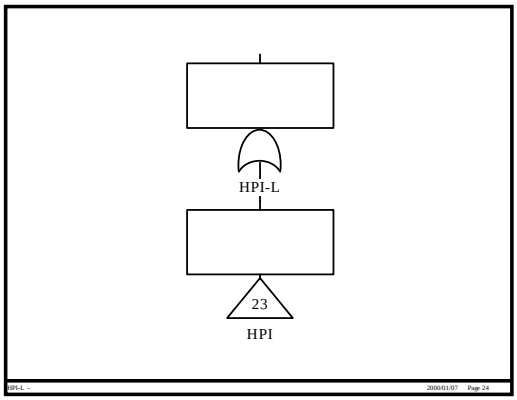
    Sequence 3 requires the house events LOSP-A and LOSP-B be set to TRUE in order for the EDGs to supply ac power to the AFW pumps. But, this sequence also requires the house events LOSP-A and LOSP-B be set to FALSE since the HPI pumps are no longer dependent upon the EDGs. So, how do we model something as both TRUE and FALSE in the same sequence?

  - In risk and reliability assessment, there are two common methods that can be used to ensure that the sequence is solved correctly.

    1. We could create two HPI fault trees that are almost identical. One fault tree (HPI-L) would transfer to the EDG fault trees for the required ac power. The other fault tree (HPI) would not transfer to the EDG fault trees, which assumes that EDGs are not needed in this mode. Note though that the analyst would have to ensure that both HPI and HPI-L fault trees are created, modified, and maintained.

2. We could use both sequence and fault tree flag sets when solving accident sequences. To use this method, a flag set would have to be created that sets LOSP-A and LOSP-B to FALSE (for HPI in accident sequence 3). This new flag set would be assigned just to the HPI fault tree.

To implement the second approach, we can use the AFW and HPI trees, but we need to create the HPI-L fault tree.



No flag set is assigned to HPI-L. HPI is assigned a flag set where LOSP-A and LOSP-B are FALSE.

◊ Since HPI is a subtree in the HPI-L fault tree, the HPI flag set will not be used on the HPI-L fault tree (flag sets are assigned only to the top gate). By using this approach, only a single fault tree model is required to be maintained.

◊ Note that SAPHIRE applies sequence flag sets first then fault tree flag sets second. Thus, fault tree flag sets will take precedence (since they will override the sequence flags).

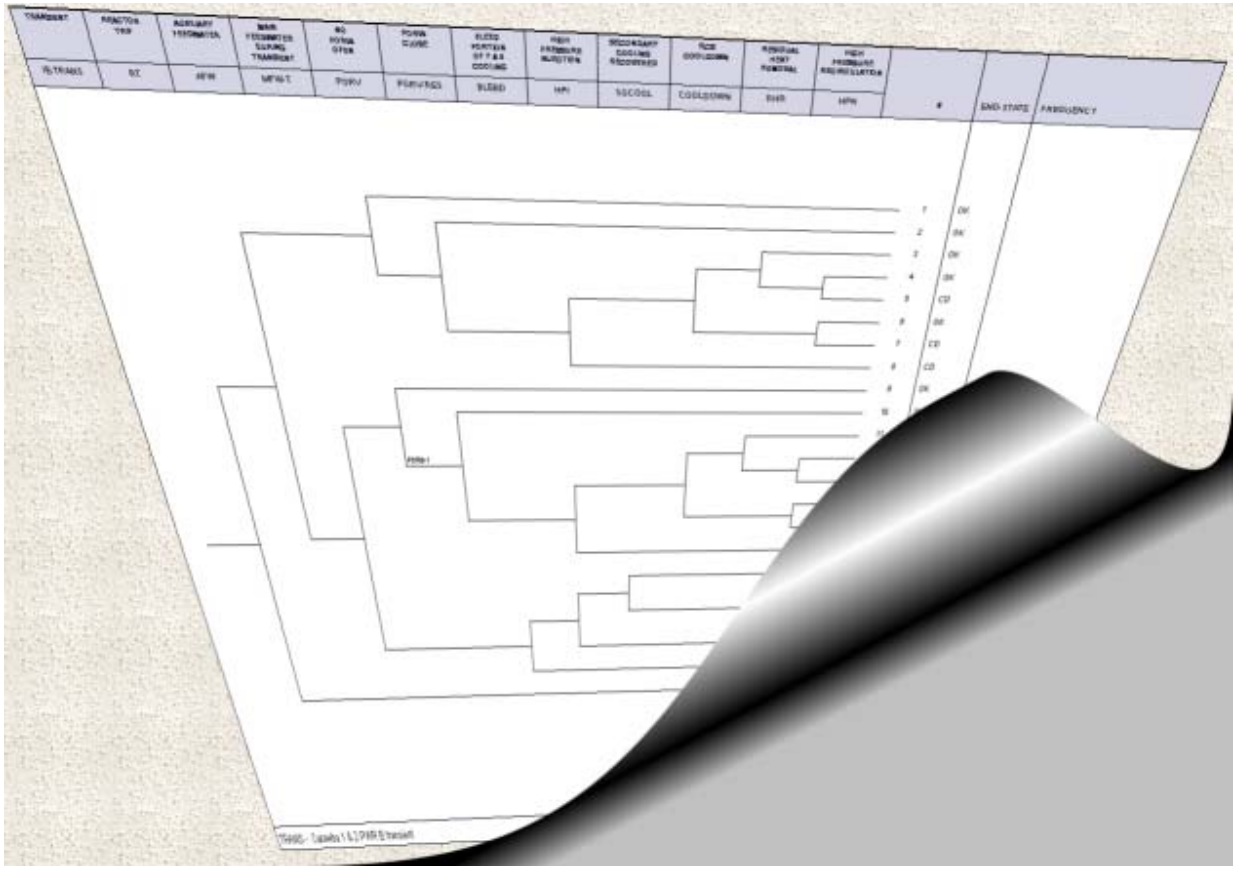
◊ When analyzing sequence 3, the house events LOSP-A and LOSP-B will be set to TRUE for the AFW fault tree but will be set to FALSE for HPI. By having these two different house event settings, the correct cut sets will be generated.

◊ When solving sequence 5, the house events LOSP-A and LOSP-B will use the sequence flag set (i.e., they will be set to TRUE).

NOTES

10 THE LARGE EVENT TREE METHODOLOGY

Section 10 describes the "large event tree" methodology and how SAPHIRE can be used to evaluate sequences using this approach. The options that allow truncation of sequences during the process of linking event tree sequences and other options related to analyzing large event trees are presented.



10.1. Large Event Tree Methodology Introduction

There are two basic approaches for accident sequence quantification:

- ◊ Fault-tree linking (covered in the SAPHIRE Basics, demonstrated in the DEMO project, and described in previous sections).

- ◊ Large event tree methodology (also called "event trees with boundary conditions").

Characteristics of the large event tree methodology include:

- ◊ Important support systems are modeled as top events in the event trees rather than being contained in the "frontline system" or "plant response system" fault trees.

  This type of modeling accounts for shared dependencies in the plant response system fault trees, and the plant response system fault trees are quantified based on the status of the support systems. This quantified probability is known as the top event split fraction.

- ◊ The paths through the event tree (i.e., sequences) can be quantified by *multiplying* the split-fractions along the path because the top events are independent (i.e., their dependencies are accounted for in the split fraction values).

  This multiplication is in contrast to the fault-tree linking approach, where simply multiplying the branch probabilities together may yield incorrect results because of the potential for double-counting component failures (i.e., a component that appears in more than one of the systems in a particular sequence).

- ◊ The split-fraction for each branch point in the model is derived from a fault tree that applies to the branch point.

  The successes and failures on the path leading to that branch point (which define the "boundary conditions" for the system fault tree) must be recognized when the fault tree is developed and solved. The resulting "split-fraction" is conditional upon the path through the event tree.

This is in contrast to the fault-tree linking approach which usually has only one fault tree that corresponds to a particular top event.

◊ The split-fractions underneath the top event are assigned by using the "Link Event Tree" rules to specify the particular fault tree that corresponds to the branch point.

◊ Each path through the event tree (i.e., sequence) is characterized by the initiating event and by the combination of failed and successful systems in the path. Success branch probabilities are retained along with the failed branch probabilities for the sequence.

In SAPHIRE, the sequence is stored as a single "cut set" even though the term "cut set" implies retaining only the failed branch probabilities.

Important features of the large event tree approach (with regard to model construction and use) are:

◊ The sharing of support system event trees with different plant response event trees, depending on the initiating event. (Described in the next section)

◊ The use of the "Link Event Tree" rules to assign split-fractions. (Described in Sections 4 and 10.4.)

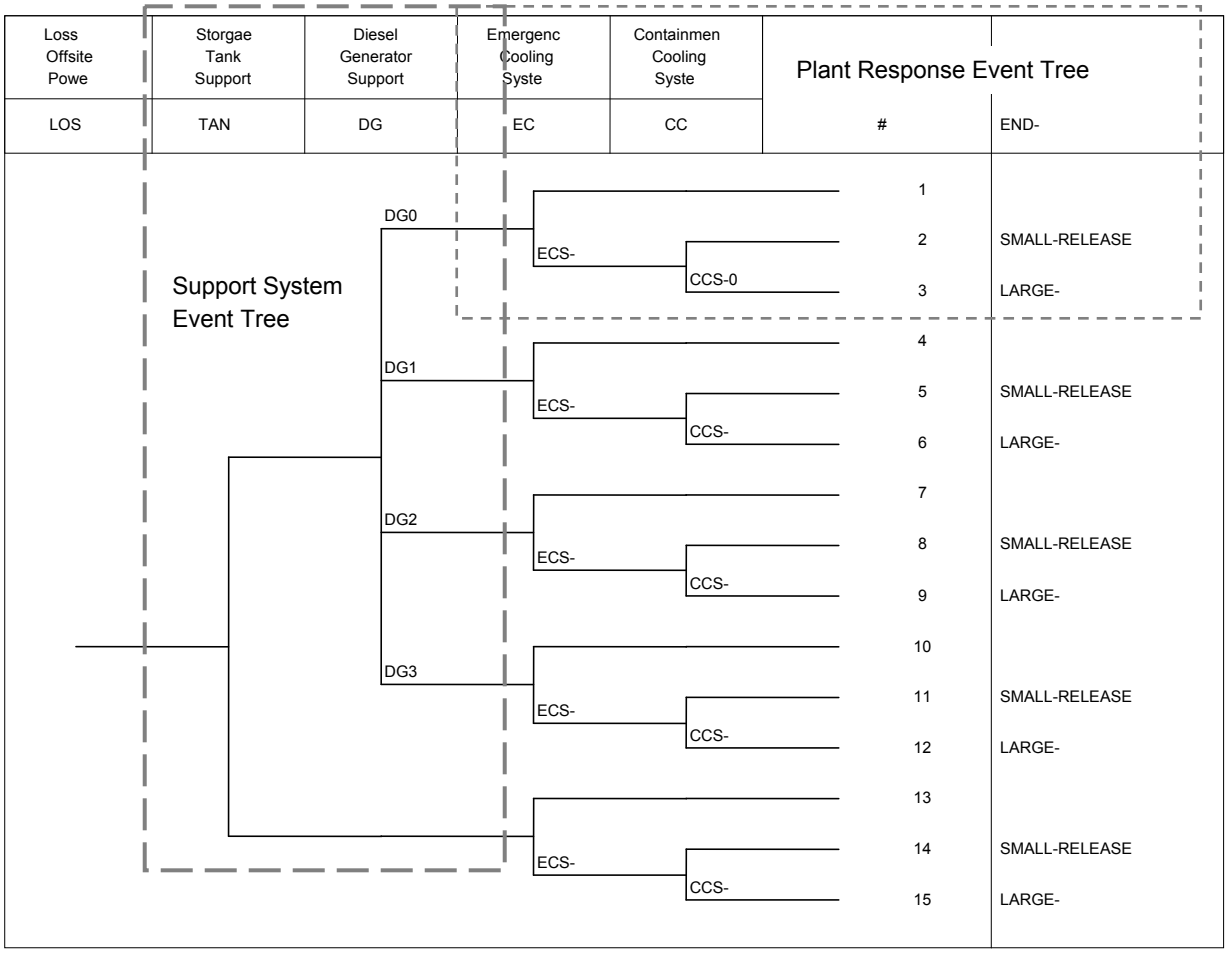
◊ The use of multiple-split branching in the event tree, such as 3-split or 4-split branching. (Described in Sections 4 and 10.4.)

◊ The use of truncation when linking event trees because of the large number of sequences that could be generated. (Described in Section 10.5.)

10.2. Large Event Trees (i.e., Initiating Event Trees, Support System Event Trees, and Plant Response Event Trees)

- Event trees for the large event tree methodology contain all of the independent components as individual top events. Because of this modeling practice, these event trees can become very large and very complicated. Therefore, these event trees are separated into distinct separate event trees that represent the different systems required to mitigate events.

- The LOSP event tree in the DEMO project is modified to show how a large event tree would be created based on the simple systems. The event tree is shown below.



- The LOSP event tree is now separated into the distinct event trees that are usually developed and discussed in PRAs that utilize the large event tree methodology.

  - The “initiating event tree” represents the different events that can cause a reactor trip and requires plant responses. The “initiating event tree” for this example contains the initiating event frequency, which passes straight through the event tree and transfers to the “support system event tree”. An illustration of the “initiating event tree” is shown on the next page.

  - The "support system event tree" represents the support systems that are required for the frontline systems to operate (i.e., power systems, instrument air, etc). These event trees may be used by several initiating events. The "support system event tree" needs to transfer to the appropriate “plant response event tree”, depending on the initiating event.

  - The “plant response event tree” represents the frontline systems that are required to mitigate the event (i.e., emergency cooling, containment cooling).

- To connect the event trees (and avoid having to duplicate event trees), the following approach is preferred:

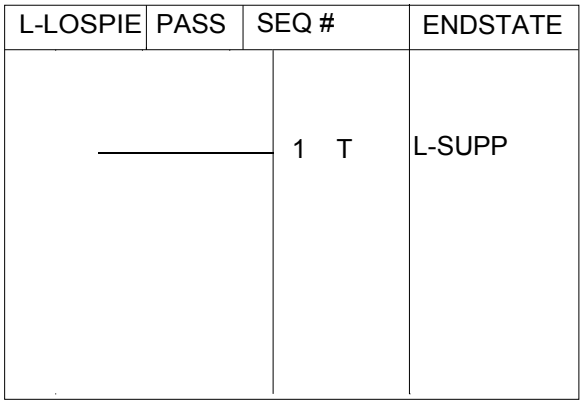
  - Create an event tree that contains the initiating event and transfers to the appropriate support system event tree. (Note: SAPHIRE requires that at least 2 tops are present in each event tree; however, there does not need to be any branching.)

  - The path through the support system event trees (which may contain many transfers to include all of the support systems) will ultimately result in the need to transfer to the appropriate plant response event tree.

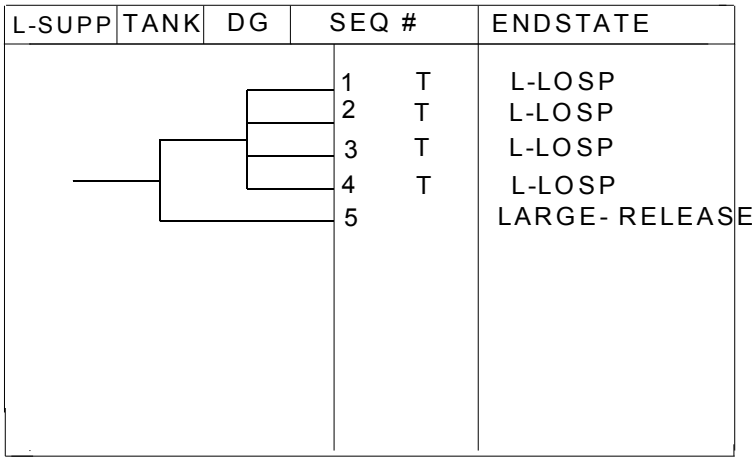
    - Rules can be written in the "Link Event Tree" rule editor to enact this transfer. (see Section 4.)

These three event trees will be used as examples in this section.

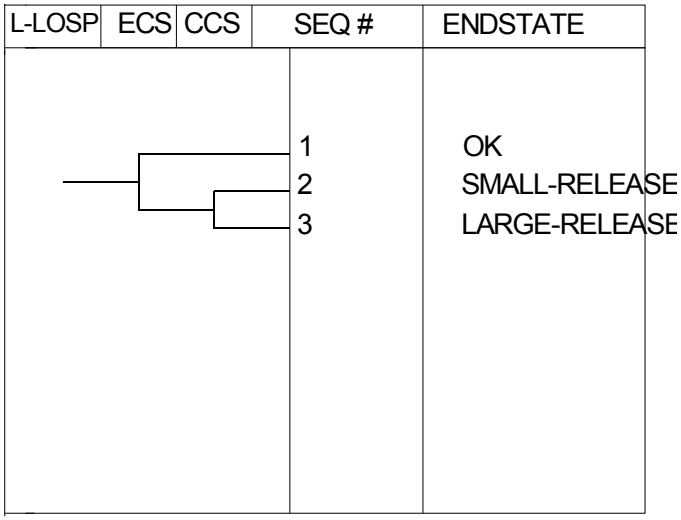
The "Initiating Event" Tree



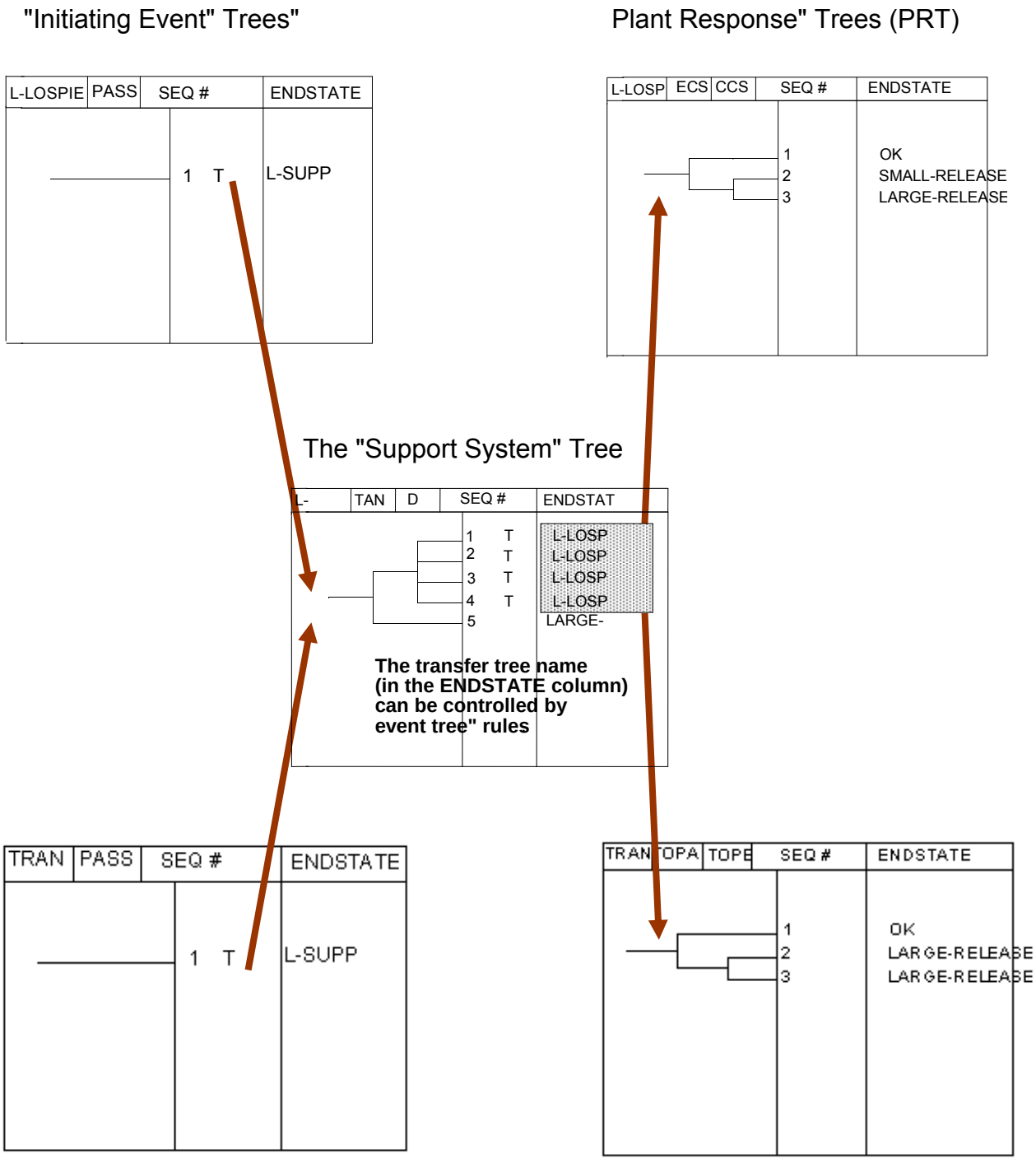
The "Support System" Tree



The "Plant Response" Tree



Notice that more than one initiating event can call the same support system tree. And the support system tree could transfer to different plant response trees.



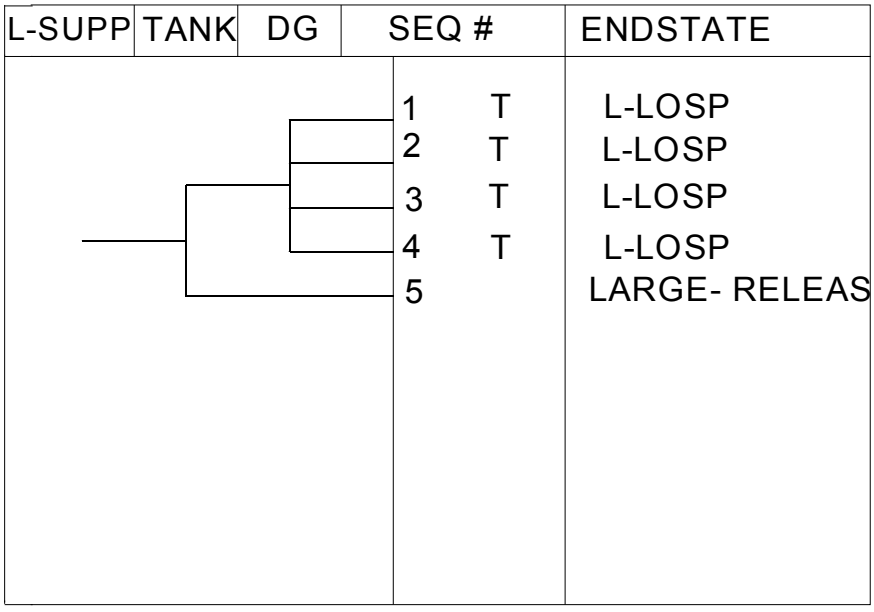
The Support System Tree

- The support system event tree contains all of the support systems that are required to operate in order for the front line systems to be available. If the support systems are unavailable or partially available this will impact the operability of the front line systems.

- In this example, top event TANK is a support component that is required for both front line systems ECS and CCS. Therefore, top event TANK questions the status of the tank, and if the tank is unavailable, then both ECS and CCS can not operate.

- The probability of failure for top event TANK is specified directly. Modification of failure probabilities for top events will be discussed in the next section. SAPHIRE determines the success probability as the complement of the failure probability.

/TANK = 1 - TANK



- Top event DG questions the status of the two diesel generators that provide support to ECS and CCS.

  - The top branch for DG represents both DGs are available to supply electrical power to the front line system components (top event assignment is DG0).

- ◊ The second branch under DG represents the success of DG-A and failure of DG-B (top event assignment is DG1). (By knowing what support system components are available, the front line components are adjusted for further evaluation through the plant response event tree.)

- ◊ The third branch under DG represents the success of DG-B and failure of DG-A (top event assignment is DG2).

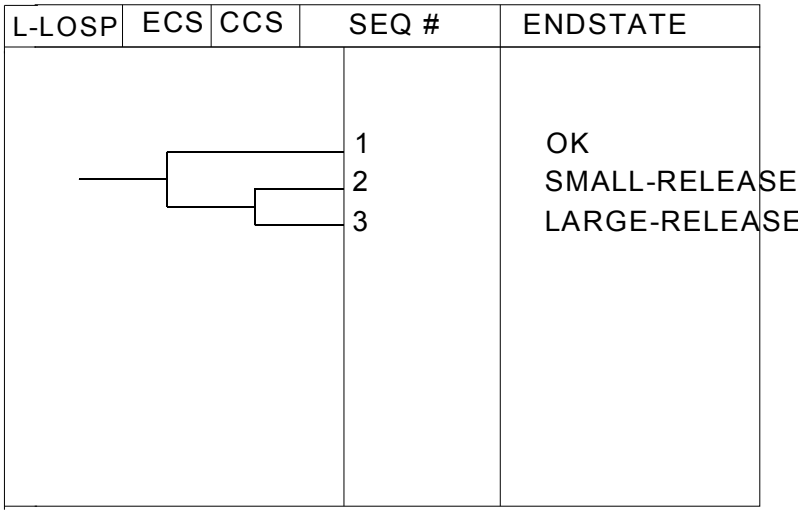
- ◊ The forth and final branch represents the failure of both DG-A and DG-B (top event assignment is DG3).

- ♦ (Again, the next section will go into detail on how to specify the failure probability (i.e., split-fraction probability for the top event DG).

- ♦ The support system event tree now transfers to the plant response event tree. The different sequences will transfer to the same plant response event tree; however, different front line top events will be questioned due to the availability or unavailability of the support system components.

The Plant Response Tree

- ♦ The "plant response tree" represents how the plant will respond to a given initiating event based on the availability or unavailability of the support system components. The top events/split fractions on the "plant response tree" are conditioned on the availability of support systems.



- ♦ For this example, the front line system top events (ECS and CCS) are modified based on the following changes:

(1) The TANK event is removed from the ECS and CCS fault trees because it supports both systems.

(2) The ECS and CCS top events are modified based on the status of the DGs (i.e., conditional probabilities for ECS and CCS are calculated based on the status of the DGs).

- The split-fraction probability (i.e., conditional failure probability) of ECS (and CCS) is dependent upon the path through the support system event tree. In other words, the split-fraction probability for ECS is different if ECS is questioned via sequence 2 versus sequence 1 of the support system tree.

- Once we know the sequence path through the support system tree, which specifies what support system is available (or unavailable), a special version of the ECS (or CCS) fault tree is created, which is used to calculate the split-fraction probability conditional on the path (shown in the next section).

10.3. Top Event Split-Fraction Probability Assignment

- There are two ways to assign the appropriate split-fraction probability to the top events in both the support system event tree and plant response event tree:

1) Assign the split-fraction probability directly to the top events once they are added to the SAPHIRE database.

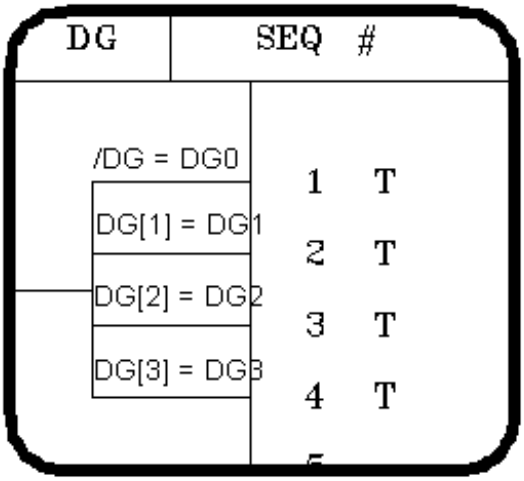
2) Create new fault trees for each top event in the event trees (TANK, DG, ECS and CCS). (The new fault trees for ECS and CCS are conditional on the status of the DGs. These new fault trees are solved and then assigned the "S" calculation type.

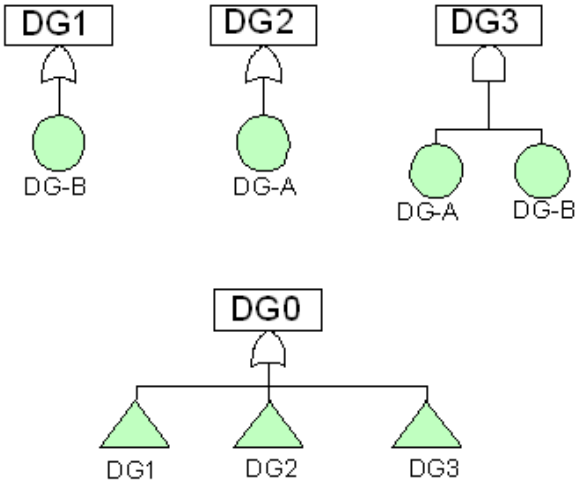
10.3.1 Assign the split-fraction probabilities directly

- To assign the split-fraction probability directly, the probability is entered via the **Modify → Basic Events** menu (because SAPHIRE recognizes top events as developed events).

- Prior to assigning the split-fraction probability, this probability needs to be calculated. This can be done by hand calculations or developing and solving a representative fault tree, depending on the complexity of the top event.

- The split-fraction probability for each DG branching is calculated using the following fault trees.
  - A “fault tree” for each diesel generator needs to be developed. The fault trees are developed based on the event tree logic.
  - Let us look at the DG top event in detail. The top branch (/DG) represents success of *both* diesel generators and is represented by fault tree DG0. The next branch down represents success of diesel generator A and failure of diesel generator B. Thus, DG1 is a fault tree containing just DG-B. The third branch represents success of diesel generator B and failure of diesel generator A. Thus, DG2 is a fault tree containing just DG-A. The bottom branch represents both diesel generators being failed. Thus, DG3 is a fault tree containing DG-A “AND” DG-B.





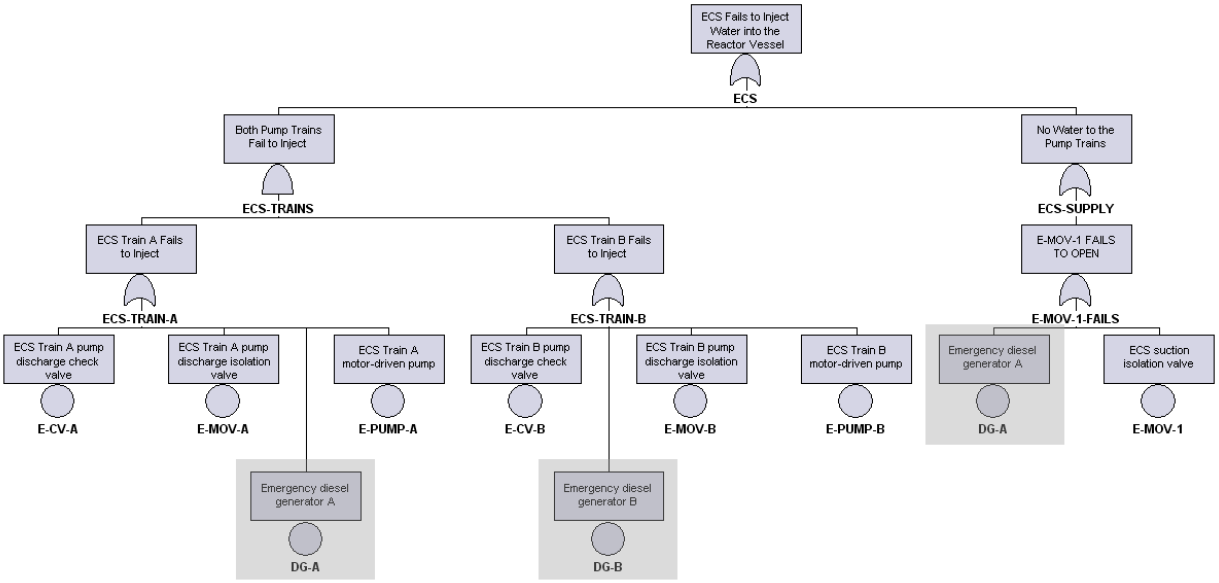
- The split-fraction probabilities for each branch are determined from the corresponding fault trees. The split-fraction for /DG is taken as the complement of the DG0 fault tree.

- The developed fault trees representing the different states of the two diesel generators can now be solved via **Fault Tree → Solve** option with no truncation.

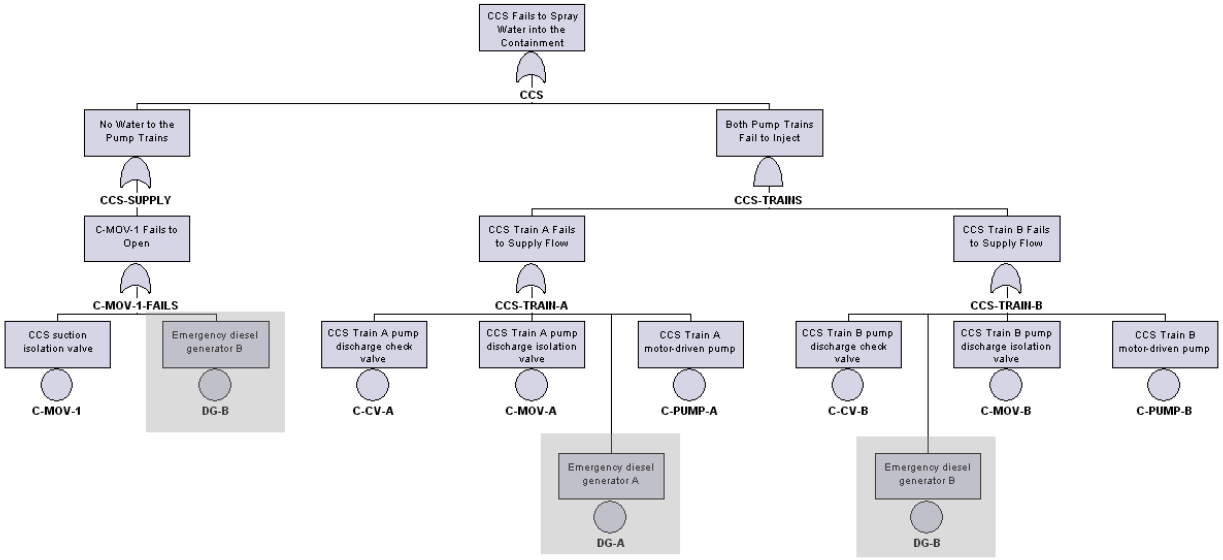
- The probability calculated from each fault tree can now be assigned to the fault tree top events in the **Modify → Basic Events** menu.

- The same process needs to be performed for the ECS system and the CCS system. The ECS and CCS fault trees need to be solved via the use of change sets to calculate their conditional probabilities. These conditional probabilities are then used as the split-fraction probability for these top events in the "plant response tree".

- The ECS and CCS fault trees are shown below with the modifications required in order to calculate their conditional probabilities and the name of the new top events that are required to handle the different conditions due to electrical support. (i.e., DGs).



DG Status	***ECS System Name***	***Split-Fraction***
No DGs failed	ECS-0	1.066E-3
Only DG-A failed	ECS-A (FAILED)	1.0
Only DG-B failed	ECS-B	9.076E-3
Both DGs failed	ECS-AB (FAILED)	1.0



DG Status	***CCS System Name***	***Split-Fraction***
No DGs failed	CCS-0	1.066E-3
Only DG-A failed	CCS-A	9.076E-3
Only DG-B failed	CCS-B (FAILED)	1.0
Both DGs failed	CCS-AB (FAILED)	1.0

10.3.2 Assign probabilities using “S” calculation

- To assign the top event probabilities using the “S” calculation, the following steps are required.

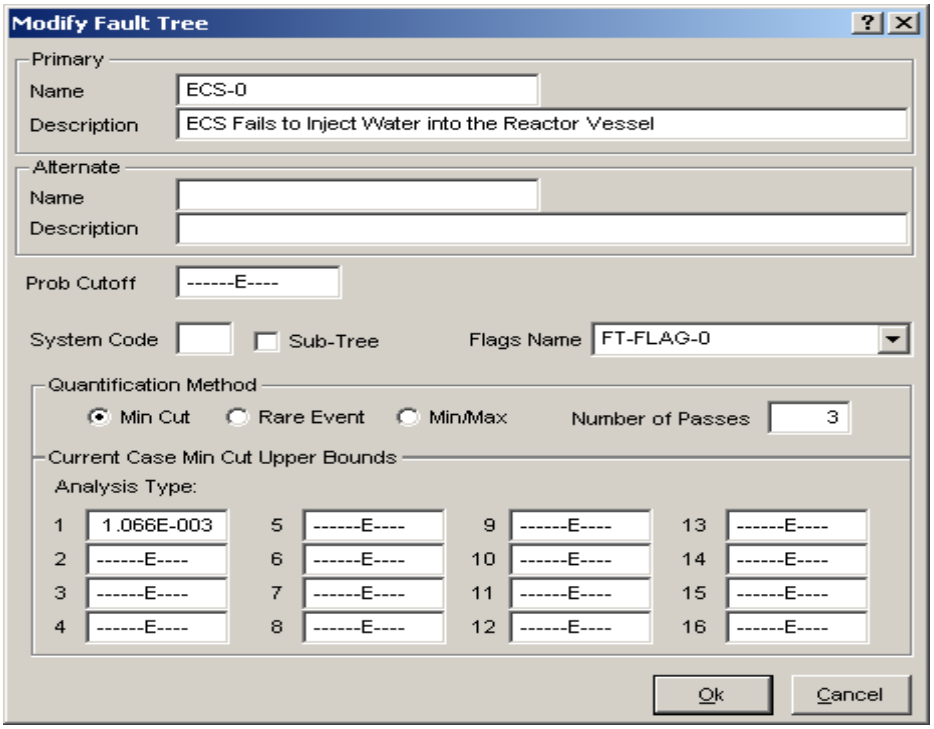
1. Multiple copies of the ECS and CCS fault trees need to be created with the names listed above (i.e., ECS-0, ECS-A, ECS-B, and ECS-AB).

2. Create fault tree flag sets that can be assigned to the different ECS and CCS fault trees in order to handle the conditional probability calculation.

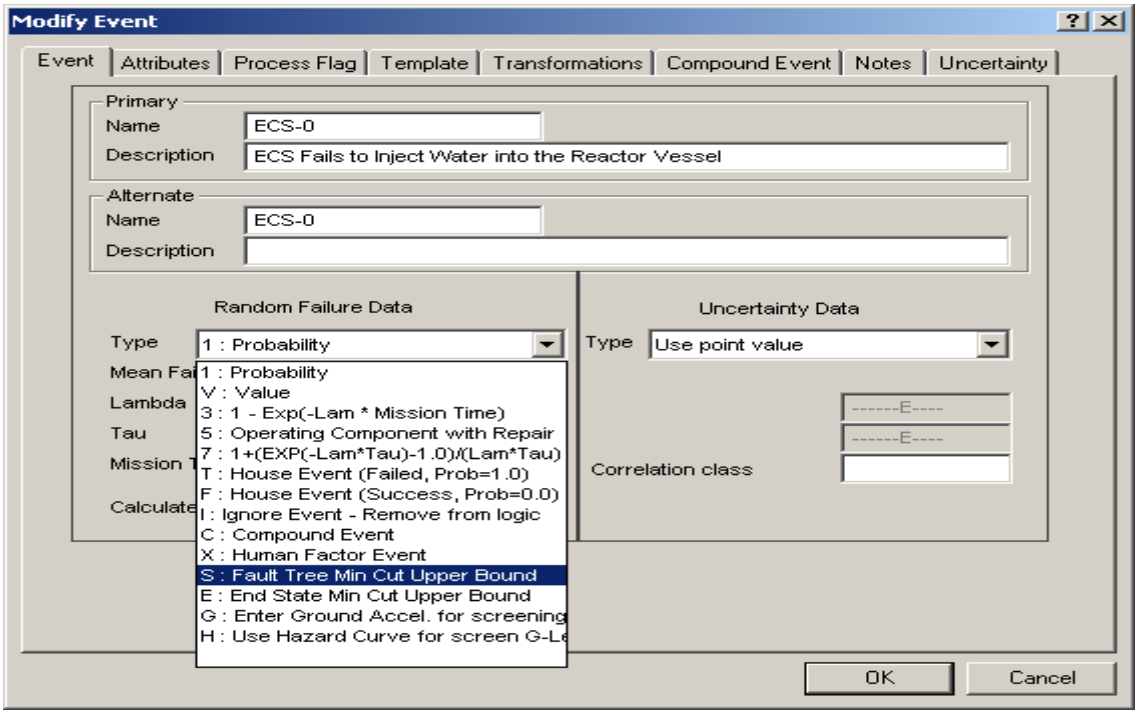
Flag Set Name	**Basic Events**	**House Event Identifier**
FT-FLAG-0	DG-A	FALSE
	DG-B	FALSE
	TANK	FALSE
FT-FLAG-A	DG-A	TRUE
	DG-B	FALSE
	TANK	FALSE
FT-FLAG-B	DG-A	FALSE
	DG-B	TRUE
	TANK	FALSE
FT-FLAG-AB	DG-A	TRUE
	DG-B	TRUE
	TANK	FALSE

3. Assign the fault tree flag sets to the appropriate fault tree for the calculation process. This is performed using the **Modify → Fault Trees** option. Then highlight each fault tree individually, right click, select **Modify** and in the **Flags Name** field assign the appropriate flag.

Flag Set Name	Fault Tree
FT-FLAG-0	ECS-0 CCS-0
FT-FLAG-A	ECS-A CCS-A
FT-FLAG-B	ECS-B CCS-B
FT-FLAG-AB	ECS-AB CCS-AB



4. Modify the calculation type for the fault trees via **Modify → Basic Events** option.
   - This step is performed to all of the fault trees (DG0, DG1, DG2, DG3, ECS-0, ECS-A, ECS-B, ECS-AB, CCS-0, CCS-A, CCS-B, and CCS-AB).



5. Generate fault tree cut sets for all of the fault trees (**Fault Tree → Solve**) with no truncation.

6. Generate event data via **Generate → Generate**.

♦ Now that all of the split-fraction probabilities have been calculated, these top events need to be assigned to the event tree for sequence cut set generation.

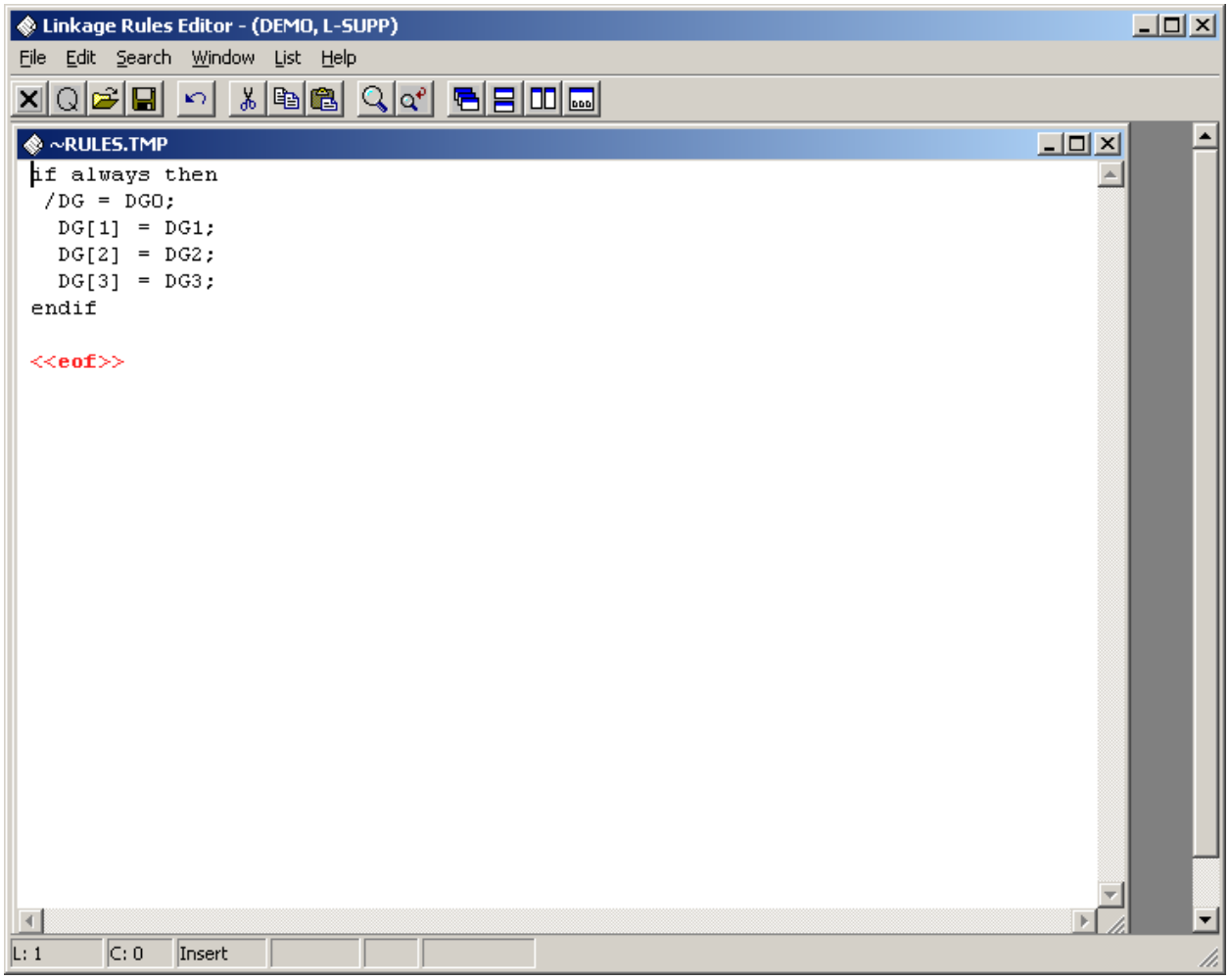
10.4. Using "Link Event Tree" Rules to Assign Split-Fractions

As discussed, the particular path through the event trees (support system and plant response trees) determines the status of support systems and frontline systems. These paths determine the appropriate “top event” substitutions for the support systems and plant response systems.

In contrast, with the fault-tree linking approach, each top event usually corresponds to a single fault tree or a single top event probability for the failed branch.

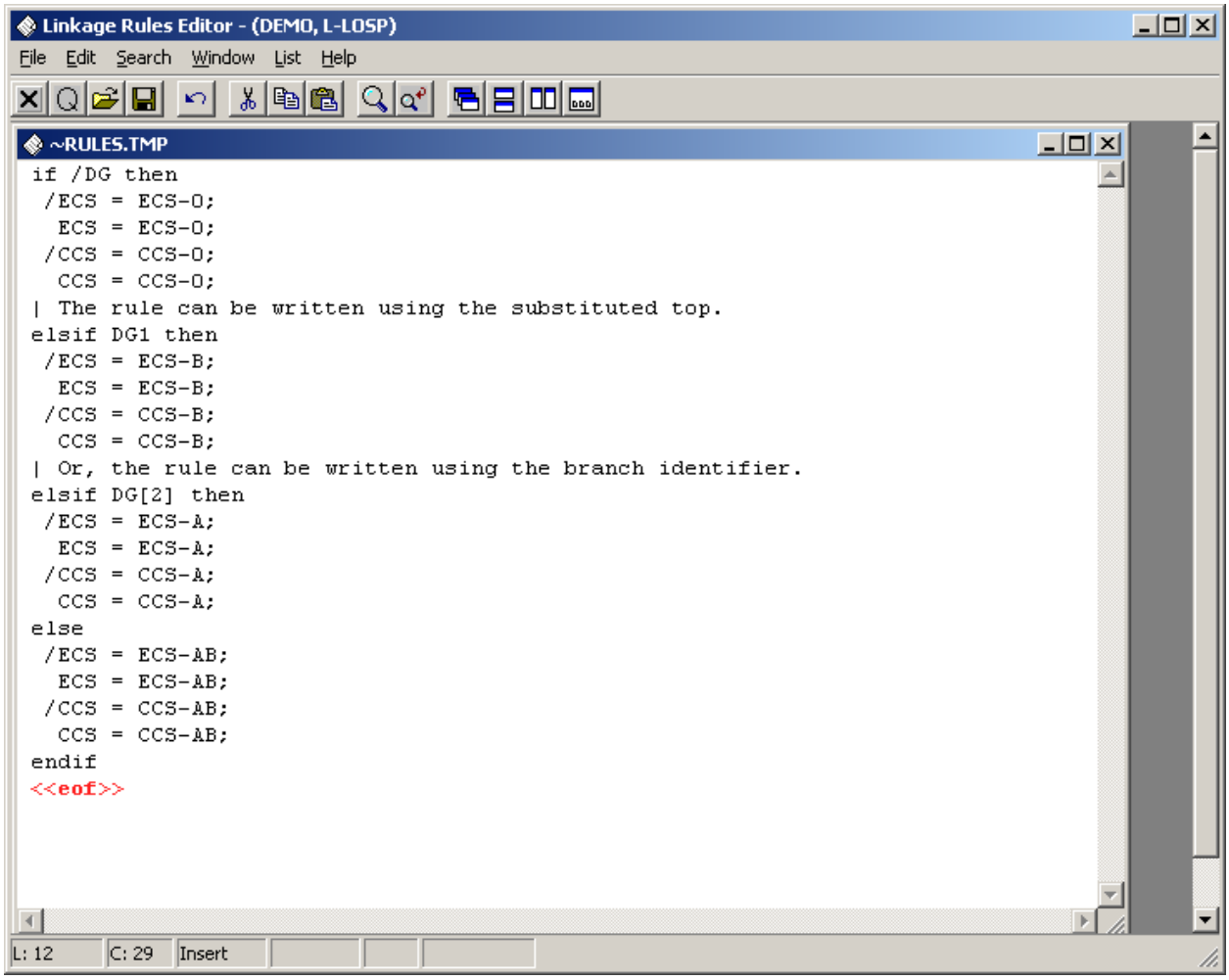
Assigning the Support System Split-Fractions

- The "Link Event Tree" rule editor is used to assign the appropriate "top event" to each branch (i.e., split-fraction probability). This top event substitution is dependent upon the path through the event tree. The rules assigned to the support system event tree for proper substitution are as shown.



Assigning the Plant Response Split-Fractions

- The "Link Event Tree" rule editor is used to assign the appropriate "top event" to each branch (i.e., split-fraction probability). This top event substitution is dependent upon the path through the event tree. The rules assigned to the plant response event tree for proper substitution are as shown.



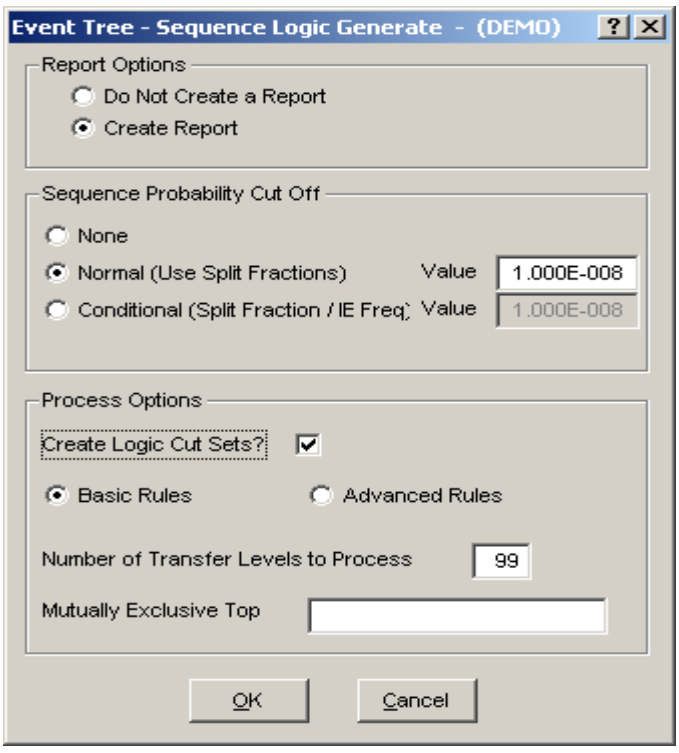
(Notice that the rules can be written in terms of the branch identifier (e.g. DG[1]) or in terms of the fault tree name (e.g. DG1). Also, the system name in the search criteria (e.g., DG) must have been assigned by earlier rules (usually the support system rules).

10.5. Truncating Sequences During Event Tree Linking

- The prerequisites for generating sequences with truncation during the link step (note this is different than the truncation during the “solve” option for sequence cut sets) are:
  - The "fault trees" that do not have fault tree logic should have a failure probability specified prior to linking the event tree(s).
  - The fault trees that have fault tree logic should also have a failure probability specified prior to generating (and truncating) sequences. This can be accomplish by either directly specifying a probability or by using the "S" Calculation Type (discussed in Section 10.3).

Generating the Sequence “Cut Sets” During Event Tree Linking

- The L-LOSPIE event tree (from Section 10.2) will transfer to the L-SUPP tree, which subsequently transfers to the L-LOSP tree.
  - To generate the sequence cut sets for the L-LOSPIE event, highlight only the L-LOSPIE tree.
  - Click the right mouse button and select **Link Trees**.



- When using the *large event tree approach*, we generally need to use sequence truncation (i.e., discard sequences with low frequencies) due to the potentially large number of sequences.

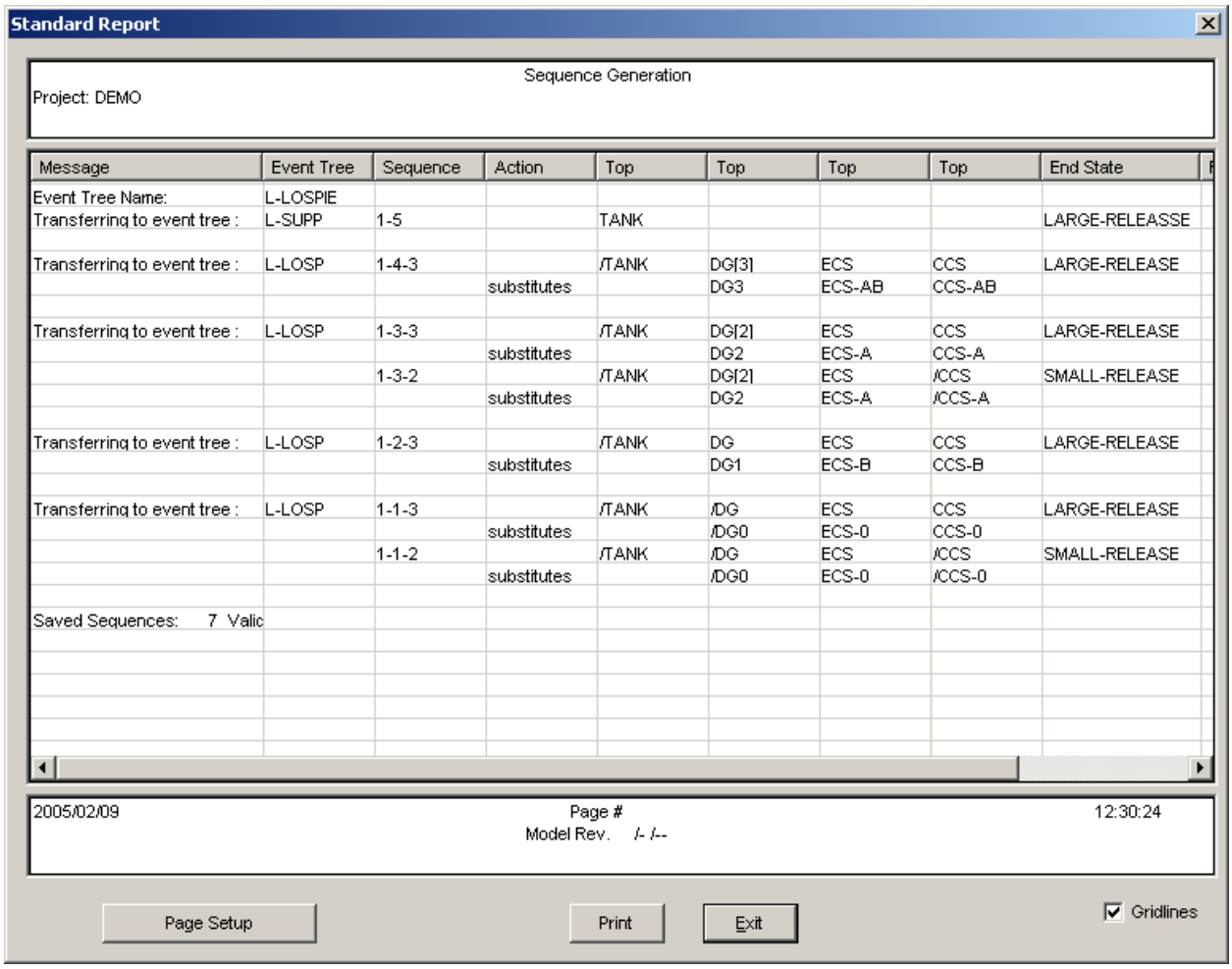
  - To perform truncation when generating sequence cut sets via the **Link Trees** option, we need to specify two options.

    1. The radio button for either Normal or Conditional truncation option needs to be selected and then enter the cut off value.
       - The **Normal** option will truncate the sequence once its value is below the "cut off" value specified divided by the initiating event frequency. In effect, this approach "equalizes" the sequences across different initiators.
       - The **Conditional** option will truncate the sequence once its value is below the "cut off" value specified.

    2. Click the **Create Logic Cut Sets** check box. This option tells SAPHIRE that the logic being created (for each sequence) via the link process should simply be treated as a cut set. Consequently, a single cut set will appear for the sequence (after linking) that is the product of the initiating event, all failure tops in the sequence, and all success tops in the sequence.

NOTE: During this sequence truncation process, fault tree logic is not evaluated. Instead, the fault tree (i.e., top event) split fractions are used to obtain the sequence frequency.

The sequences generated from the L-LOSPIE tree are determined as:



Standard Report

Sequence Generation

Project: DEMO

Message	Event Tree	Sequence	Action	Top	Top	Top	Top	End State
Event Tree Name:	L-LOSPIE							
Transferring to event tree :	L-SUPP	1-5		TANK				LARGE-RELEASSE
Transferring to event tree :	L-LOSP	1-4-3		/TANK	DG[3]	ECS	CCS	LARGE-RELEASE
			substitutes		DG3	ECS-AB	CCS-AB	
Transferring to event tree :	L-LOSP	1-3-3		/TANK	DG[2]	ECS	CCS	LARGE-RELEASE
			substitutes		DG2	ECS-A	CCS-A	
		1-3-2		/TANK	DG[2]	ECS	/CCS	SMALL-RELEASE
			substitutes		DG2	ECS-A	/CCS-A	
Transferring to event tree :	L-LOSP	1-2-3		/TANK	DG	ECS	CCS	LARGE-RELEASE
			substitutes		DG1	ECS-B	CCS-B	
Transferring to event tree :	L-LOSP	1-1-3		/TANK	/DG	ECS	CCS	LARGE-RELEASE
			substitutes		/DG0	ECS-0	CCS-0	
		1-1-2		/TANK	/DG	ECS	/CCS	SMALL-RELEASE
			substitutes		/DG0	ECS-0	/CCS-0	
Saved Sequences: 7 Valic								

2005/02/09 Page # 12:30:24

Model Rev. /- /--

Page Setup Print Exit Gridlines

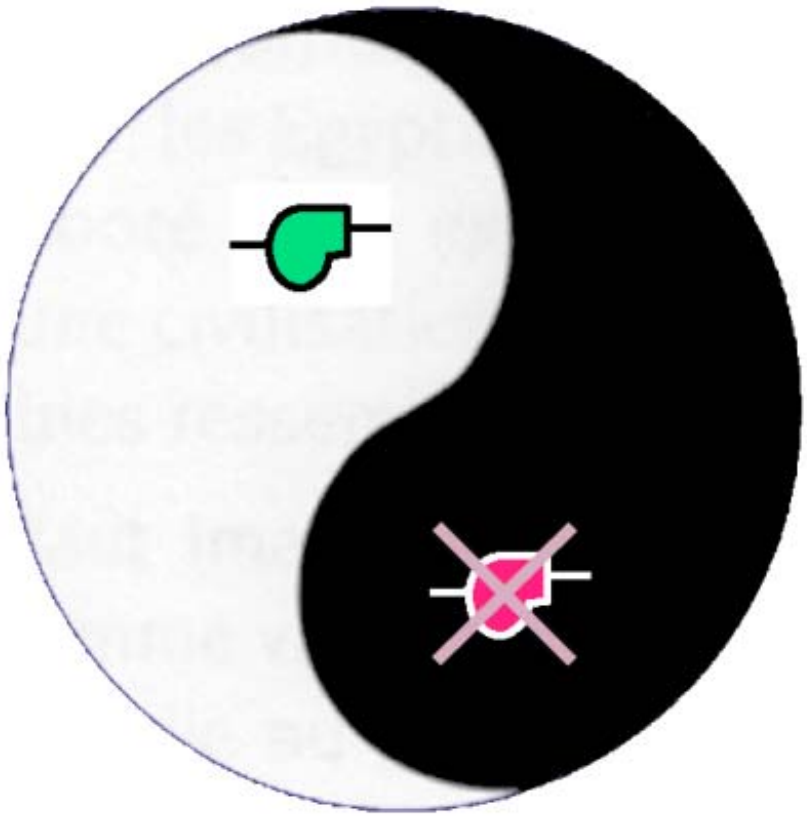
After completing the Link process, these sequences will now appear in the project list of sequences. For example, going to the **Sequence** option and viewing the cut sets for sequence 1-5 (from event tree L-LOSPIE) would display a single cut set:

L-LOSPIE * TANK

If many sequences are generated, the sequence generation process may take a long time. For some large event tree risk assessments, the potential number of sequences exceeds one billion.

11 MUTUALLY EXCLUSIVE EVENTS

Section 11 presents the topic of mutually exclusive events. A review of mutually exclusive events is provided along with methods to remove these events from SAPHIRE PRA results.



11.1. Mutually Exclusive Events Introduction

The term "mutually exclusive events" refers to two or more basic events that appear in a single cut set which should not appear together.

- Technical specifications or other facility restrictions may prevent two components from being tested or in maintenance at the same time.

- Other general logic modeling concerns may lead the analyst to remove specific combinations of events.

- A component can not be both failed and working (success) in the same cut set.

Most mutually exclusive groups include only two or three components.

An analyst may recognize "up-front" that mutually exclusive event combinations will appear just by knowing how the fault or event tree logic modeling was performed.

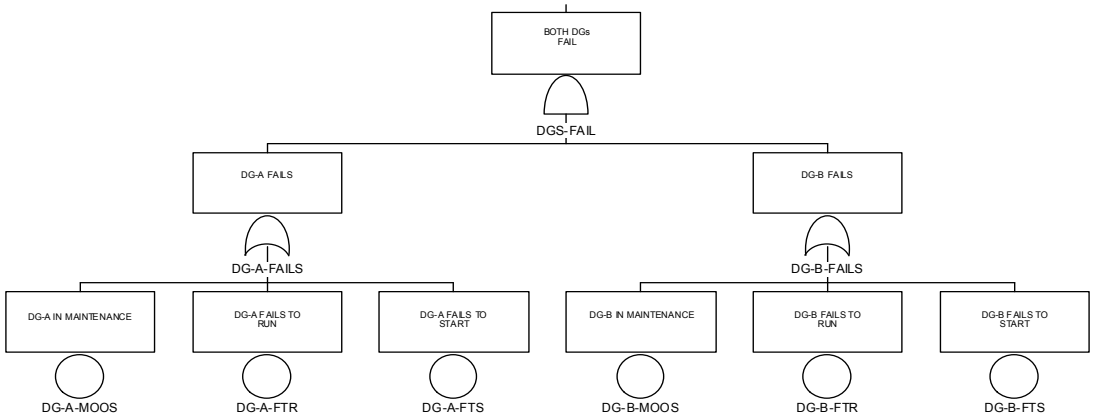
- Other unrecognized mutually exclusive events may not be evident until the analyst solves and evaluates the fault tree or sequence cut sets.

Mutually Exclusive Event Example

The fault tree logic (on the following page) will produce a cut set containing the two maintenance events

DG-A-MOOS * DG-B-MOOS

Assuming that the facility procedures restrict both diesel generators from being in maintenance simultaneously while at power, this cut set is an example of mutually exclusive events.



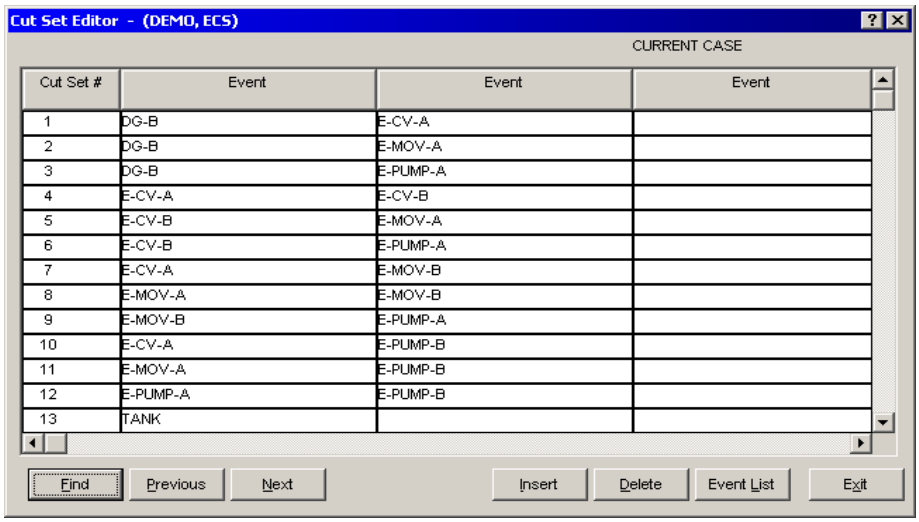
Several methods exist to remove cut sets containing mutually exclusive events. These methods, in order of increasing preference, include:

- Editing the cut sets manually using the cut set editor to "weed-out" the mutually exclusive events.
- Using the "mutually exclusive top event" feature when linking event trees.
- Modify logic models (via NOT gates or complemented events) to remove prohibited combinations of events.
- Using the Recovery Rules to define combinations of events in cut sets that would be deleted (via the DeleteRoot keyword).

Let us discuss these methods in turn.

Manual cut set editor method

Once you have generated sequence (or fault tree) cut sets, these cut sets can be manually modified using the **Cut Sets → Edit** option. But, this process is not recommended since it is both error prone and time consuming.



Cut Set #	Event	Event	Event
1	DG-B	E-CV-A	
2	DG-B	E-MOV-A	
3	DG-B	E-PUMP-A	
4	E-CV-A	E-CV-B	
5	E-CV-B	E-MOV-A	
6	E-CV-B	E-PUMP-A	
7	E-CV-A	E-MOV-B	
8	E-MOV-A	E-MOV-B	
9	E-MOV-B	E-PUMP-A	
10	E-CV-A	E-PUMP-B	
11	E-MOV-A	E-PUMP-B	
12	E-PUMP-A	E-PUMP-B	
13	TANK		

Mutually exclusive top method

The "mutually exclusive top" method is only applicable when solving sequence cut sets. To use this method, the analyst must first define fault tree logic that represents the combination of events that are mutually exclusive. Thus, the tree

ME-TOP AND DG-A-MOOS DG-B-MOOS

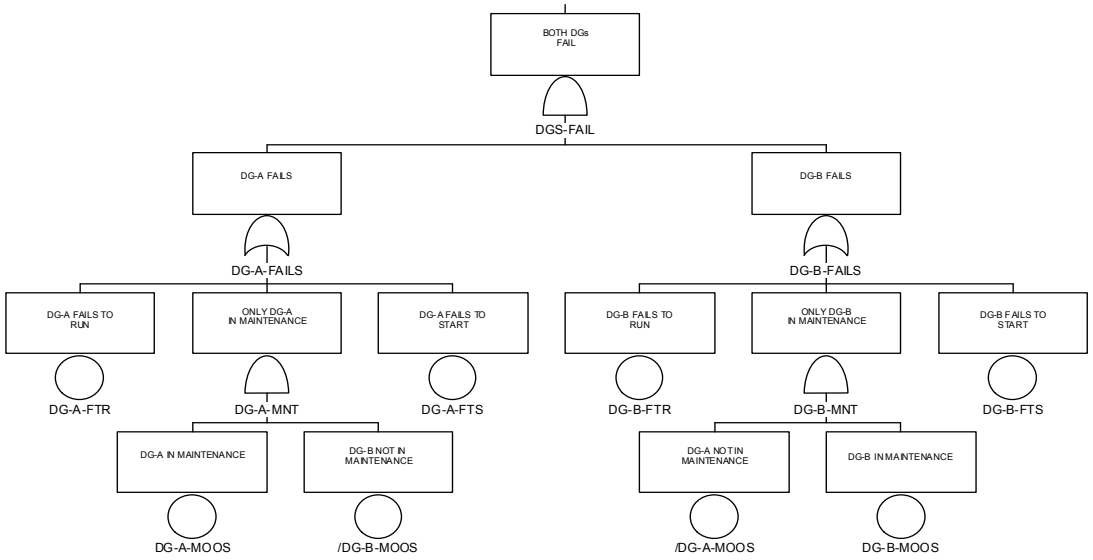
would delete cut set containing both DG-A-MOOS and DG-B-MOOS.

This method has been superceded by the recovery rule option.

Logic modification method

This method requires that the analyst modify the fault tree logic in order to remove excluded combinations of events. An example of the “modified” example fault tree is shown on the next page. Drawbacks to the “logic modification” method include:

1. The effort needed to modify the fault tree logic
2. The fact that complemented basic events (i.e., success event) will appear in the list of cut sets.



11.2. Mutually Exclusive Event Removal Via Recovery Rules

Recovery Rules, discussed in Section 5, are heuristics which allow the user to define groups of events that, if appearing together, results in the deletion of the cut set.

Mutually exclusive rules may be specified for either fault trees or sequences.

In most cases, the preferred method of removing cut sets is through the use of Recovery Rules. For example, the NRC's Standardized Plant Analysis Risk (SPAR) Revision 3 models use recovery rules to remove mutually exclusive events.

During cut set generation, the recovery rules may be automatically applied. Thus,

- ◊ No changes to logic models are needed.
- ◊ No manual manipulations to cut sets are required.
- ◊ No "mutually exclusive" fault trees are necessary.

The rules for removing mutually exclusive events may be developed for a single fault tree, all fault trees, a single sequence, a single event tree, or all sequences.

To apply or edit the *FAULT TREE* Recovery Rules, select the **Fault Tree** menu.

To edit the Recovery Rules for a particular fault tree, highlight the fault tree name, right click the mouse and select **Cut Set → Recover → Edit Rules**, then select the **Fault Tree** radio button.

To edit the Recovery Rules for all fault trees, highlight a fault tree, right click the mouse and select **Cut Set → Recover → Edit Rules**, and then select the **Project** radio button.

To apply or edit the *SEQUENCE* Recovery Rules, select the **Sequence** menu.

To edit the Recovery Rules for a particular sequence, highlight the sequence name, right click the mouse and select **Cut Set → Recover → Edit Rules**, then select the **Sequence** radio button.

To edit the Rules for a particular event tree, highlight a sequence that is part of the event tree, right click, and select **Cut Set → Recover → Edit Rules**, then select the **Event Tree** radio button.

To edit the Recovery Rules for all sequences, highlight a sequence, right click the mouse and select **Cut Set → Recover → Edit Rules**, then select the **Project** radio button.

To demonstrate the uses of the Recovery Rules, the example below shows how the rules could be used to remove the cut set containing both diesel generators failing from the DEMO project. The rule for the LOSP sequence project rules is:

```
| This rule searches for both diesel generators failing
if  DG-A * DG-B  then
  DeleteRoot;        |  Delete the cut set matching the search criteria
endif
```

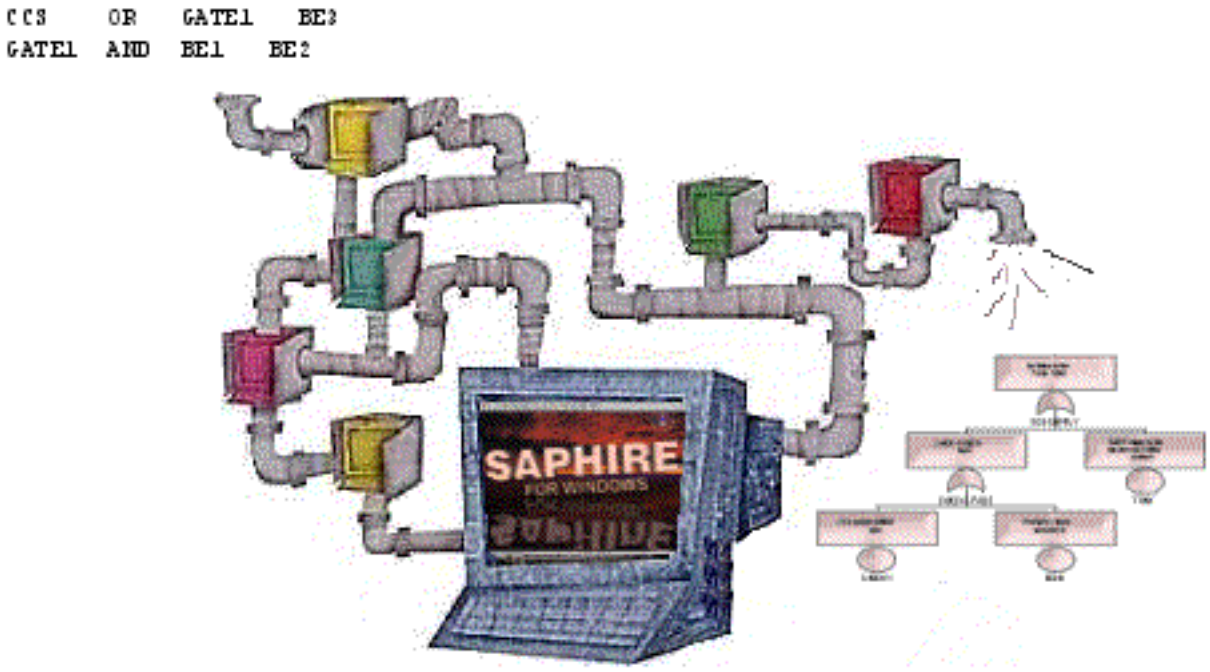
The Recovery Rule was then applied to both sequences 2 and 3 when we solved for sequence cut sets (using no truncation).

Sequence 3 changed from an original value of 1.8E-3 to a value of 8.4E-4.

◊ Only *one* cut set was removed from sequence 3, but it happened to be the dominant cut set.

12 USING MAR-D

Section 12 explains the use of the MAR-D module for transferring PRA model data. The different types of MAR-D files that contain model information and their file formats are introduced.



12.1. Uses of the MAR-D Module

MAR-D provides an interface to load or extract data files that define the PRA database. The files are in a "flat-file" or ASCII file format.

Typical uses of MAR-D include:

- Transfer PRA information between data bases — Extracting MAR-D files from one SAPHIRE project and loading them (via MAR-D) into another SAPHIRE project. (The SAPHIRE project may be a new one or a previously existing project.)

- Import other PRA code information — Formatting the model information from another PRA code to use MAR-D file formats and creating a SAPHIRE project by loading the files via MAR-D.

- Edit PRA files using a text editor — Extracting MAR-D files from a SAPHIRE project, editing the files to make changes to the model or model descriptions, and loading those files (via MAR-D) back into the SAPHIRE project.

- Archiving PRA files — Saving the MAR-D files for long term storage in a text format rather than the native binary SAPHIRE format.

12.2. MAR-D File Format

The MAR-D file text format and field descriptions are provided in SAPHIRE reference material (NUREG-6116, Volume 2 (Section 12.3 and Appendix B) and Volume 8).

Some general MAR-D formatting rules are:

Use UPPER CASE for event and model names, i.e., CCS, C-CV-A.

Upper and lower case can be used for descriptive text fields.

Entries longer than the allowed field length will be truncated.

Commas are field delimiters in most formats; therefore, commas cannot be used in descriptive text fields.

Leading or trailing "empty" spaces are allowed.

An "*****" denotes a comment field in most formats; however, " | " denotes a comment field in rule files.

A single line should not exceed 250 characters in length.

^EOS is used to signal the separation of MAR-D input contained in a single file. For example, when fault tree logic for more than one fault tree is contained in a single file, the **^EOS** signals that the data for the current fault tree is complete and that another fault tree follows.

Mar-D File Descriptions and General Guidance for Usage

Project Information

Mar-D File Description	Extension	General Guidance
Project Names/Description	.FAD	Not needed if defined in the "receiving" project.
Project Attribute File	.FAA	Descriptive, contains default mission time
Project Textual Information	.FAT	Descriptive information only.
Project Recovery Rules	.FAY	Needed if feature is used.
Project Partition Rules	.FAP	Needed if feature is used.
Project System Recovery	.FAS	Needed if feature is used.

Attribute Descriptions

Mar-D File Description	Extension	General Guidance
Failure Mode Descriptions	.FMD	Needed if feature is used.
Basic Event Type Descriptions	.CTD	Needed if feature is used.
System Type Descriptions	.STD	Needed if feature is used.
Location Descriptions	.LCD	Needed if feature is used.
Train Types Descriptions	.CAD	Needed if feature is used.

Fault Tree Information

Mar-D File Description	Extension	General Guidance
Fault Tree Names/Descriptions	.FTD	Load prior to other fault tree/event tree files.
Fault Tree Graphics	.DLS	Loads graphic and associated logic.
Fault Tree Logic	.FTL	Not needed if DLS loaded. If .DLS is not used, then associated descriptions come from the .BED and .GTD files.
Fault Tree Cut Sets	.FTC	Generally not used since SAPHIRE can generate the cut sets.
Fault Tree Attributes	.FTA	Usually not needed.
Fault Tree Textual Information	.FTT	Descriptive information only.
Fault Tree Graphical P&ID	.PID	Needed if feature is used.
Fault Tree Recovery Rules	.FTY	Needed if feature is used.

Basic Event Information

Mar-D File Description	Extension	General Guidance
Basic Event Names/Descriptions	.BED	Load prior to other basic event information files.
Basic Event Rate Information	.BEI	Usually needed.
Basic Event Attribute Codes	.BEA	Needed if feature is used.
Basic Event Transformations	.BET	Needed if feature is used.
Basic Event Compound	.BEC	Needed if feature is used.
Basic Event Text	.BEN	Needed if feature is used.

Event Tree Information

Mar-D File Description	Extension	General Guidance
Event Tree Names/Descriptions	.ETD	Load prior to other event tree/sequence info.
Event Tree Attributes	.ETA	Usually needed (specifies initiating event – event tree correspondence).
Event Tree Graphics	.ETG	Load either this file or .ETL.
Event Tree Logic	.ETL	Load either this file or .ETG.
Event Tree Rules	.ETR	Needed if feature is used.
Event Tree Textual Information	.ETT	Descriptive information only.
Event Tree Recovery Rules	.ETY	Needed if feature is used.
Event Tree Partition Rules	.ETP	Needed if feature is used.

End State Information

Mar-D File Description	Extension	General Guidance
End State Names and Descriptions	.ESD	Load prior to other end state information files.
End State Information	.ESI	Undefined MAR-D file.
End State Textual Information	.EST	Descriptive information only.
End State Cut Sets	.ENC	Not needed since SAPHIRE can resolve cut sets.

Sequence Information

Mar-D File Description	Extension	General Guidance
Sequence Names and Descriptions	.SQD	Load prior to other sequence information files.
Sequence Cut Sets	.SQC	Not needed since SAPHIRE can resolve cut sets.
Sequence Attributes	.SQA	Needed to specify sequence-to-FLAG SETS relationships (if used).
Sequence Logic	.SQL	Not needed if event tree exists (can Link tree).
Sequence Textual Information.	.SQT	Descriptive information only.
Sequence Recovery Rules	.SQY	Needed if feature is used.
Sequence Partition Rules	.SQP	Needed if feature is used.

Gate

Mar-D File Description	Extension	General Guidance
Gate Description	.GTD	Descriptive information (will be needed for gate text to appear if .FTLs used instead of .DLSs).
Gate Attributes	.GTA	Usually not needed because the information is loaded via the .DLS or .FTL.

Change Sets

Mar-D File Description	Extension	General Guidance
Change Set Description	.CSD	Descriptive information only.
Change Set Information	.CSI	Needed if feature is used.
Change Set Attributes	.CSA	Needed if feature is used.

Histograms

Mar-D File Description	Extension	General Guidance
Histogram Description	.HID	Descriptive information only.
Histogram Information	.HII	Needed if feature is used.
Histogram Attributes	.HIA	Needed if feature is used.

Slice

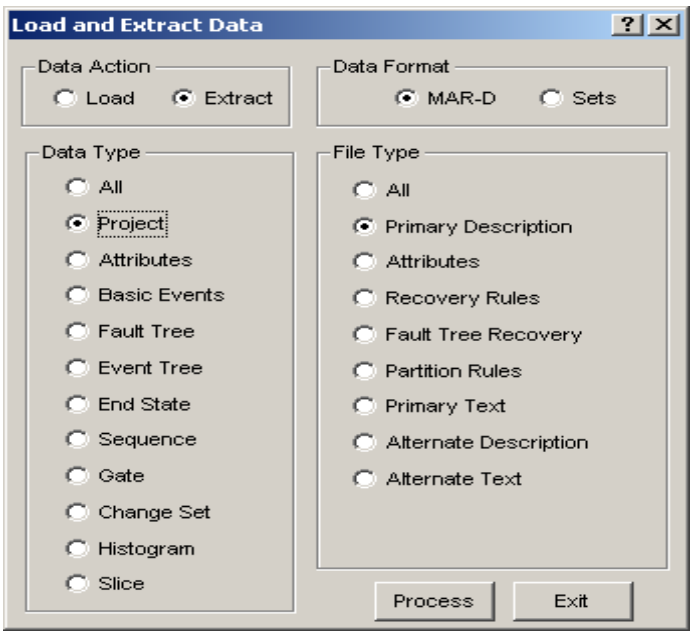
Mar-D File Description	Extension	General Guidance
Slice Names and Descriptions	.SLD	Sliced (partitioned) group of cut sets name and descriptive information.
Slice Basic Events	.SLB	Sliced (partitioned) cut set basic events.
Slice Information	.SLI	General Slice (partition) information.
Slice Attributes	.SLA	General Slice (partition) information.

- Note that the guidance provided above is of a general nature and is intended to provide insights into when it is necessary to load the particular MAR-D file. However, the particular needs of the user and characteristics of the model will determine the optimal combination of MAR-D files that should be loaded.

12.3. MAR-D Load and Extract Menus

The MAR-D menus are provided via the **Utility → Load and Extract** option (or Utility → MAR-D if using the toolbar icons) menu.

- The “Load” option allows you to load MAR-D files contained in the project’s subdirectory. The files to load must have the designated 3-character extension (e.g., .FTL) and format must conform to MAR-D specification.
- The “Extract” option allows you to save MAR-D files from the database to the hard drive.

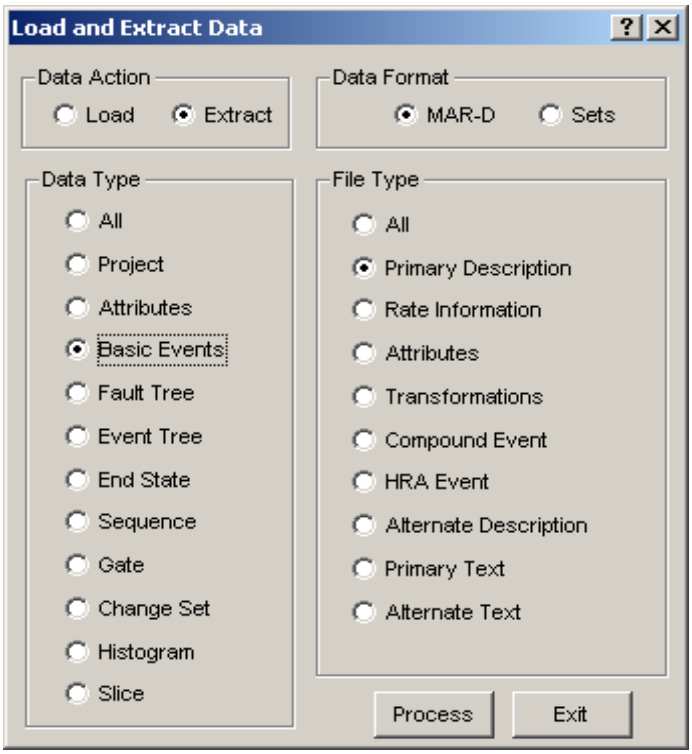


To extract information from SAPHIRE click the **Extract** radio button, then select what information you want extracted by clicking that particular radio button.

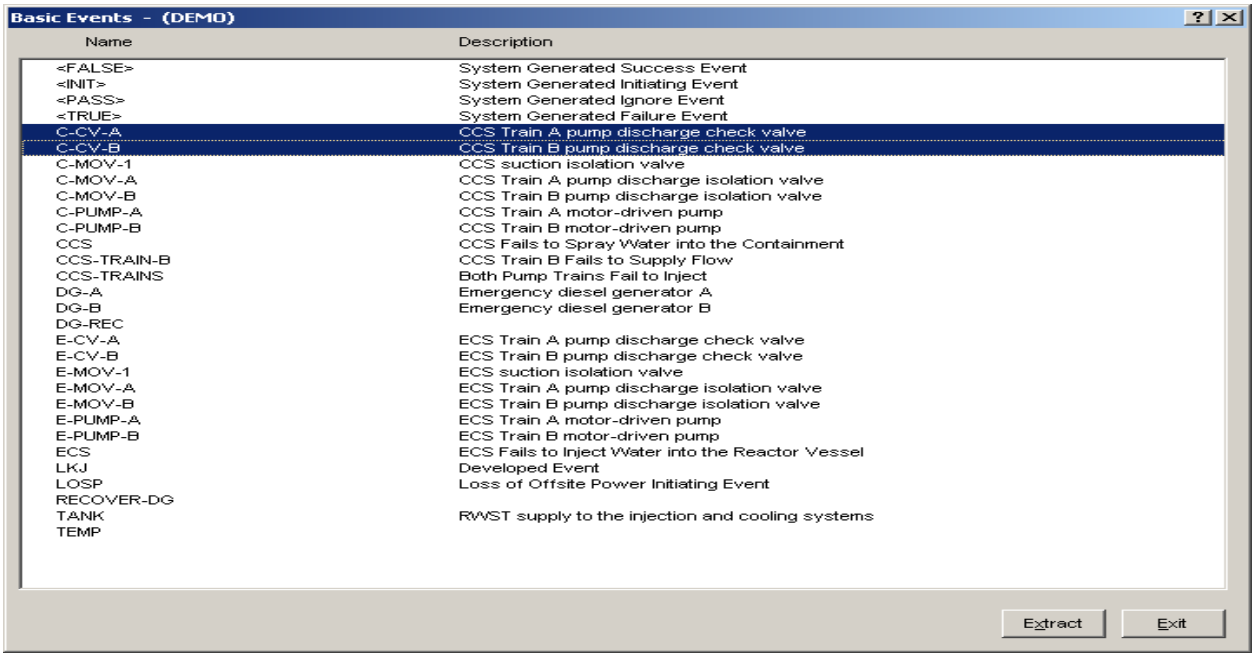
- The basic event information MAR-D file was selected as shown. Click **Extract → Basic Events → Primary Description** radio buttons. Then, press the **Process** button.

  You will then be presented with the list of all basic events.

  You can mark individual basic events, a range of basic events, or all of the basic events, then click **Extract**.

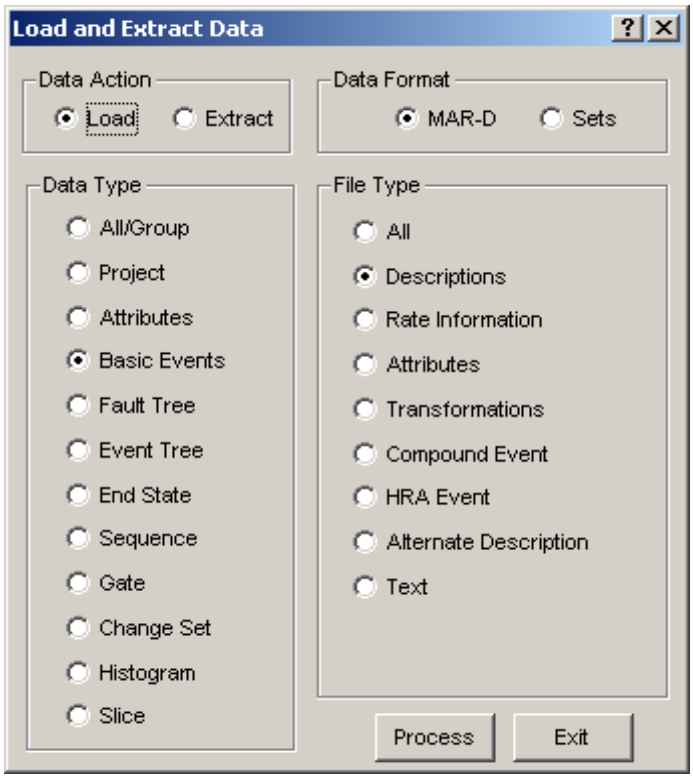


You will be prompted to accept or change the MAR-D file name. The default name is usually the *project* name and the MAR-D file 3-character extension, e.g., DEMO.BED.



Note that the file will be saved in the project subdirectory.

To load information back into SAPHIRE, the MAR-D steps are similar to those shown for the extract process. First, indicate that the **Load** function is to be used, then click the desired radio buttons to load that information. Click **Process** to continue.



The files in the project directory that have the extension .BED will be listed on the screen. Highlight the applicable file and click **Load** to load it into the project.

Examples of MAR-D Files

DEMO.BED

```
DEMO            =
C-CV-A         ,CCS Train A pump discharge check valve
C-CV-B         ,CCS Train B pump discharge check valve
C-MOV-1        ,CCS suction isolation valve
C-MOV-A        ,CCS Train A pump discharge
C-MOV-B        ,CCS Train B pump discharge
C-PUMP-A       ,CCS Train A motor-driven
C-PUMP-B       ,CCS Train B motor-driven
CCS             Developed Event
DG-A
DG-B
E-CV-A
E-CV-B
E-MOV-1
E-MOV-A
E-MOV-B
```

DEMO.BEI

```
DEMO            =
*  Name        ,FdT,UdC,UdT, UdValue , Prob      , Lambda    , Tau       , Mission ,Cat,PF, UdValue2
<FALSE>        ,1,   , ,------E----,+0.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, _ , ,------E----
<INIT>         ,1,   , ,------E----, 1.000E+000,+0.000E+000
<PASS>         ,1,   , ,------E----, 1.000E+000,+0.000E+000
<TRUE>         ,1,   , ,------E----, 1.000E+000,+0.000E+000
C-CV-A         ,1,1  ,L, 3.000E+000, 1.000E-004,+0.000E+000
C-CV-B         ,1,1  ,L, 3.000E+000, 1.000E-004,+0.000E+000
C-MOV-1        ,1,3  ,L, 5.000E+000, 1.000E-003,+0.000E+000
C-MOV-A        ,1,2  ,L, 5.000E+000, 5.000E-003,+0.000E+000
C-MOV-B        ,1,2  ,L, 5.000E+000, 5.000E-003,+0.000E+000
C-PUMP-A       ,1,4  ,L, 5.000E+000, 3.000E-003,+0.000E+000
```

DEMO.FTL

```
DEMO, CCS =
CCS             OR   CCS-SUPPLY CCS-TRAINS
CCS-SUPPLY      OR   C-MOV-1-FAILS TANK
C-MOV-1-FAILS   OR   DG-B C-MOV-1
CCS-TRAINS      AND  CCS-TRAIN-A CCS-TRAIN-B
CCS-TRAIN-A     OR   C-CV-A C-MOV-A DG-A C-PUMP-A
CCS-TRAIN-B     OR   C-CV-B C-MOV-B C-PUMP-B DG-B
^EOS
DEMO, ECS =
ECS             OR   ECS-SUPPLY ECS-TRAINS
ECS-TRAINS      AND  ECS-TRAIN-A ECS-TRAIN-B
ECS-SUPPLY      OR   E-MOV-1-FAILS TANK
E-MOV-1-FAILS   OR   DG-A E-MOV-1
ECS-TRAIN-A     OR   E-CV-A E-MOV-A DG-A E-PUMP-A
ECS-TRAIN-B     OR   E-CV-B E-MOV-B E-PUMP-B DG-B
```

12.4. Creating a New Project Using MAR-D Files

In this example, a new project that replicates the DEMO database will be reconstructed from MAR-D files.

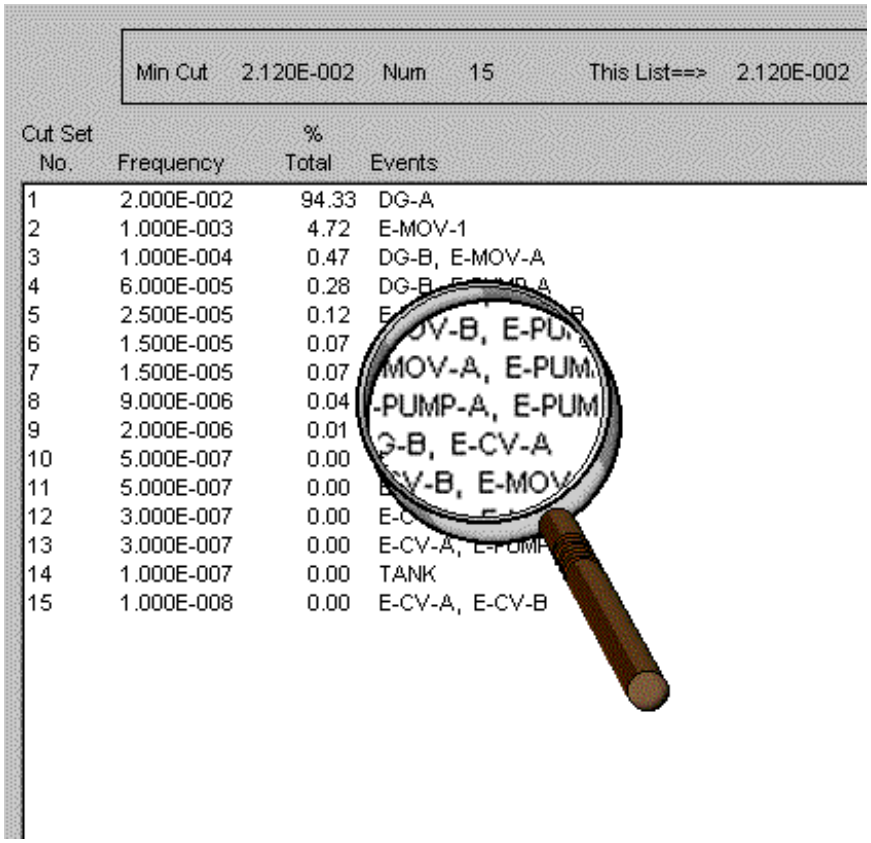
> Note that other projects may require additional MAR-D files to be extracted/loaded depending on the actual SAPHIRE features utilized.

Step	Description
1	Extract the following MAR-D files from the DEMO database: DEMO.BED Basic Event Names and Descriptions DEMO.BEI Basic Event Rate Information DEMO.BEA Basic Event Attributes DEMO.FTD Fault Tree Names and Descriptions DEMO.FTL Fault Tree Logic DEMO.ETD Event Tree Names and Descriptions DEMO.ETA Event Tree Attributes DEMO.GTD Gate Descriptions LOSP. ETG The LOSP Event Tree Note that the .ETG file is created via the graphical editor. If it has been deleted from the directory, it still exists within the SAPHIRE relational database and can be extracted by using MAR-D (or from **Utilities → Extract Event Trees**).
2	Make a new folder (in Windows) called DEMO-M. To make a folder, start *Windows Explorer*™, navigate to the SAPHIRE directory (e.g., C:\Saphire7), click the File menu, select "New Folder," and type in DEMO-M.
3	Copy the files from Step 1 (in the DEMO folder, e.g., C:\ Saphire7\DEMO) to the new DEMO-M folder. To copy files, use *Windows Explorer*™ to navigate to the DEMO folder, and highlight the files, click the right mouse button, and then select "copy." Now, navigate to the DEMO-M folder, click the right mouse button, and select "paste."
4	Invoke SAPHIRE by going to Start → Programs → Saphire for Windows and click the SAPHIRE icon (or double click the SAPHIRE icon if it is on the desktop). Make a new project, called DEMO-M, by clicking the **New** button or **File → New Project**. This new project will automatically be selected.
5	Load the MAR-D files into DEMO-M from the **Utilities → Load and Extract** MAR-D option. Load the files in the order that they were listed in Step 1.
6	Generate event data from the **Generate → Generate** data menu.

Step	Description
7	Recover the database from the **Utilities → Recover Data Base** menu. (This step is optional, but recommended, since it ensures that the relational data files have been correctly written to the hard drive.)
8	To be able to view the fault trees graphics, we need to have SAPHIRE make the graphic from the fault tree logic. To make the fault tree graphic, use the **Utilities → Alpha to Graphics** menu to the convert the logic into a .DLS file. Note that an alternate approach (to the .FTL method) is to use the .DLS file from the DEMO project. The .DLS may be loaded via the **Utilities → Load Fault Trees** option.
9	Link the event tree using the **Event Tree → Link Trees** menu.
10	Now, the database is set up to be able to solve either fault tree or event tree cut sets. To solve fault tree cut sets, use the **Fault tree → Solve** option. To solve sequence cut sets, use the **Sequence → Solve** option.

13 VIEWING CUT SETS

Section 13 describes the cut set display feature that allows you to “slice” cut sets into different lists based on user-defined sort criteria. The sliced lists may then be viewed or reported. The cut set slicing features is available for fault tree, sequence, or end state cut sets.



13.1. The Cut Set Display Option

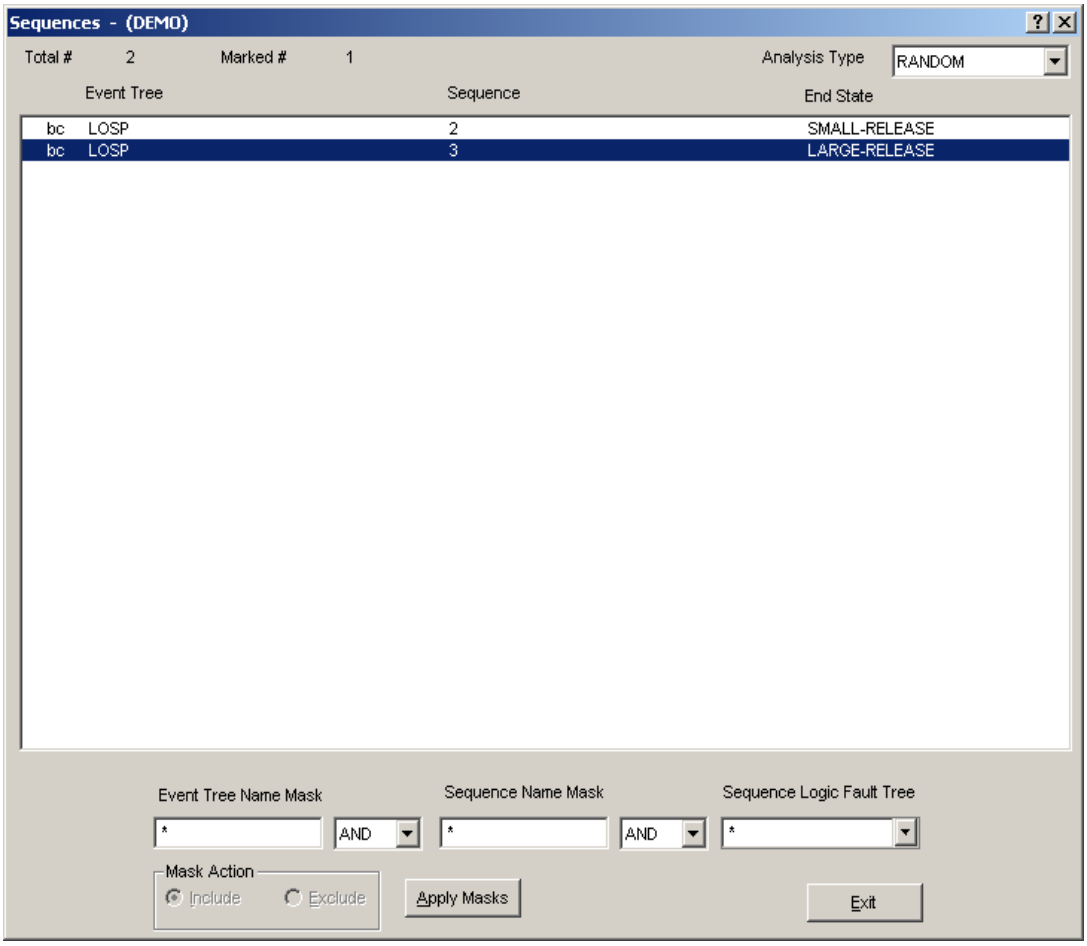
The cut set display option is available in the Fault Tree, Sequence, and End State menus.

> While one or more sequences or end states can be highlighted simultaneously to display their respective cut sets (as a group), only a single fault tree can be selected to display its cut sets.
>
> Note that if multiple sequences or end states are selected, all the cut sets from the highlighted sequences or end states will be displayed without eliminating any non-minimal cut sets for the group selected.

With the desired fault tree, sequences, or end states highlighted, then select the **Display → Cut Sets** option.

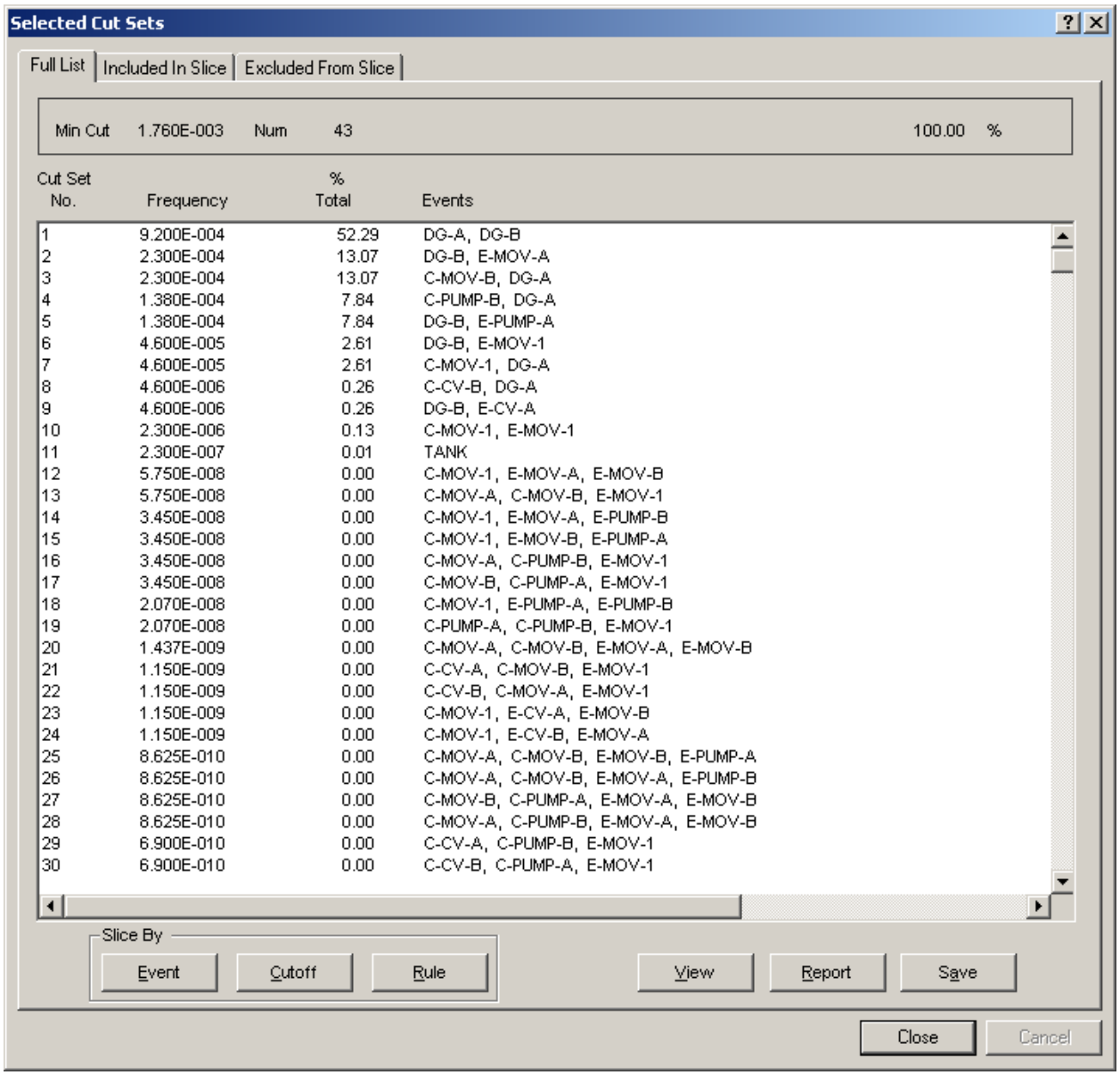
As an example, only the cut sets for LOSP sequence 3 are to be displayed following cut set generation using a truncation of 1.0E-10.

Select the **Sequence** menu then highlight LOSP sequence 3.



Now, select the **Display → Cut Sets** option.

The cut sets will then be displayed.

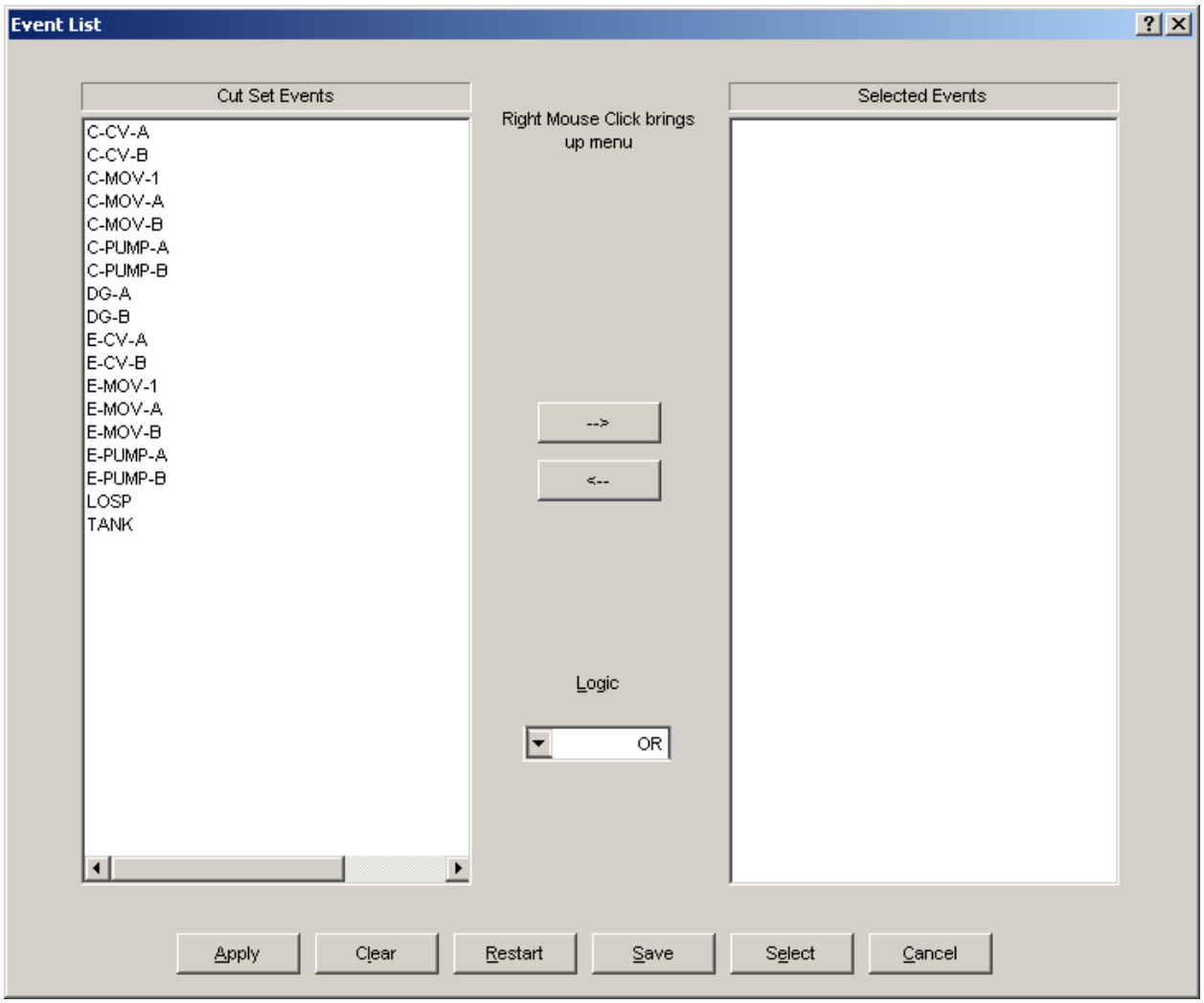


From the LOSP sequence 3 cut sets, the minimal cut set upper bound approximation frequency is 1.76E-3 and there are a total of 43 cut sets.

13.2. The Event Slice Option

The event slice option is used to subdivide the cut sets into two lists. This option allows you to specify individual or combinations of desired basic events to appear in the “included in slice” list of cut sets.

With the LOSP sequence 3 cut sets displayed, select **Event** in the “Slice By” box at the bottom of the cut set display window. A window with just the basic events in the cut sets will be displayed.



From the list of basic events, specify those of interest by:

- Right clicking the mouse; select the Wild Card Mark option; and then specify the name (you may use wild characters ? and *), basic event attribute, and/or susceptibilities. Then select **Ok**. All events meeting the search criteria will be highlighted and then either

  Right mouse click and select Add Event, or

  Use the → button.

- Highlighting individual basic events

  Right mouse click and select Add Event, or

  Use the → button.

All the marked events will then appear in the “Selected Events” window

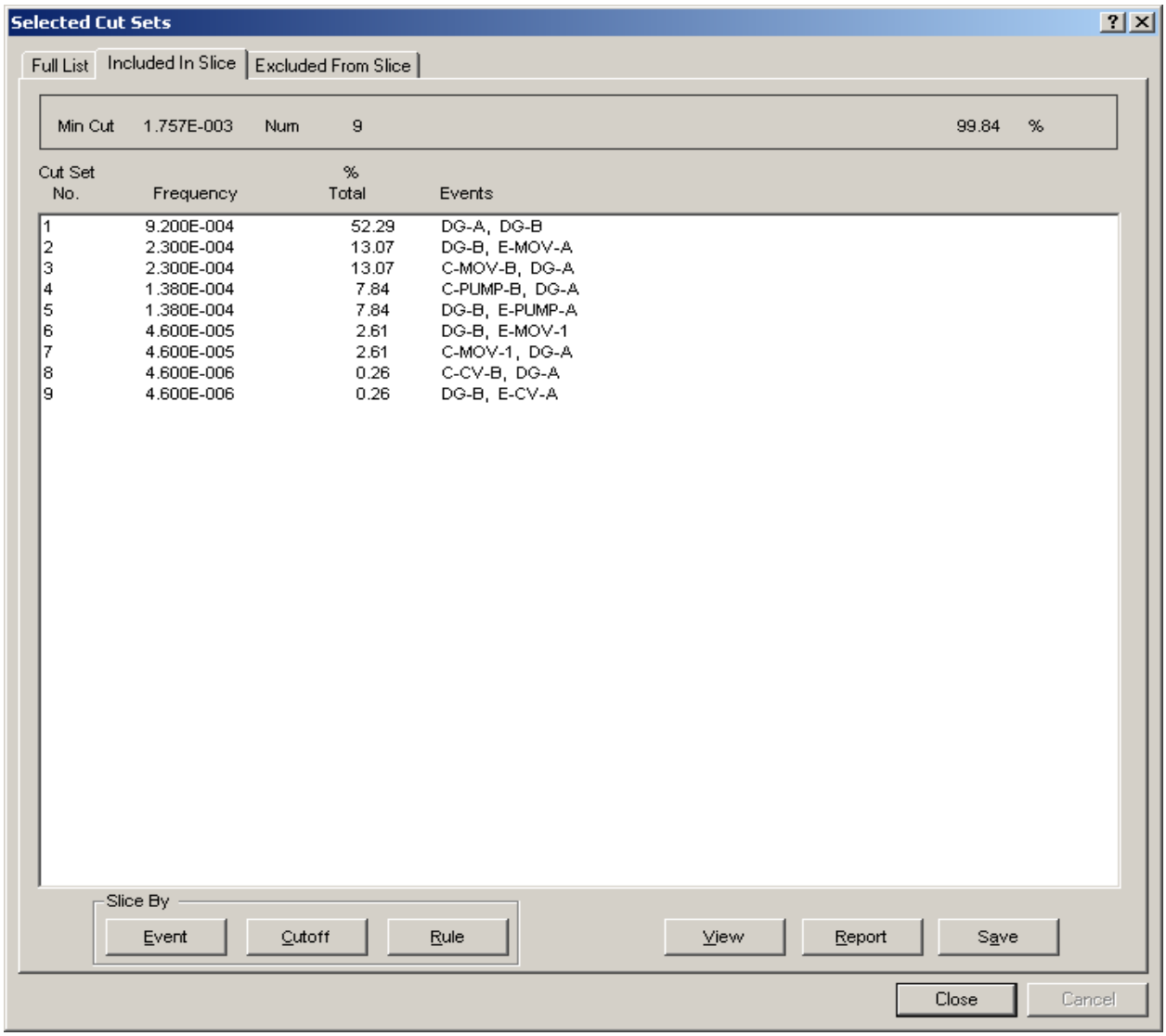
- The right mouse button also allows you to specify if you want cut sets that just contain the failed basic event, success basic event, or the selected event not contained in the cut sets.

- **OR** or **AND** criteria must be specified for the selected events, using the Logic option in the center of the window. For example, if you want to find cut sets that contain **all** of the events in the “Selected Events” window, use the OR option.

With the desired events indicated and “logic” option specified, select **Apply**.

At this point, a list of the cut sets meeting the event slice criteria will be displayed along with the quantified probability (or frequency), percent contribution to the total, and the number of cut sets meeting the event slice criteria.

  - DG-A and DG-B are the basic events that will be selected to show only those cut sets that contain these basic events.



The **View** option allows you to look at the events for a single cut set.

To obtain a report of the sliced cut sets, select the **Report** option and enter the desired report (i.e., to the screen, printer, or file). The event slice criteria can also be printed from the report option.

The **Save** option allows you to copy the on-screen list of cut sets directly to a user-specified end state for further analysis or storage.

Tabs on the top of display:

Full List – This tab when selected provides the list of all of the cut sets.

Included in Slice – This tab provides the list of cut sets that meet the search criteria.

Excluded from Slice – This tab provides the list of remaining cut sets that do not meet the search criteria.

Select the **Event** option in the "Slice By" box to return to the event selection window.

From the event selection window, several options are available at the bottom of the window. These options and their function include:

◊ Clear - clears the Selected Events field

◊ Restart - returns to the original list of cut sets

◊ Save - saves the slice criteria (i.e., selected events) for future use.

◊ Select - select a previously saved slice criteria

◊ Cancel - returns to the list of cut sets

13.3. Demonstration of the Event Slice Option

The example of the **Event Slice Option** will make use of the LOSP sequence 3 cut sets following cut set generation using a truncation of 1.0E-10.

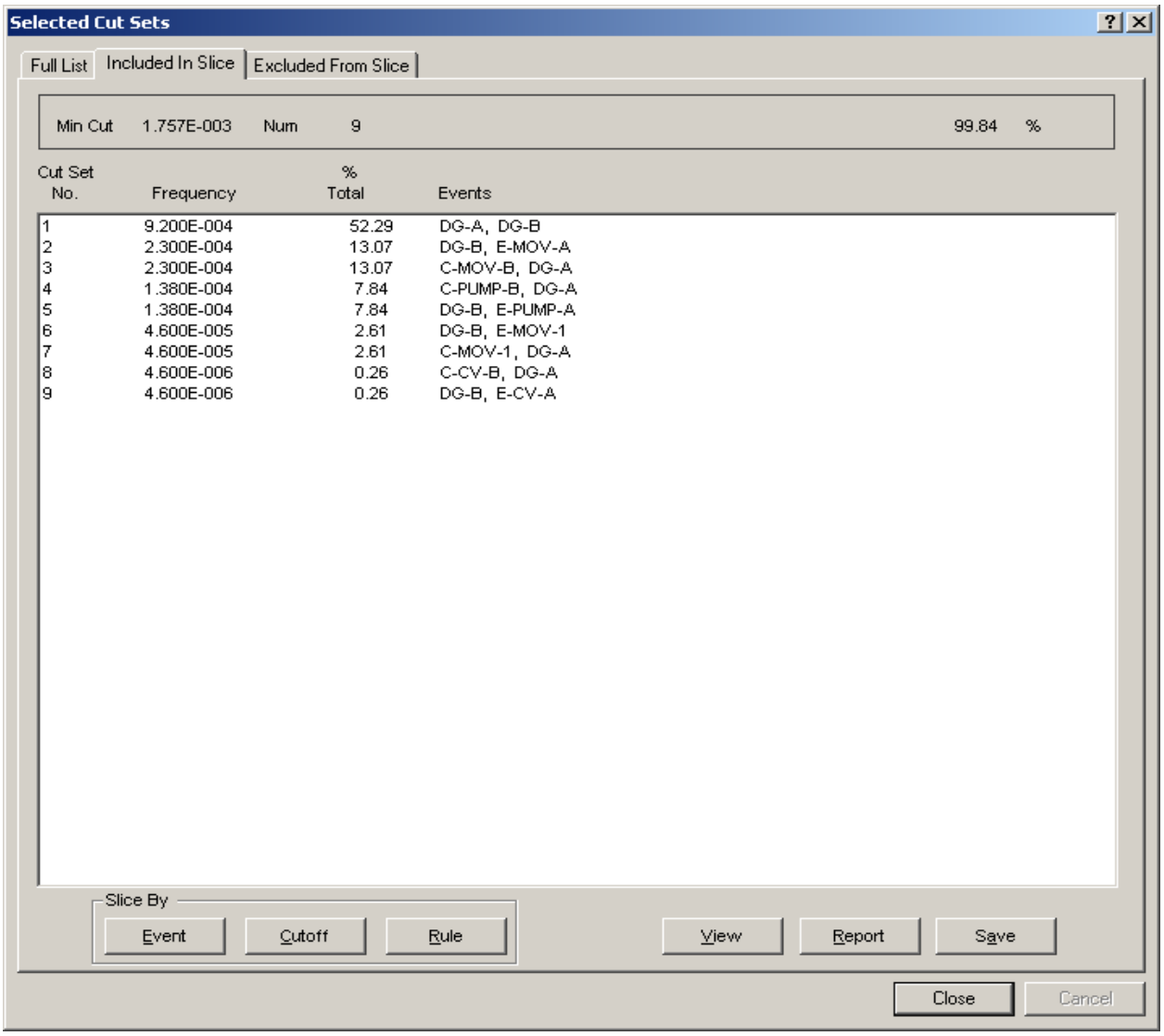
Select the **Sequence** menu, then highlight LOSP sequence 3.

Select the **Display** → **Cut Sets** option (min cut = 1.760E-3 with 43 cut sets).

Select the **Event** option in the "Slice By" box.

Add basic events DG-A and DG-B to the Selected Events list, and make sure the Logic is set to the "OR" option.

Select **Apply**. There are 9 cut sets that contain basic events DG-A or DG-B and their min cut upper bound is 1.757E-3, which is 99.84% of the total min cut upper bound for LOSP sequence 3.



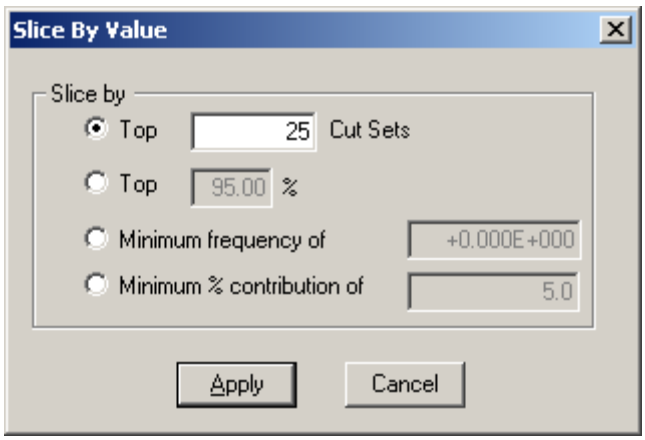
Now, select **Excluded From List**. The cut sets not containing *either* basic event DG-A or DG-B will be displayed.

There are 34 cut sets that do not contain basic events DG-A or DG-B and their min cut upper bound is 2.841E-6, which is 0.16% of the total min cut upper bound for LOSP sequence 3.

13.4. The Cutoff Slice Option

The cutoff slice option is used to display cut sets where truncation options will be used to parse the overall list of cut sets into two smaller lists.

With the LOSP sequence 3 cut sets displayed, select **Cutoff** in the "Slice By" box.



- The **Top (Cut Sets)** option will parse the overall list of cut sets down to only the *X* number of cut sets specified (i.e., if 25 is typed into the field, then only the top 25 cut sets will be displayed).

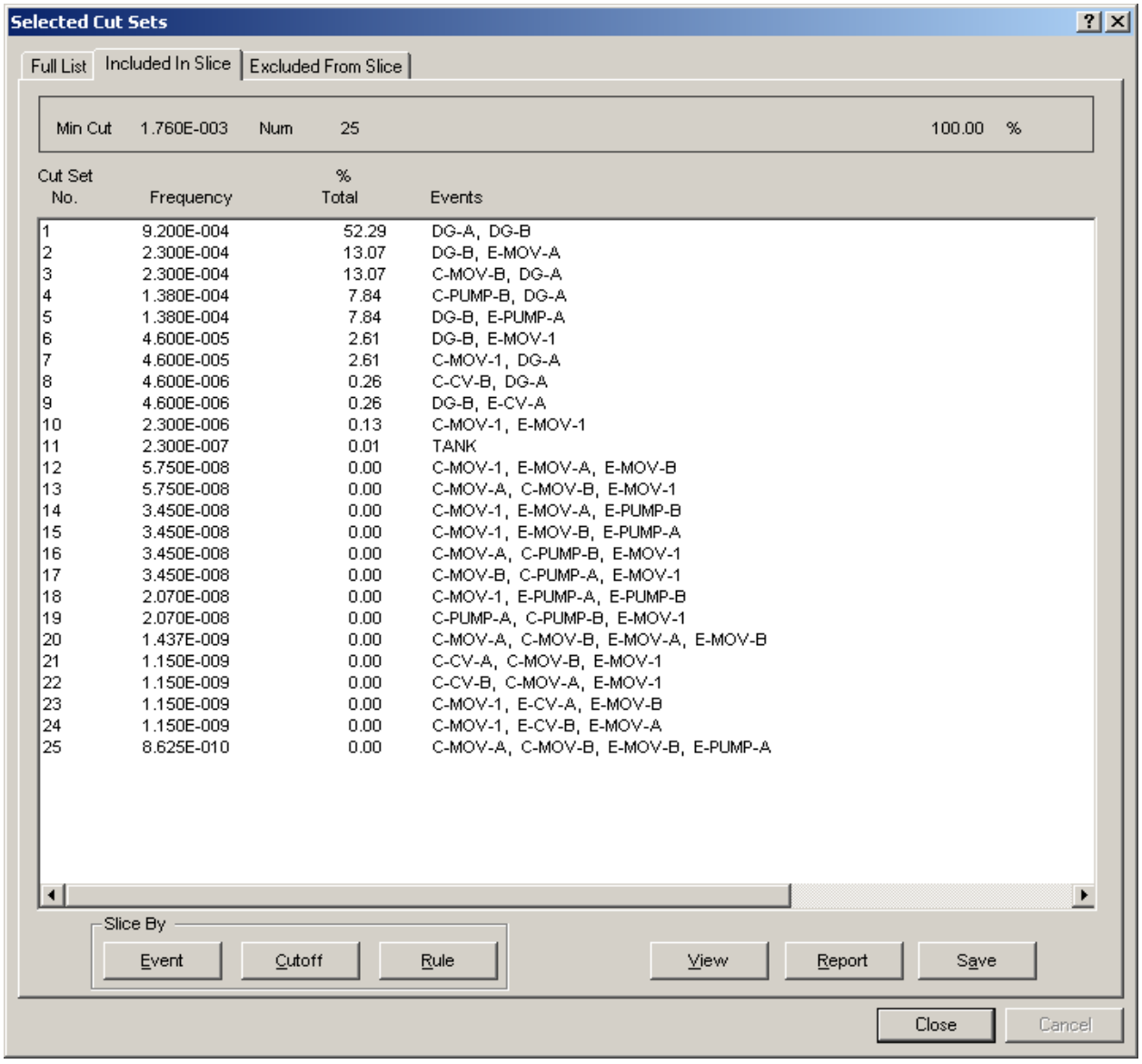
- The **Top (%)** option will parse the overall list of cut sets down to only the *Y* percent contributing cut sets specified (i.e., if 90% is typed into the field, then only the top 90 percent cut sets will be displayed).

- The **Minimum frequency of** option will parse the overall list of cut sets down to only those cut sets above the specified truncation level (i.e., if 1E-06 is typed into the field, then only those cut sets above the truncation level of 1E-06 will be displayed).

- The **Minimum percent contribution** option will parse the overall list of cut sets down to only those cut sets that contribute *Z* percent to the overall probability (i.e., if 5% is typed into the field, then only those cut sets that contribute *at least* 5% to the overall probability will be displayed).

With the desired cutoff option selected and the specific truncation level specified, select **Apply**.

The cut sets meeting the cutoff slice criteria will be displayed along with the quantified probability (or frequency), percent contribution to the total, and the number of cut sets meeting the cutoff slice criteria. In this case the top 25 cut sets was selected.

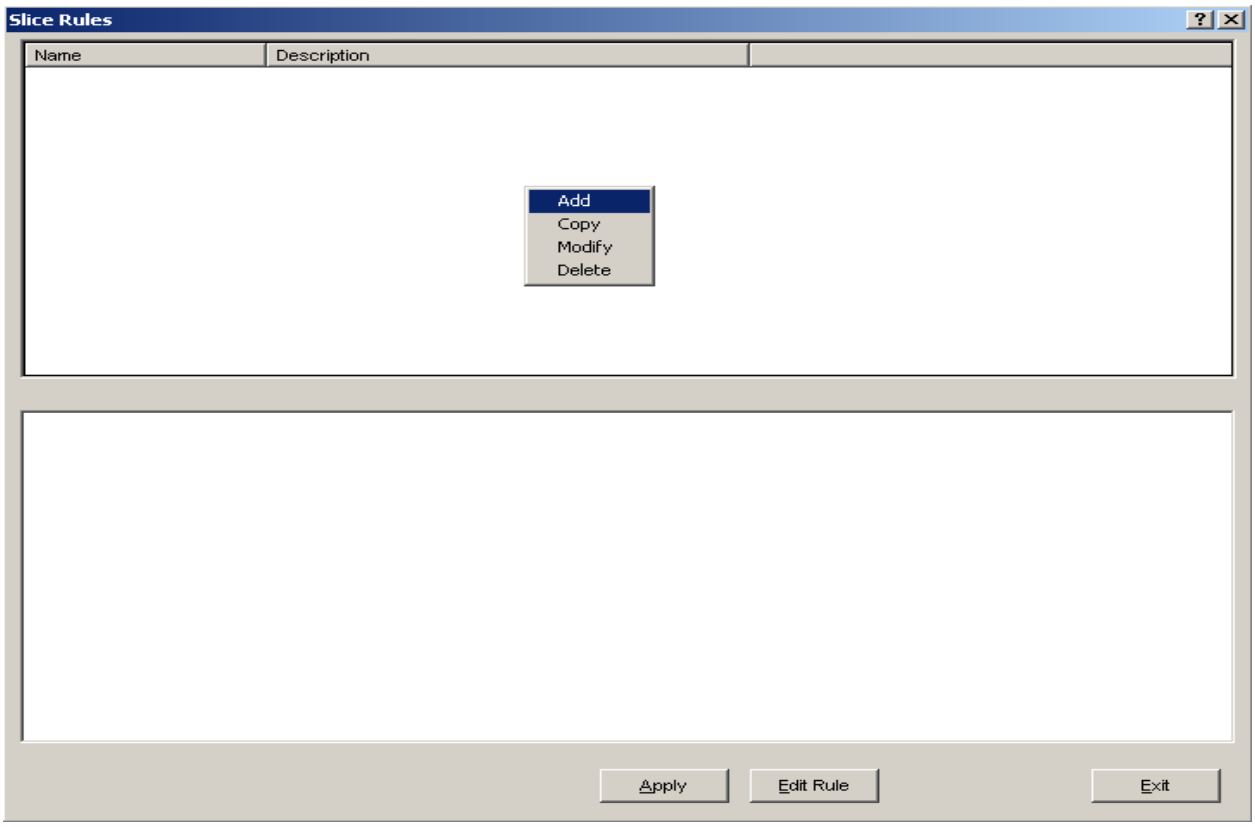


Cut Set No.	Frequency	% Total	Events
1	9.200E-004	52.29	DG-A, DG-B
2	2.300E-004	13.07	DG-B, E-MOV-A
3	2.300E-004	13.07	C-MOV-B, DG-A
4	1.380E-004	7.84	C-PUMP-B, DG-A
5	1.380E-004	7.84	DG-B, E-PUMP-A
6	4.600E-005	2.61	DG-B, E-MOV-1
7	4.600E-005	2.61	C-MOV-1, DG-A
8	4.600E-006	0.26	C-CV-B, DG-A
9	4.600E-006	0.26	DG-B, E-CV-A
10	2.300E-006	0.13	C-MOV-1, E-MOV-1
11	2.300E-007	0.01	TANK
12	5.750E-008	0.00	C-MOV-1, E-MOV-A, E-MOV-B
13	5.750E-008	0.00	C-MOV-A, C-MOV-B, E-MOV-1
14	3.450E-008	0.00	C-MOV-1, E-MOV-A, E-PUMP-B
15	3.450E-008	0.00	C-MOV-1, E-MOV-B, E-PUMP-A
16	3.450E-008	0.00	C-MOV-A, C-PUMP-B, E-MOV-1
17	3.450E-008	0.00	C-MOV-B, C-PUMP-A, E-MOV-1
18	2.070E-008	0.00	C-MOV-1, E-PUMP-A, E-PUMP-B
19	2.070E-008	0.00	C-PUMP-A, C-PUMP-B, E-MOV-1
20	1.437E-009	0.00	C-MOV-A, C-MOV-B, E-MOV-A, E-MOV-B
21	1.150E-009	0.00	C-CV-A, C-MOV-B, E-MOV-1
22	1.150E-009	0.00	C-CV-B, C-MOV-A, E-MOV-1
23	1.150E-009	0.00	C-MOV-1, E-CV-A, E-MOV-B
24	1.150E-009	0.00	C-MOV-1, E-CV-B, E-MOV-A
25	8.625E-010	0.00	C-MOV-A, C-MOV-B, E-MOV-B, E-PUMP-A

13.5. The Rule Slice Option

The **Rule** slice option provides the analyst flexibility on slicing the displayed cut sets into more specific groups for further review or reporting.

With the LOSP sequence 3 cut sets displayed, select **Rule** in the "Slice By" box.



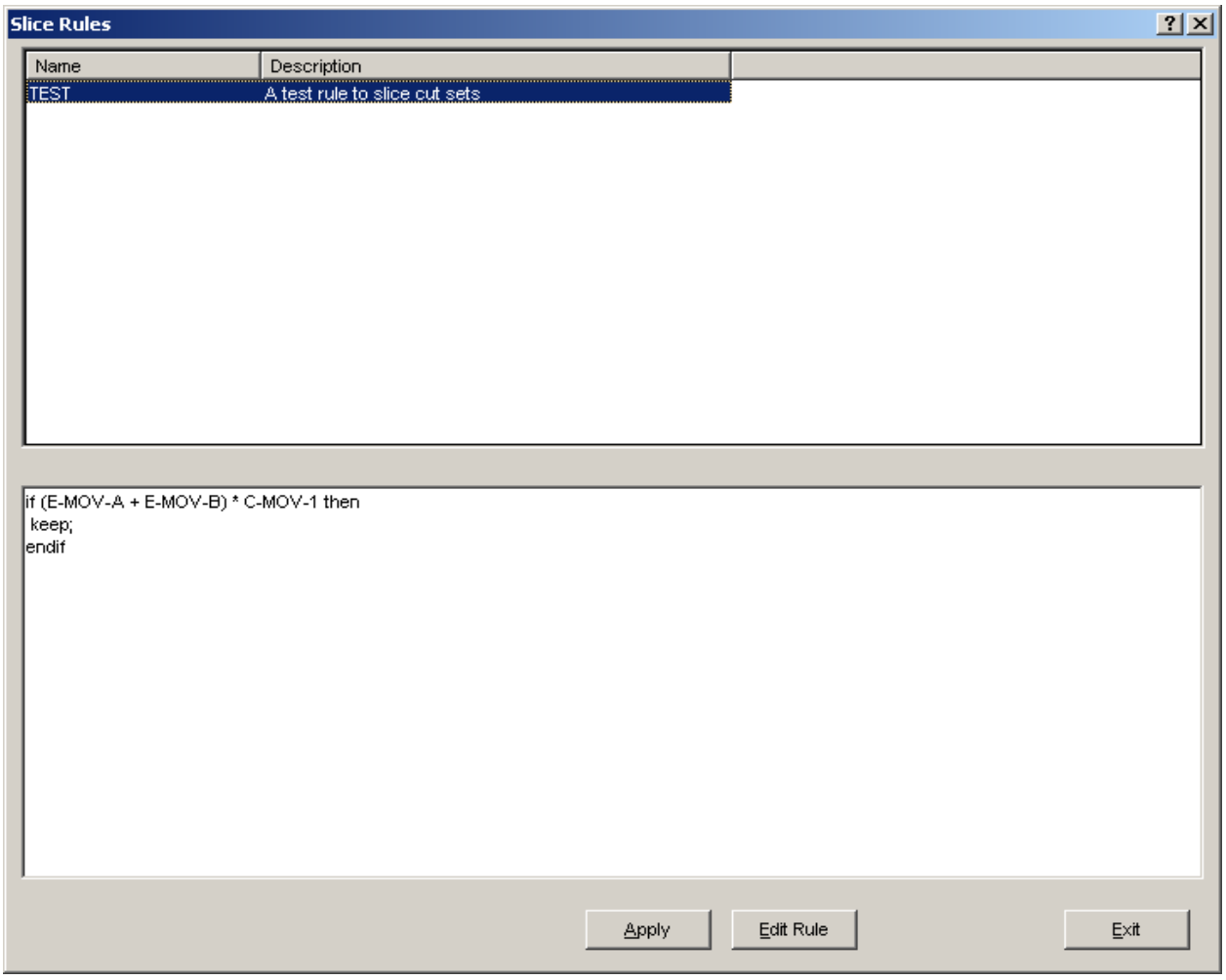
- The screen shown above will pop up. From this screen, the slice rules are created, modified and applied.
  - ◊ Within the rule editor, right mouse click and select **Add**.
  - ◊ The rule name and its description need to be added in the appropriate fields, then select **Ok**.
  - ◊ Highlight the rule name, then select **Edit Rules**. This rule editor is similar as those previously discussed.

◊ The slice rule can now be entered. The following rule was added to the rule editor:

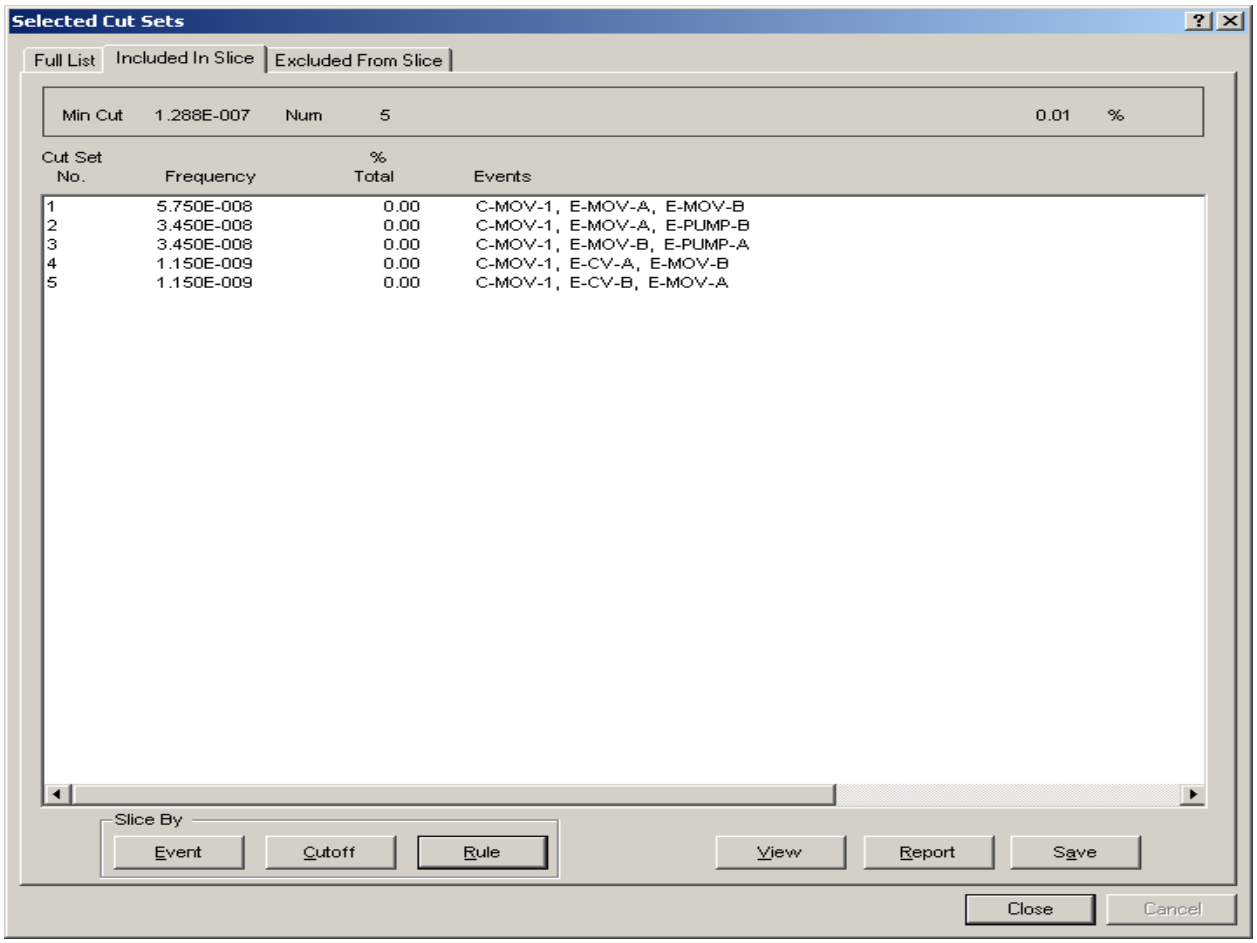
if (E-MOV-A + E-MOV-B) * C-MOV-1 then
keep;
endif

This rule will search for combinations of E-MOV-A and C-MOV-1 or E-MOV-B and C-MOV-1.

◊ After the rule is entered, it is saved and compiled. Once the rule is compiled it shows up in the lower part of the rule editor screen.



◊ The rule is applied to the displayed cut sets by clicking **Apply**.



- ◊ The cut sets in the “Include In Slice” tab are those cut sets that met the search criteria.

♦ The use of this rule editor is more flexible at slicing cut sets into groups than using the logical “OR” or “AND” operator required when using the **Event** option.

♦ This option contains two keywords used in the rule development. The keywords are:

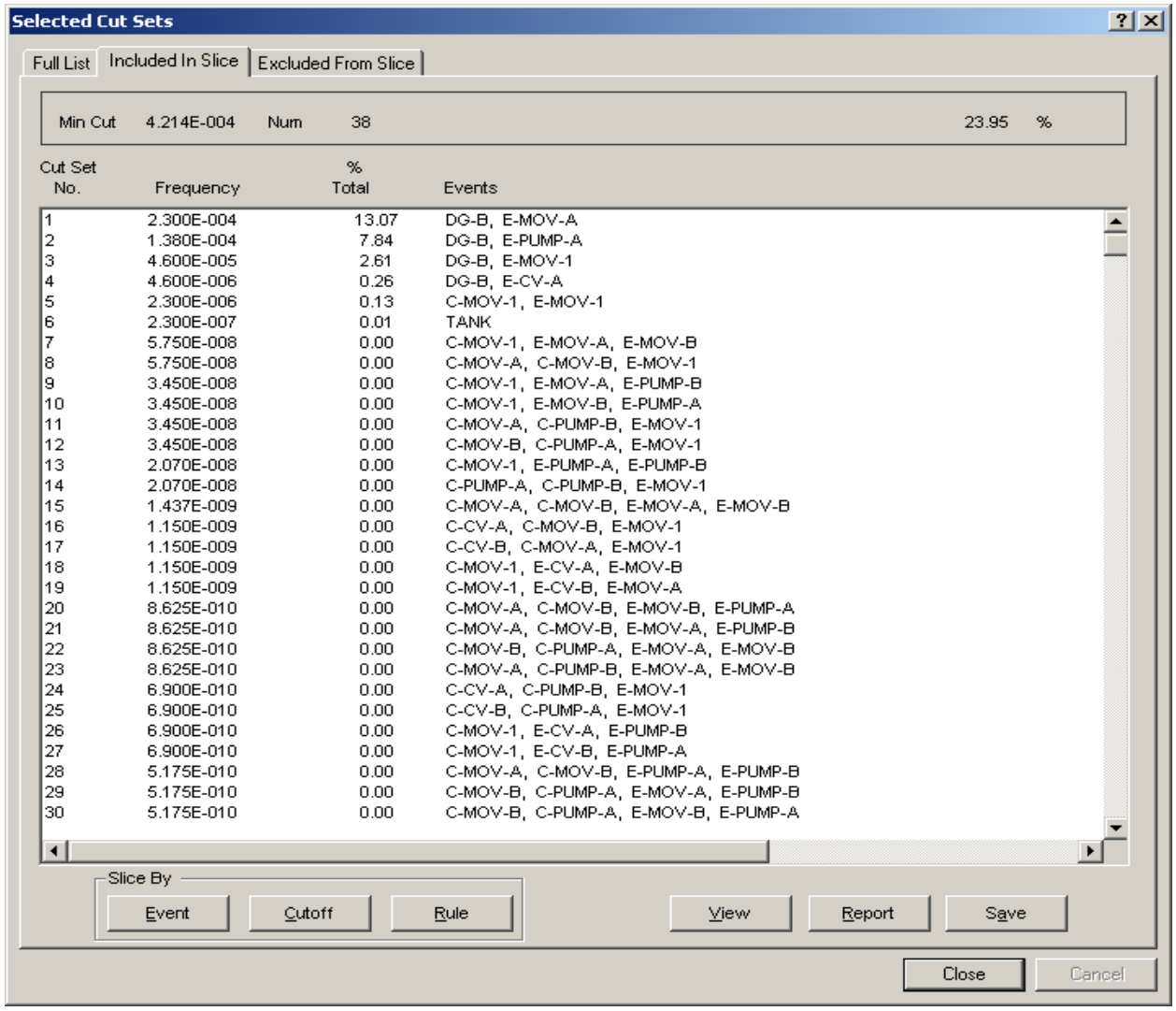
- ◊ **Keep** – This keyword tells SAPHIRE to include all cut sets that meet the search criteria. These cut sets are stored in the “Included In Slice” tab.

- ◊ **Discard** – This keyword tells SAPHIRE to exclude all cut sets that meet the search criteria. This keyword requires the addition of the **else** statement telling SAPHIRE to **keep** the remaining cut sets in the “Included In Slice” tab and the search criteria cut sets to be placed in the “Excluded From Slice”.

Example Rule:

```
if DG-A then
  discard;
else
  keep;
endif.
```

The output cut sets that are shown below will have all of the cut sets that do not contain DG-A in the “Included In Slice” tab and the cut sets that contain DG-A will be in the “Excluded From Slice” tab.



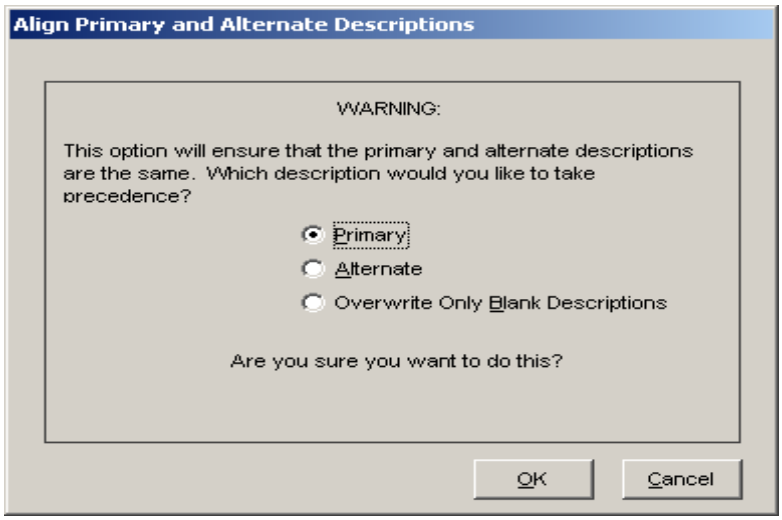
14 Utility Menu Option

Section 14 describes the different options located under the Utility drop down list. Each of these options will be discussed.

14.1. Utility Options

The first group of applications separated by the dissecting line is discussed below.

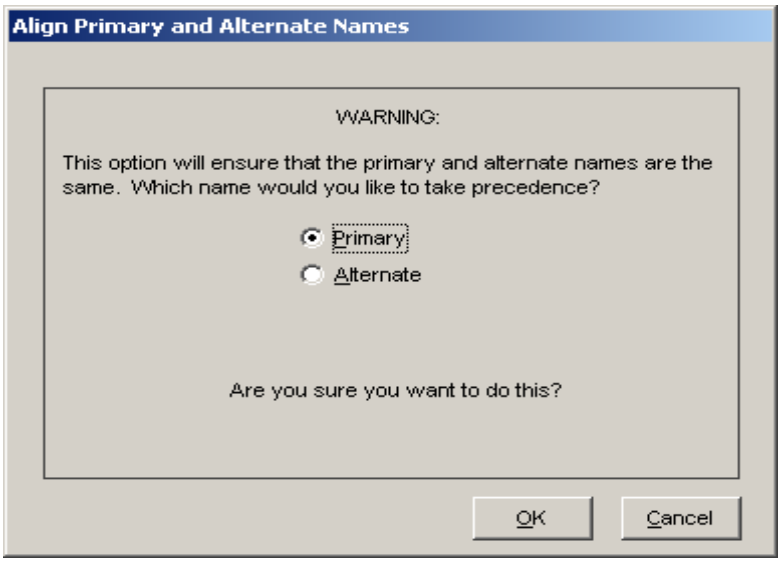
- The first option is the define constants. This option has been discussed in the SAPHIRE Basics course.
- The second option is the MAR-D function (i.e., Load and Extract). This option was discussed in Section 12 of this manual.
- The third option is recover database. This option has been discussed in the SAPHIRE Basics course.
- The fourth option (Update Description) is a new option that provides the analyst a means to copy the primary description or alternate description into the opposite description fields for all basic events, fault trees, and event trees.
  - The first radio button will copy the primary description over to the alternate description. This process will overwrite any description that may already be in the field or fill the empty field with the primary description for all basic events, fault trees, and event trees.
  - The second radio button will copy the alternate description over the primary description for all basic events, fault trees, and event trees.
  - The last radio button will only overwrite the blank description fields with the information in the primary description field.



- The next option (Align Names) is an option that provides the analyst a means to make the alternate name the same as the primary name or visa versa for all basic events, fault trees, and event trees.

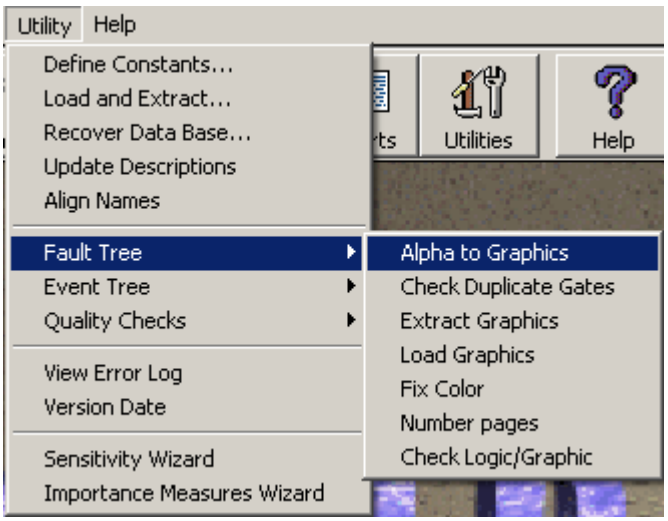
  - The first radio button will copy the primary name over to the alternate name. This process will overwrite any name that may already be in the field or fill the empty field with the primary name for all basic events, fault trees, and event trees.

  - The second radio button will copy the alternate name over the primary name for all basic events, fault trees, and event trees.



The next group of applications that is separated from the others by the dissecting line is fault tree and event tree. These options are discussed below.

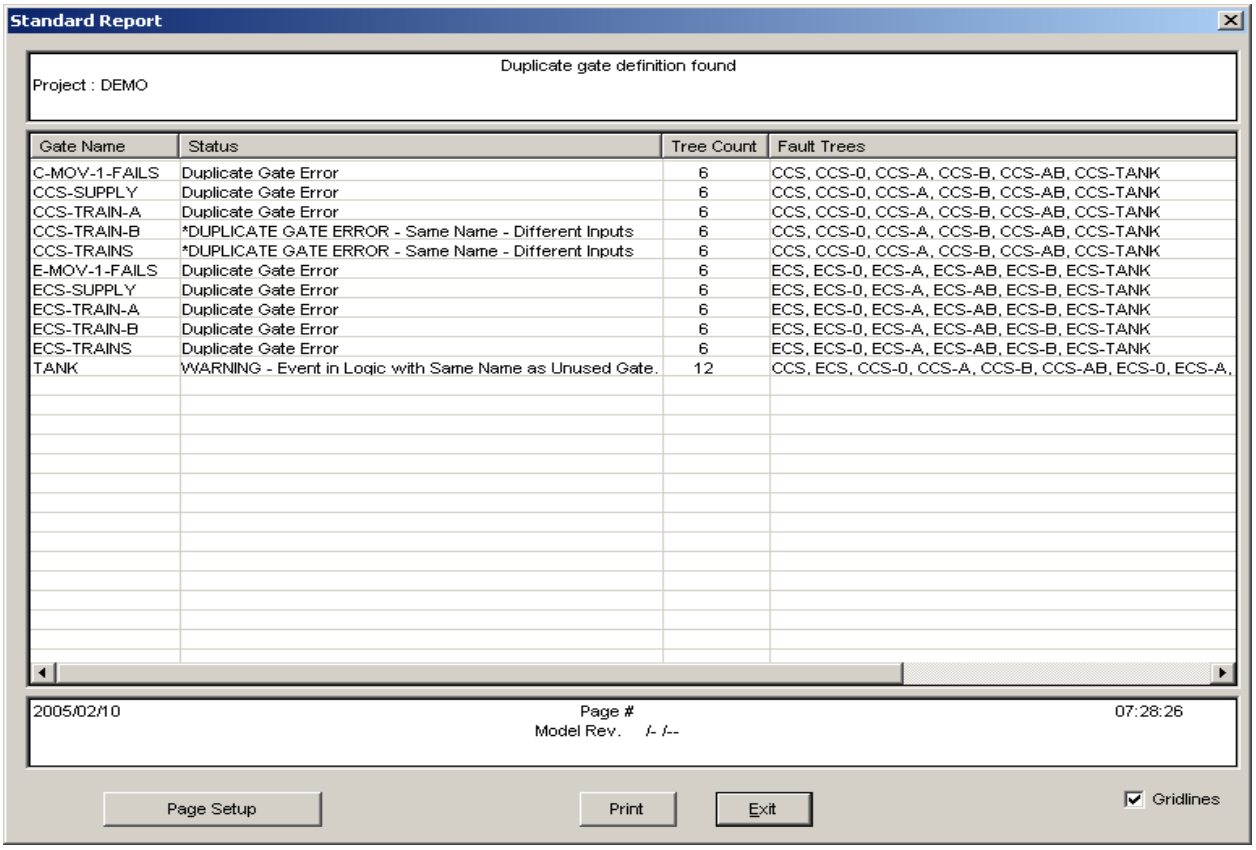
Fault Tree Options



- The first option under fault tree is converting the alpha numeric logic into fault tree graphics. This option has been discussed in the SAPHIRE Basics course.

- The next option is check duplicate gates. This option searches all of the fault trees in the project and lists those fault trees that contain gates with the same name. This is a check to make sure there will not be any logic problems that will arise during fault tree/event tree sequence cut set generation.

  - If this option is selected, a report is generated listing all of the duplicate gates and what fault trees they are located in. These gates may need to be modified prior to fault tree or event tree sequence cut set generation. The following is the report that is generated.



Gate Name	Status	Tree Count	Fault Trees
C-MOV-1-FAILS	Duplicate Gate Error	6	CCS, CCS-0, CCS-A, CCS-B, CCS-AB, CCS-TANK
CCS-SUPPLY	Duplicate Gate Error	6	CCS, CCS-0, CCS-A, CCS-B, CCS-AB, CCS-TANK
CCS-TRAIN-A	Duplicate Gate Error	6	CCS, CCS-0, CCS-A, CCS-B, CCS-AB, CCS-TANK
CCS-TRAIN-B	*DUPLICATE GATE ERROR - Same Name - Different Inputs	6	CCS, CCS-0, CCS-A, CCS-B, CCS-AB, CCS-TANK
CCS-TRAINS	*DUPLICATE GATE ERROR - Same Name - Different Inputs	6	CCS, CCS-0, CCS-A, CCS-B, CCS-AB, CCS-TANK
E-MOV-1-FAILS	Duplicate Gate Error	6	ECS, ECS-0, ECS-A, ECS-AB, ECS-B, ECS-TANK
ECS-SUPPLY	Duplicate Gate Error	6	ECS, ECS-0, ECS-A, ECS-AB, ECS-B, ECS-TANK
ECS-TRAIN-A	Duplicate Gate Error	6	ECS, ECS-0, ECS-A, ECS-AB, ECS-B, ECS-TANK
ECS-TRAIN-B	Duplicate Gate Error	6	ECS, ECS-0, ECS-A, ECS-AB, ECS-B, ECS-TANK
ECS-TRAINS	Duplicate Gate Error	6	ECS, ECS-0, ECS-A, ECS-AB, ECS-B, ECS-TANK
TANK	WARNING - Event in Logic with Same Name as Unused Gate.	12	CCS, ECS, CCS-0, CCS-A, CCS-B, CCS-AB, ECS-0, ECS-A,

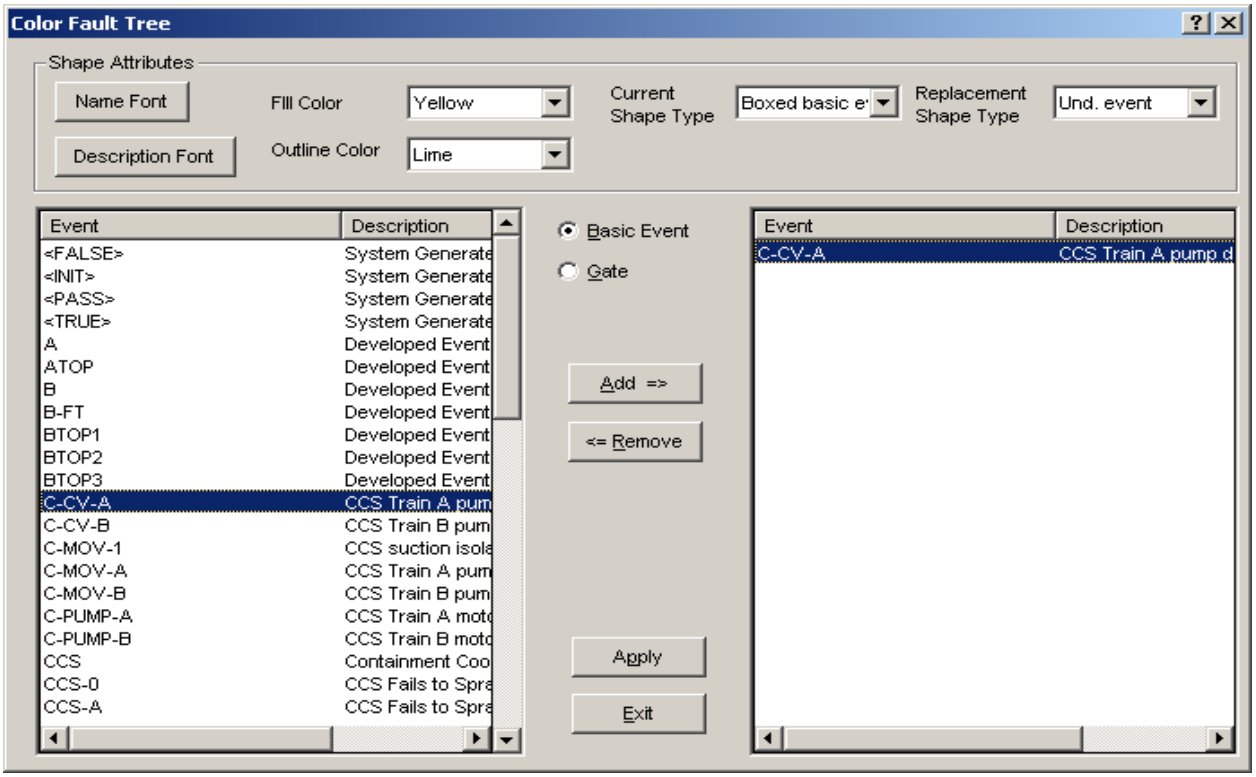
  - As the report shows, gate C-MOV-1-FAILS is used in 6 different fault trees.

  - Also, the gate that might cause problems during event tree sequence cut set generation would be CCS-TRAIN-B, since it is used in multiple fault trees; however, it has different logic associated to it in one of the six fault trees. This gate may need to be modified.

- The next option is extract graphics. This option was not talked about in the SAPHIRE Basics or SAPHIRE Advanced, since all fault trees that are created in the fault tree graphics editor are automatically extracted into the working project folder. This option would allow fault trees that were created using the logic editor to be extracted into the project folder as a graphic file (name.dls).

- The next option is load graphics. This option was talked about in the SAPHIRE Basics course and Section 12.

- The fix color option provides the analyst a mechanism to modify the fault tree basic events and gates colors (both outside line and shading), font, and shape type. The different options are shown in the figure below.

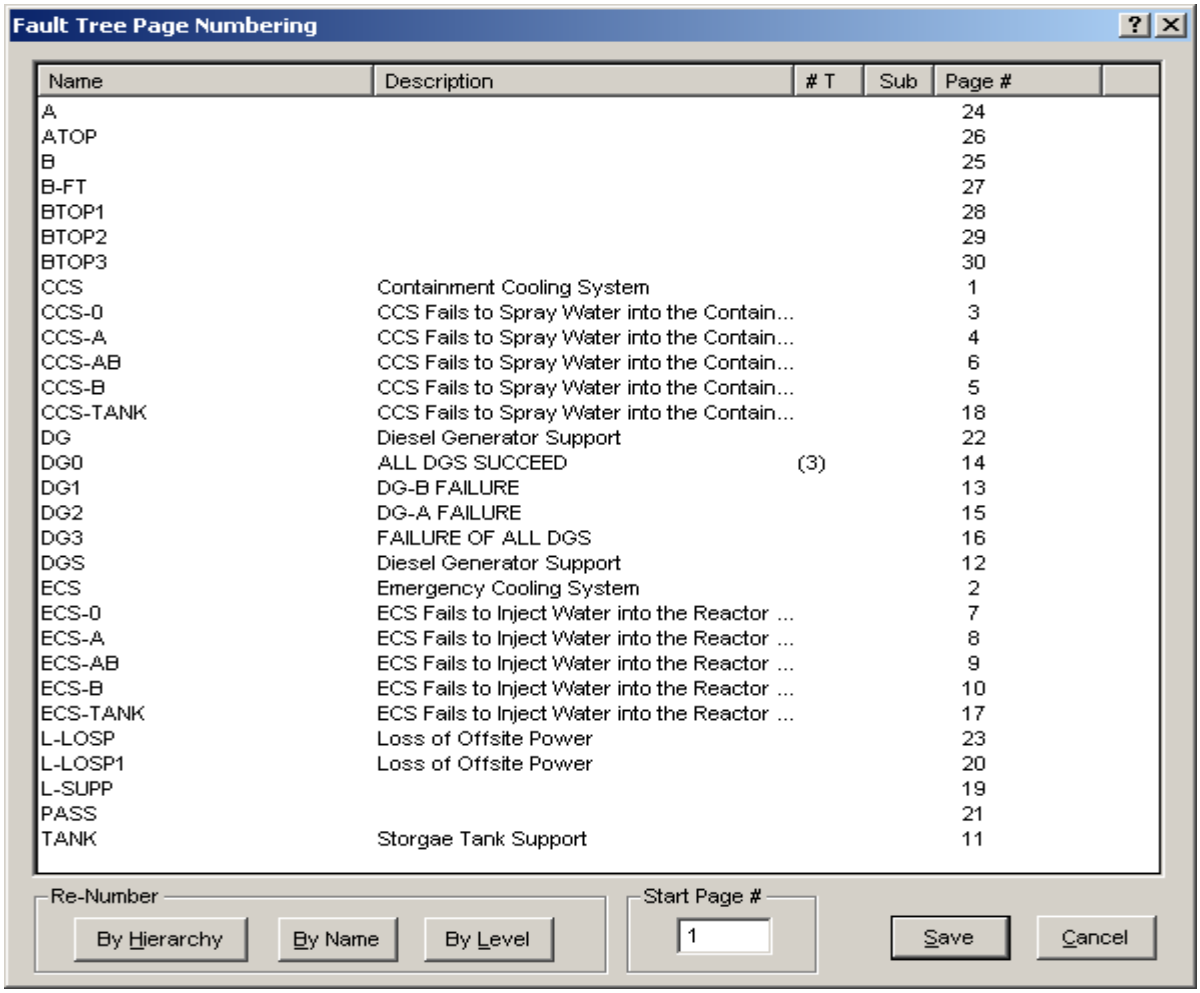


  - ◊ The basic events or gates can be listed depending upon which radio button is selected.

  - ◊ To change the fill color, outline color or any of the other options listed on the top, highlight a basic event or gate then select the **Add** button and this event/gate will show up in the blank screen on the right. Now, what ever option was selected above will be applied to this event/gate(s) by clicking the **Apply** button.

◊ To see the changes go the fault tree(s) that the basic event or gate is located and the graphics will be updated with the change(s).

♦ The number pages option provides the analyst a mechanism to define the page numbering of the fault trees in the project database.



◊ The fault tree page numbering option screen shows the page numbers that are assigned to the fault trees if they were printed with the page option turned on. As the screen shows, fault tree CCS would be printed with page number 1. The default page numbers as the fault trees were built.

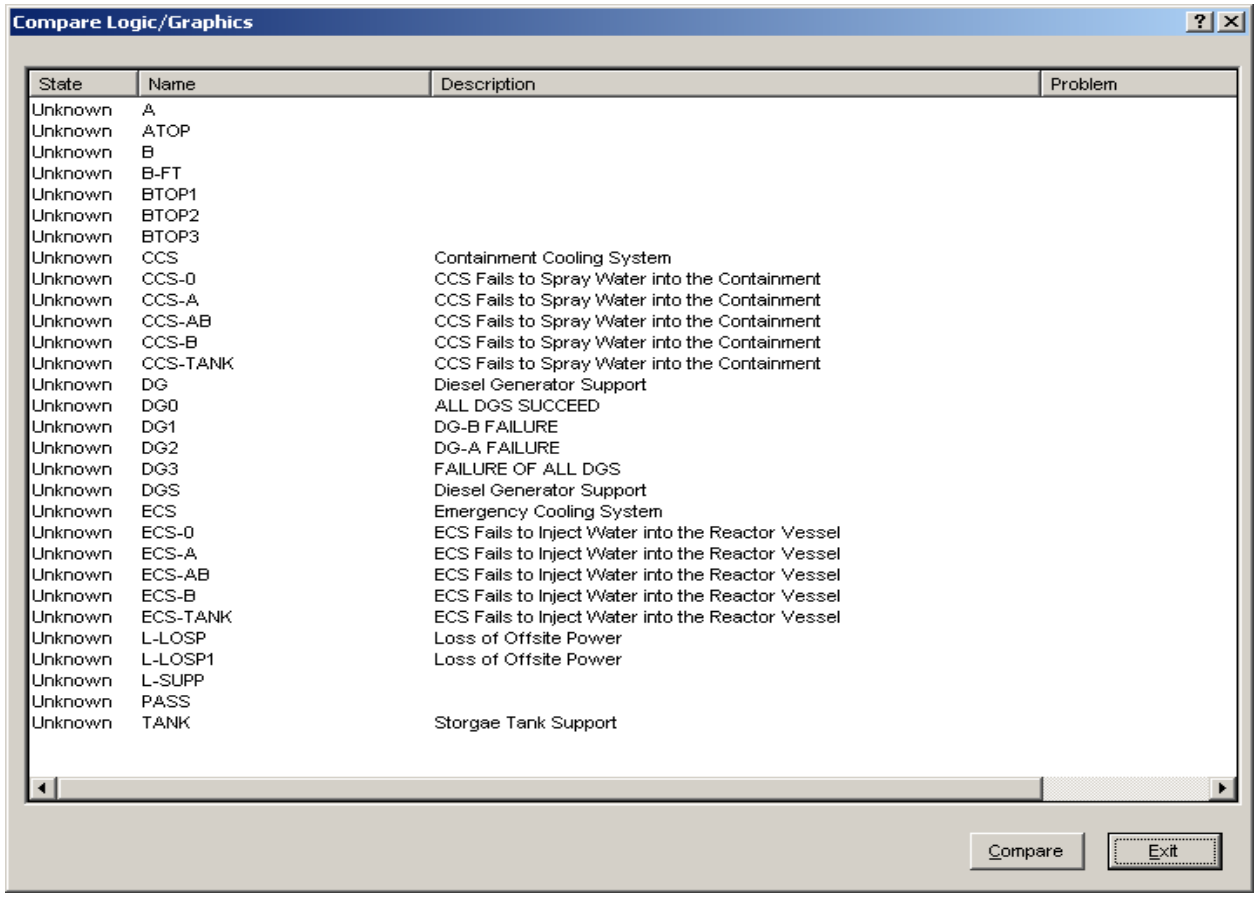
◊ The fields across the top can be selected in order to sort the fault trees by name, description, number of transfers, sub-fault tree, or page number.

◊ The page numbering can be changed from the default by selecting one of the three option buttons in the Re-Number area.

- **By Hierarchy** groups fault trees with their sub-trees. Top level trees appear in relative alphabetic order.
- **By Name** option orders fault trees strictly alphabetically.
- **By Level** option orders top level fault trees first, followed by sub-trees.

◊ The starting page number can be changed in order to provide a cover page or information page prior to the fault tree graphics.

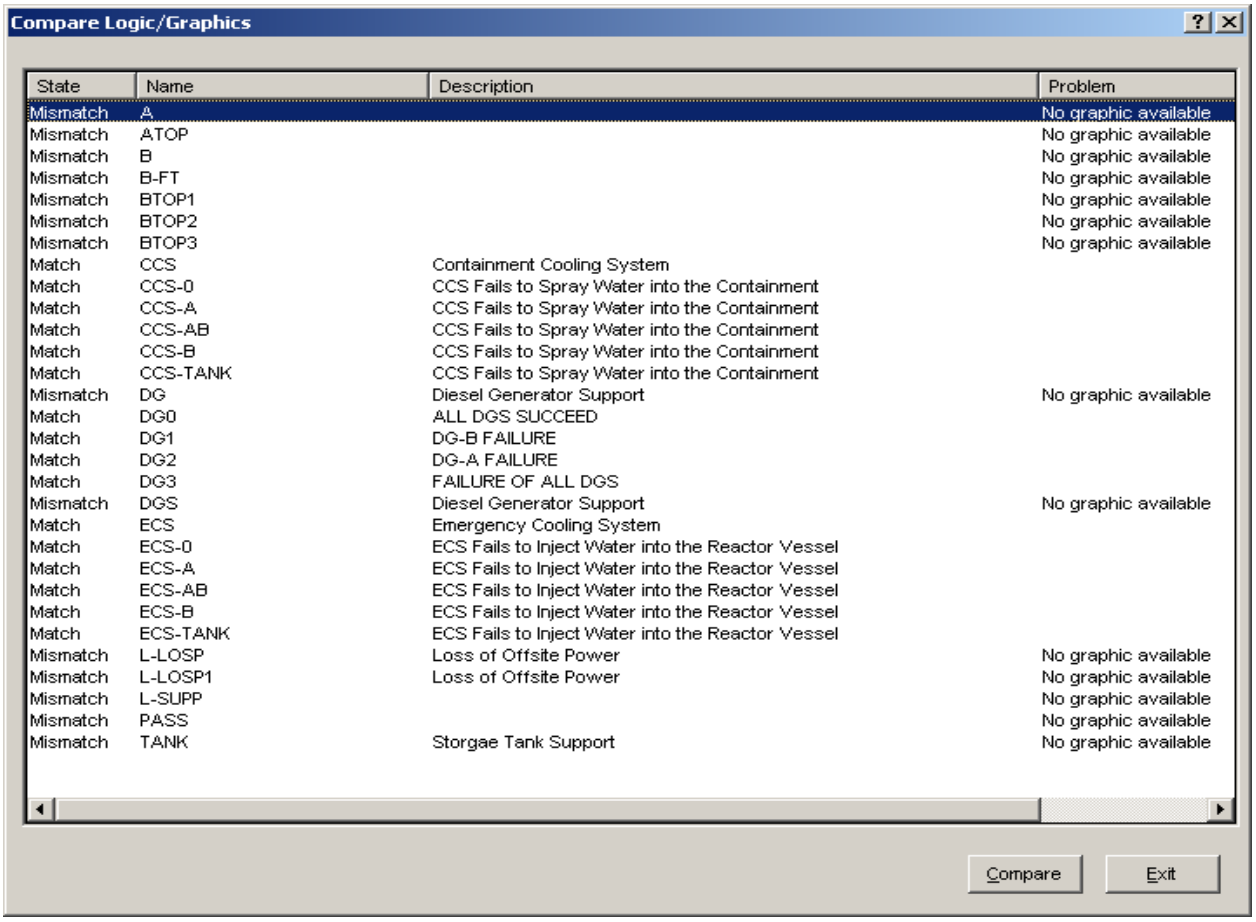
♦ The last fault tree option will check the fault tree logic against its graphic. This will ensure that both the logic and graphic are the same, remember, SAPHIRE uses the fault tree logic when it generates cut sets. The fault tree logic/graphic compare screen is shown below.



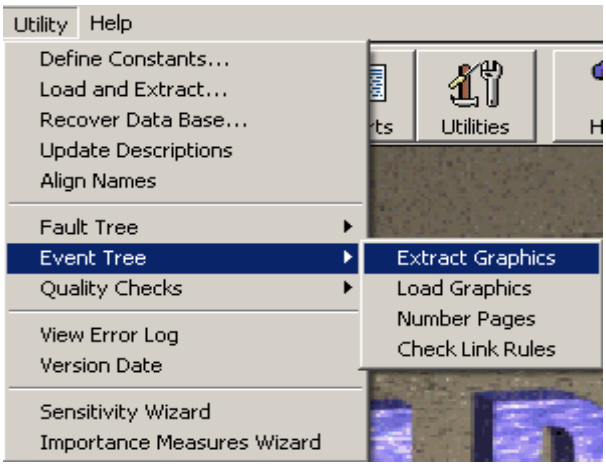
◊ The columns listed in the screen show the state of the comparison, name of the fault tree and then the description. As the screen shows, the logic/graphic comparison has not been completed.

◊ To perform the comparison, highlight the fault tree(s), then click the **Compare** button on the lower right. SAPHIRE will then compare the fault tree logic to its graphic file.

◊ The **Unknown** statement will change based on the results of the comparison. If the fault tree logic matches the graphic file, then **Unknown** will be replaced with **Match** and if the fault tree logic does not match, then **Unknown** is changed to **Mismatch**. The Mismatch fault trees may need to be modified prior to fault tree or event tree sequence cut set generation.



Event Tree Options



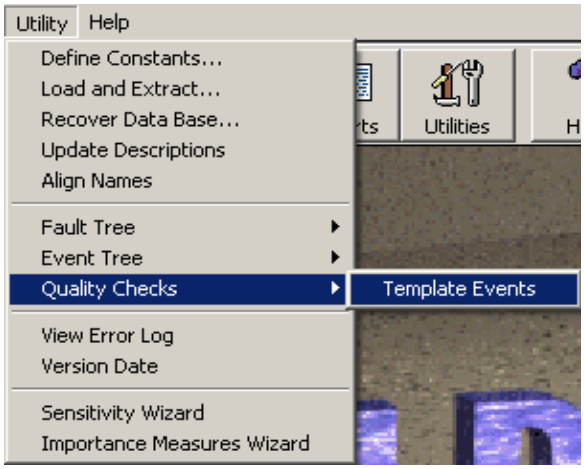
- The first option is extract graphics. This option was not talked about in the SAPHIRE Basics or SAPHIRE Advanced, since all event trees that are created in the event tree graphics editor are automatically extracted into the working project folder (name.etg).

- The next option is load graphics. This option was talked about in the SAPHIRE Basics course and Section 12. This option will load event tree graphics (which includes all descriptions, etc.) into the working project.

- The number pages option provides the analyst a mechanism to define the page numbering of the event trees in the project database. This option is the same as that discussed above for the fault trees.

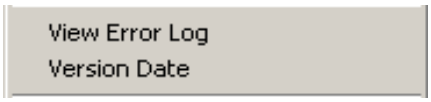
- The check link rules option verifies that all event trees that contain linking rules are compiled and can be applied with generating the accident sequences.

Quality Checks



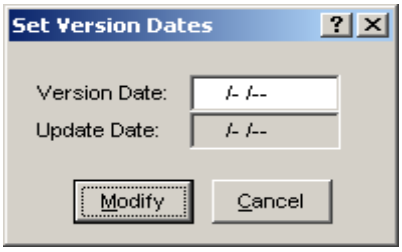
- The quality check option verifies that all basic events are correctly linked to their appropriate template basic event.

The next group of applications that is separated from the others by the dissecting line is View Error Log and Version Date.



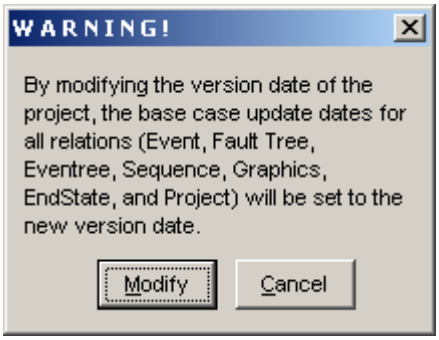
- The View Error Log provides information about the project database. If there are any changes or problems that occurred during the cut set generation, basic event generation, etc., this information will be documented in this report.

- The Version Date provides the completion date for the project database. This field allows the analyst to document when the project was finished.



  - The date field is YYYY/MM/DD.

  - Once the date is placed in the Version Date field, then **Modify** is clicked. SAPHIRE will provide a new pop-up box explaining what is about to be done to the project.



  - By selecting Modify at this point all of the report information will be date stamped with the one specified in the Version Date. Therefore, if someone else picks up the project they will know when it was completed or the date of the last modification.

Appendix A – Link, Recovery, and Partition Rule Keyword List

Keyword or symbol	Type	Definition	Example Usage
( )	General	Symbols to indicate a specific grouping of items.	if (A + B) * (C + D) then The search criteria above would return all top events that contain: [A * C], [A * D], [B * C], or [B * D].
*	General	Symbol to indicate the logical AND command.	if SEARCH-CRITERIA1 * SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match SEARCH-CRITERIA1 and SEARCH-CRITERIA2.
/	General	Symbol used to represent a complemented event (i.e., the success of a failure basic event).	if (/BASIC-EVENT) * "other search criteria" then The search criteria will be satisfied for all cut sets that contain the complement of BASIC-EVENT (and also contains the optional "other search criteria").
;	General	Symbol to indicate the end of a macro line or a line that modifies the cut set being evaluated.	| usage for a macro command MACRO-NAME = "search criteria" ; | usage for a cut set modification line partition = ENDSTATE ;

[ ]	Link	Indicates the number of the event tree branch for multiple-split branch points. The first branch under the top branch is designated as 1. The second is designated as 2, etc.	if "search criteria" then
			/ET-FT = NEW-TREE-NAME1;
			ET-FT[1] = NEW-TREE-NAME2;
			ET-FT[2] = NEW-TREE-NAME3;
			endif

|	General	Symbol used to represent a comment contained in the rules. Everything on a line to the right of this symbol will be ignored by the rule compiler.	| Place your comments here! | Note that blank lines are also permissible!

Keyword or symbol	Type	Definition	Example Usage
~	General	Symbol used in the search criteria to indicate that a particular event will not be in the cut set that is being evaluated.	if (~SEARCH-CRITERIA) * "other search criteria if needed" then ... The search criteria will be satisfied for all cut sets that do not contain SEARCH-CRITERIA (and also contains the optional "other search criteria"). SEARCH-CRITERIA may be an initiating event, basic event, macro, or logic expression.
+	General	Symbol to indicate the logical OR command.	if SEARCH-CRITERIA1 + SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match either SEARCH-CRITERIA1 or SEARCH-CRITERIA2.

=	General	Keyword to indicate the substitution of one event tree top (i.e., fault tree) for another event.	if "search criteria" then
			ET-FT = ET-FT1;
			endif

AddEvent =	Recovery	Keyword that indicates that an event will be added to the cut set being evaluated.	if "search criteria" then
			AddEvent = EVENT-NAME;
			endif

always	General	Keyword that indicates that every fault tree top event satisfies the search criteria.	if always then
			perform some action on the sequence.;
			endif

CopyCutset;	Recovery	Keyword that indicates that the cut set being evaluated will be copied and added to the list of cut sets. This copied cut set then becomes the cut set that is being evaluated.	if "search criteria" then
			CopyCutset;
			now make modification to a copy of the cut set...
			endif

CurrentPart( )	Partition	Keyword that searches for cut sets that have already been assigned to the endstate indicated.	if CurrentPart(CORE-DAMAGE) then
			partition = “NEW-CORE-DAMAGE”;
			endif

DeleteEvent=	Recovery	Keyword that indicates that an event will be deleted from the cut set being evaluated.	if "search criteria" then
			DeleteEvent = EVENT-NAME;
			endif

Keyword or symbol	Type	Definition	Example Usage

DeleteRoot;	Recovery	Keyword that indicates that the original cut set (i.e., that cut set that satisfied the search criteria) will be deleted.	if "search criteria" then
			DeleteRoot;
			endif

else	General	Keyword that specifies some action to be taken if all the search criteria(s) are not met. The else should be the last condition in the event tree linking rule.	if "search criteria" then
			perform some action on the sequence;
			else
			perform some other action on the sequence
			if search criteria not met;
			endif

elsif	General	Keyword that specifies an alternative search criteria. Any number of elsifs can be used within an event tree linking rule.	if "search criteria" then
			perform some action on the sequence.;
			elsif "2nd search criteria" then
			perform some other action on the sequence;
			elsif "3rd search criteria" then
			perform some other action on the sequence;
			endif

endif	General	Keyword that indicates the end of a particular rule.	if "search criteria" then
			perform some action on the sequence.;
			endif

endstate	Link	Keyword to assign a sequence (based upon sequence logic) to a particular end state.	If “search criteria” then
			eventree(ET-NAME) = endstate(ES-NAME);
			endif

eventree( )	Link	Keyword to indicate a change in the sequence transfer name.	if "search criteria" then
			eventree(ORIG-TRAN) = eventree(NEW-TRAN);
			endif

False( )	Link	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to FALSE for the applicable sequence. Multiple basic events should be separated using commas.	if “search criteria” then
			eventree(ET-NAME) = False (EVENT1, EVENT2, EVENT3, …);
			endif

Keyword or symbol	Type	Definition	Example Usage

Flag( )	Link	Keyword to assign an existing Flag Set to sequences meeting the search criteria.	if “search criteria” then
			eventree(ET-NAME) = Flag (FS-NAME);
			endif

GlobalPartition=	Partition	Keyword to indicate that all cut sets in a particular sequence will be assigned to the end state identified after the equal sign.	if "search criteria" then
			GlobalPartition = “MY-END-STATE”;
			endif

if then	General	Keyword that indicates search criteria is being specified.	if "search criteria" then
			perform some action on the sequence.;
			endif

Ignore( )	Link	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to IGNORE for the applicable sequence. Multiple basic events should be separated using commas.	if “search criteria” then
			eventree(ET-NAME) = Ignore (EVENT1,
			EVENT2, EVENT3, …);
			endif

init( )	General	Keyword used in the search criteria to indicate that a sequence cut set has a particular initiating event.	if init(INITIATOR-NAME) * "other search criteria if needed" then
			perform some action on each cut set;
			endif

MACRO	General	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 24 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
			if MACRO-NAME then
			perform some action on each sequence.;
			endif
			
			|Macros are only applicable in the particular
			|rule set where they appear. In other words,
			|you can not define a macro in event tree
			|”A” and expect to use it in event tree “B.”

Keyword or symbol	Type	Definition	Example Usage

NewCutset;	Recovery	Keyword that indicates that a new, empty cut set will be added to the list of cut sets. This new cut set then becomes the cut set that is being evaluated.	if "search criteria" then
			NewCutset;
			now make additions to the empty cut set...
			endif

partition =	Partition	Keyword that indicates the end state characters for the cut sets meeting the search criteria will be modified according to the text after the equal sign.	if "search criteria" then
			partition = “END_STATE_NAME”;
			endif

Recovery =	Recovery	Keyword that indicates that a recovery event is going to be added to the cut set being evaluated (SAPHIRE keeps record of all recovery events).	if "search criteria" then
			recovery = NAME-OF-RECOVERY;
			endif

Skip( )	Link	Keyword to indicate that a sequence meeting the search criteria will be “skipped” (i.e., not generated and will not show up in the database).	if "search criteria" then
			ET-FT = Skip(ET-FT);
			endif

system( )	General	Keyword used in the search criteria to indicate that the sequence logic contains the particular top event. Can be used in either recovery rules or partition rules.	if system(TOP EVENT) * “other search criteria if needed” then
			perform action on each sequence;
			endif

Keyword or symbol	Type	Definition	Example Usage

transfer =	Partition	Keyword to indicate the event tree to be created and transferred to for the sequence meeting the search criteria. The sequence end state frequency will be used as the initiating event frequency for the new event tree.	if "search criteria" then
			GlobalPartition = “CORE-DAMAGE”;
			transfer = LEVEL-2-TREE;
			endif

True( )	Link	Keyword to construct a Flag Set where the identified basic events (in parenthesis) are set to TRUE for the applicable sequence. Multiple basic events should be separated using commas.	if “search criteria” then
			eventree(ET-NAME) = True (EVENT1,
			EVENT2, EVENT3, …);
			Endif

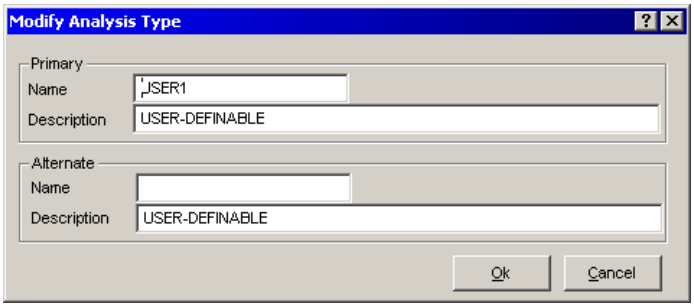
keep	Slice	Keyword to group the cut sets that meet the search criteria together for display in the “Included In Slice”.	If “search criteria” then
			keep;
			endif

discard	Slice	Keyword to group the cut sets that meet the search criteria together for display in the “Excluded From Slice”.	If “search criteria” then
			discard;
			endif

Appendix B – Using External Events in SAPHIRE

Appendix B provides an introduction to the external event features of SAPHIRE. The purpose of this Appendix is not to inform the user on external event methodologies but, to introduce the user to the external event capabilities in SAPHIRE. The discussion assumes the availability of an "internal-events" PRA. Specifically, random-failure composed system-models, accident sequence progression, and initiating events have all been defined and developed for the engineered system of interest (e.g., nuclear power plant). The external event analysis is being factored into that engineered system, which is already well understood and comprehensively modeled. Therefore, functional vulnerabilities have been identified and the seismic, fire, or flood analysis consists of converting to and adding in the externally-induced failures.

SAPHIRE is designed to perform various types of external analysis including seismic, fire and flood. Other types of external analysis can be defined by modifying the eight user-defined analysis types by selecting **Modify → Analysis Types** for the drop down menu.



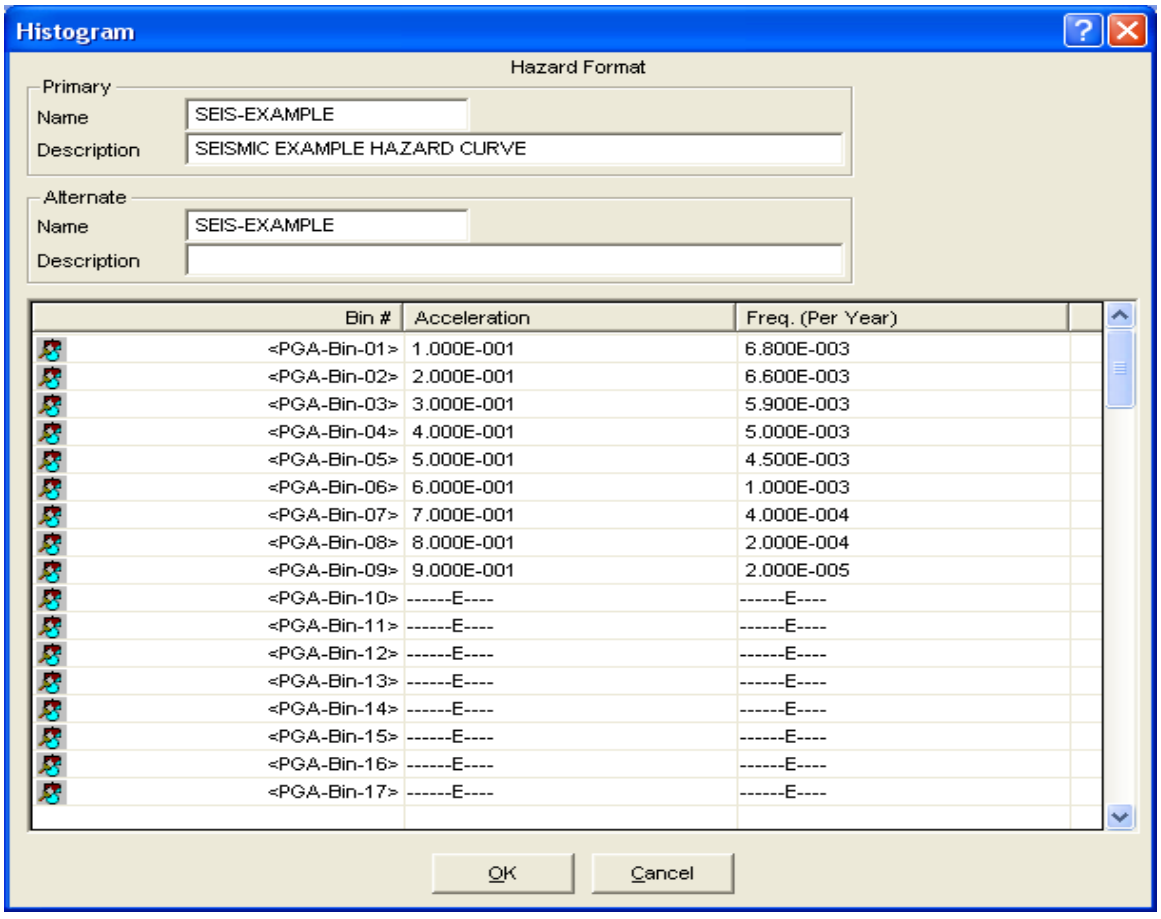
B.1 Seismic Analysis

SAPHIRE provides the flexibility to construct seismic risk analysis models by either (or a combination) of two methods. First, seismic-specific event tree and fault tree models can be developed via the graphical interface. Second, SAPHIRE contains a provision for performing transformations in the form of Boolean identities (i.e., A=A+B, A=B, or A=A*B). This allows the user to build on an internal events analysis when developing a seismic model. More specifically, after site-specific seismic vulnerabilities have been identified (through plant walk-downs or some other site-specific review), they can be incorporated into an existing internal events analysis using a set of basic-event transformations that either replace or add the seismic failure events to the existing basic events.

Hazard Curve

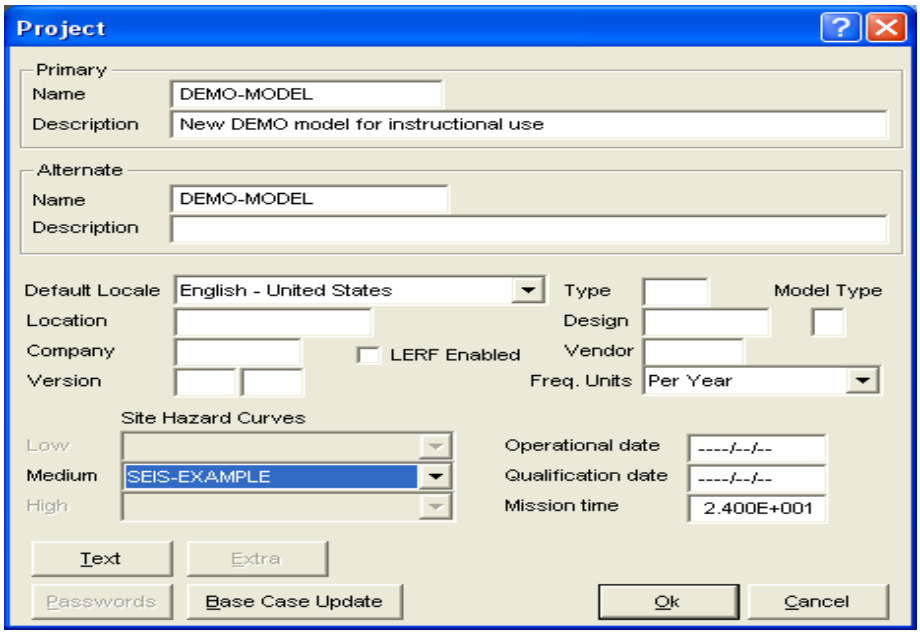
The hazard curve is the representation of the range of possible earthquakes. It is commonly found in the form of a probability of exceedence curve, with the earthquake ground acceleration on the horizontal and the probability of exceeding that acceleration on the vertical axes. (One source for this information is NUREG-1488.) However, SAPHIRE utilizes this information in the form of a histogram (or more precisely, a discreet probability density distribution). Specifically, the density needs to be arraigned into a maximum of 100 ground acceleration bins with each one assigned a yearly frequency of occurrence.

To input a histrogram into SAPHIRE, select **Modify → Histograms**. This brings up the "Edit Histograms" dialog box. <Right Click> the mouse button to **Add** a new histogram or **Modify** an existing histogram.



Each bin (numbered 1-100) is associated with an event name (e.g., PGA-BIN-01), which is how that specific earthquake event (magnitude and frequency) is identified in the analysis.

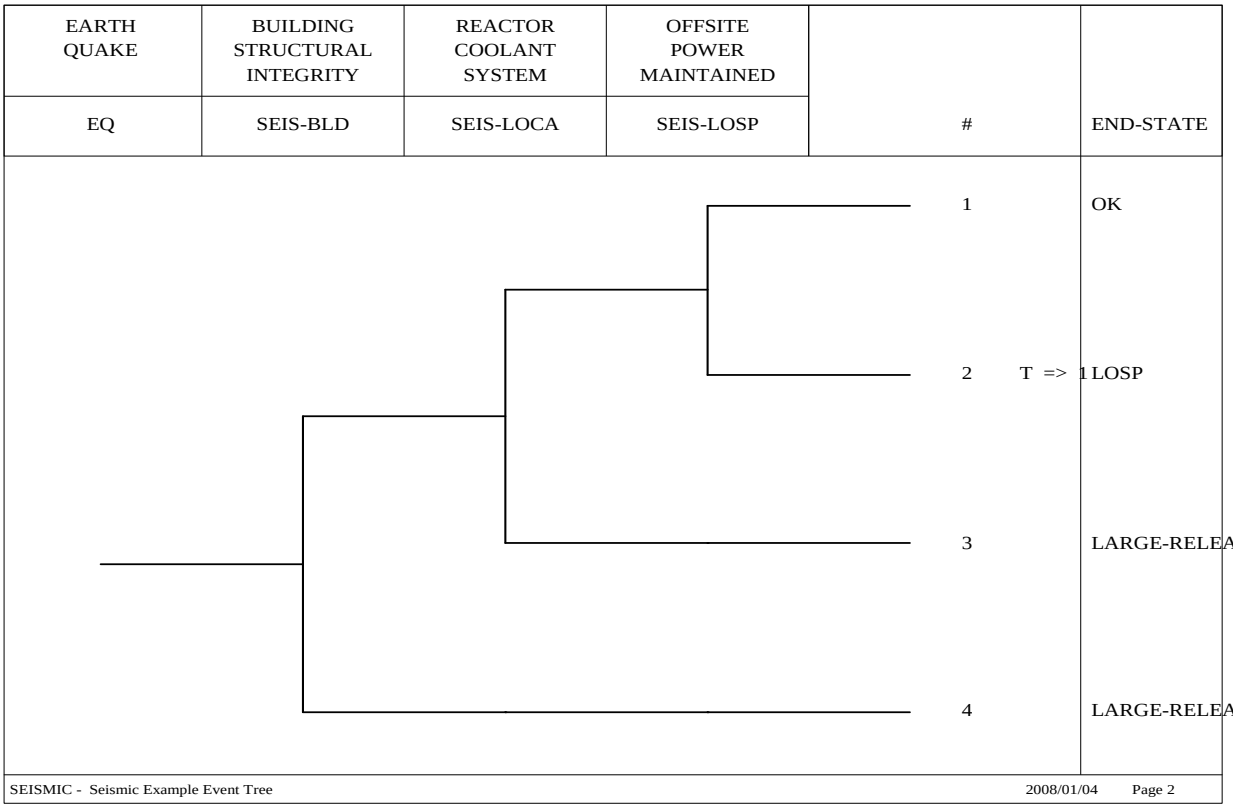
Several histograms may be created for a particular project. For SAPHIRE to know which will be used in the analysis, it must be identified by selecting the **Modify → Project** Only the histogram listed in the "*Medium*" field, under the heading "Site Hazard Curves," is actually used in SAPHIRE 7.0. The "High" and "Low" fields are not used at this time and are reserved for future development.



Uncertainty information for the earthquake data is entered directly into the basic event dialogs [i.e., from the histogram dialog, select the bin of interest and selecting **Modify → Basic Events** and then the event name (histogram bin name) of interest.

Seismic Event Tree

The most straightforward approach (at least with respect to using SAPHIRE) for creating a seismic analysis model involves the development of a seismic event tree that prioritizes and links the seismic-induced internal events initiators with the earthquake (the true initiating event). This single seismic event tree begins with a generic seismic-initiating event set to a value of 1.0. [The actual magnitude (g- level) and frequency of the earthquake of interest are identified by the user and factored into the analysis when the cut sets are generated and quantified.] The event tree top-events are those internal events initiators that have the potential to be induced by an earthquake. They are listed in order of severity (in terms of challenging plant safety systems), with the more severe induced-initiators listed first. This addresses the potential pitfall of over-counting core damage sequences (i.e., a single earthquake inducing both a large LOCA and a small LOCA at the same time). The event tree shown below is a simple illustration of the development for the DEMO project.



These event tree top- events are treated as seismic basic- events (or fault trees) with associated seismic fragility data. The resulting end states are therefore the frequencies of seismically induced challenges (i.e., internal- events type initiators) to the plant. These in turn can be identified as transfers to the systems analysis (internal-events accident sequences) event trees. This linking will automatically replace the internal events analysis initiator on the systems analysis event tree with the transferred information from the seismic event tree. This is how the linking between the earthquake, induced initiating event, and the system models are made.

Seismic Fault Trees

The actual system models are commonly fault trees and were used during the internal event analysis. Using SAPHIRE, they can also be integrated with the seismic event analysis.

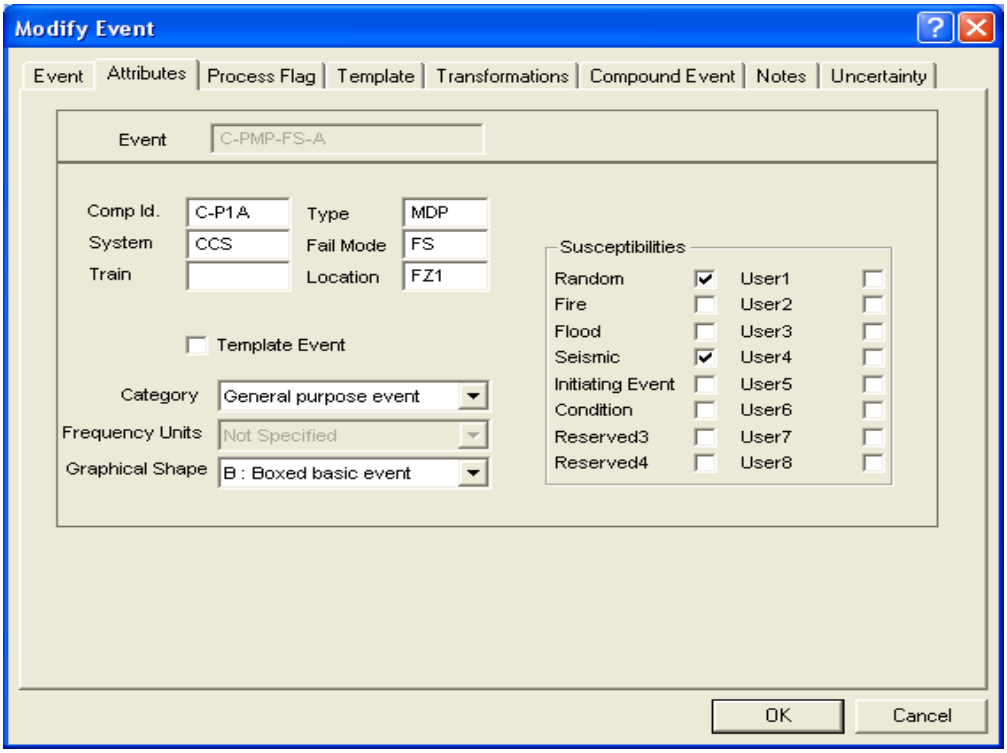
Note: The seismic event trees and seismic fault trees can be created as independent, and stand-alone. However, with SAPHIRE, by utilizing the “Seismic Analysis Type”, they can also be integrated with the internal events analysis.

Along with linking (i.e., establishing transfers) between the seismic event tree and the accident sequence event trees, the system models supporting development of the accident sequence event tree top-events need to be modified to include seismic-induced failures. This may be done by defining "transformations" of the random failures modeled in the internal events analysis into seismic failures. The transformations can be performed such that the original event is kept in the model and the seismic event is simply added (internally in SAPHIRE) to the fault tree. This allows the user the option of incorporating random failures in the seismic analysis.

Seismic Basic Events

The transformations are created in the **Modify –-> Basic Event** menu selection. However, before creating a transformation, the seismic basic events must be added to the SAPHIRE database in order to be incorporated into the fault tree. For example, if a motor driven pump, C-PUMP-FS-A is susceptible to seismic induced failures, a second basic event must be created in SAPHIRE **(using Modify → Basic Events → <right click the mouse button> → Add)**. This "seismic event" could be called "SEIS-C-PUMP-A".

Once this event is added to the data base, it must be identified as being susceptible to seismic initiators. This is accomplished through **Modify → Basic Events** and selecting the "Attributes" tab.

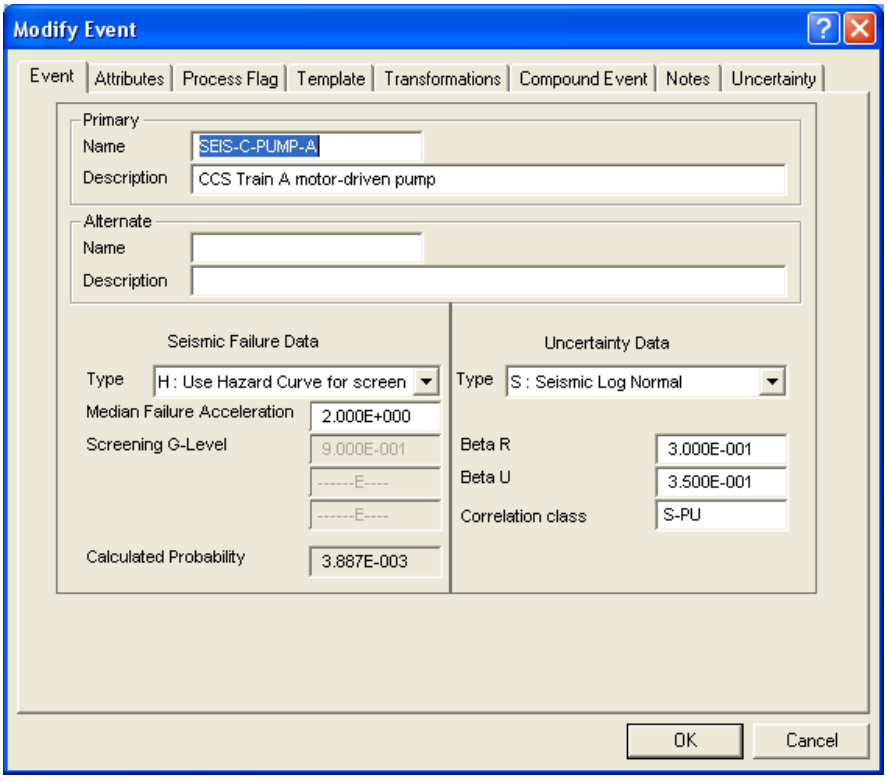


Click on

the “Seismic” box under Susceptibilities. This informs SAPHIRE that this event is susceptible to seismic initiators. Now, SAPHIRE will automatically look for transformations whenever a seismic analysis is performed.

Seismic failure data is usually characterized by a median fragility and two uncertainty terms representing the random uncertainty and the confidence uncertainty (Beta- R and Beta- U, respectively). There is also an added factor that might or might not be included in seismic failure data called the structural response factor (SRF). The SRF quantifies the amount of amplification or dampening of ground motion a particular piece of equipment experiences during an earthquake, by virtue of its location. For example, during a postulated earthquake, a relay on the fourth floor of a building would likely experience a different magnitude of shaking compared to a relay on the first floor. The SRF accounts for this difference. SAPHIRE does not maintain the SRF information separately. Before entering the seismic fragility data, the SRF needs to be factored in, and then the SRF- adjusted fragility data is entered into the database.

To enter seismic data into a seismic basic event record, select **Modify → Basic Events** and go to the "Random Failure Data" and select the “**Type**” drop down box. Selecting "G" or "H" defines the basic event as a seismic basic event. The "G" and "H" simply identify the basis for the assumed magnitude of the peak ground acceleration (PGA) or g-level, for initially generating cut sets.



If the "G" Type option is selected, the user will need to specify a particular g level to be used (with the fragility curve) to calculate a point-estimate probability for the cut set generation process. The "H" Type option instructs SAPHIRE to utilize the highest g level found on the user-specified hazard curve.

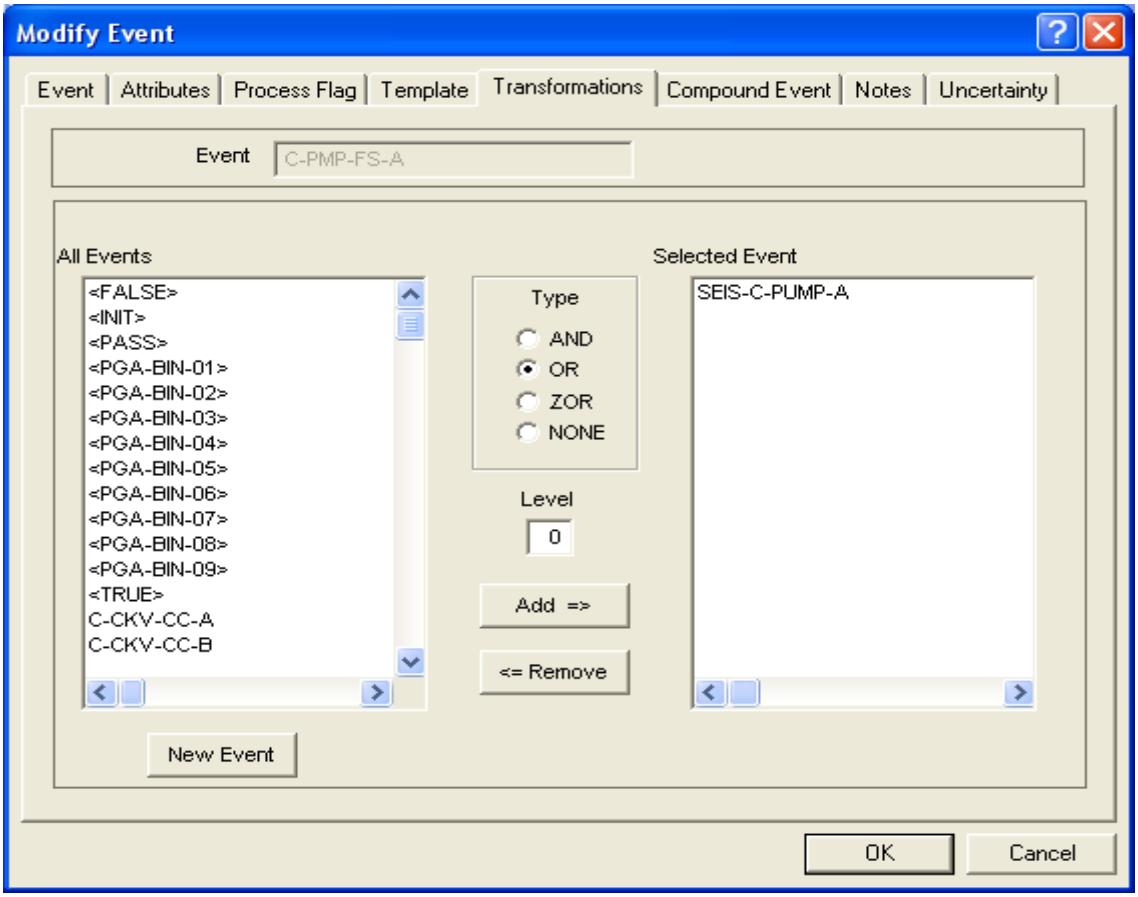
The Seismic “Beta R” and “Beta U” values and inputted into SAPHIRE by selecting the “Uncertainty Data” **Type** drop down box and selecting “S: Seismic Log Normal”.

Transformations

A transformation is a replacement or addition of basic events inside the fault tree logic. During a seismic transformation, a seismic event (or events) is added to the fault tree logic. Given a basic event representing a seismic susceptible component (For example C-PUMP-FS-A), it’s seismic basic event, SEIS-C-PUMP-A, that represents the component’s fragility or “robustness” during a specific g-level earthquake must be added to the fault tree logic.

To perform a seismic transformation in SAPHIRE select **Modify → Basic Event** and select the seismic susceptible random basic event (For example C-PUMP-FS-A).

Select the **Transformations** tab.



There are three “Types” of Transformations:

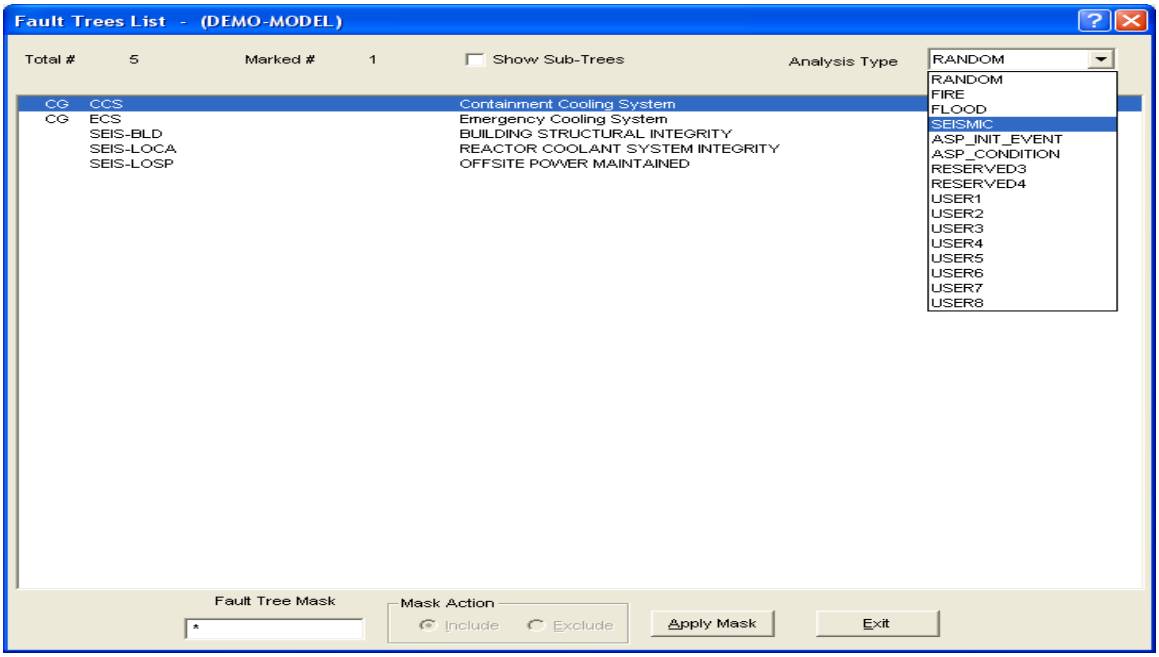
AND =	For this type of transformation, all included events must fail. The event is replaced with an AND gate, with all marked events as inputs.
OR =	For this type of transformation, any included events must fail. The event is replaced with an OR gate, with all marked events as inputs.
ZOR =	Event make up a Zone. If any events fail in the list fail, all event fail.

For the seismic analysis, the seismic event must be added to the fault tree logic so select the “OR” radio button. Highlight the seismic event (For example, S-C-PUMP-A) and select the Add => button to move the event to the “Selected Event” list. Select the “OK” button to complete the transformation. This step must be completed for all seismic susceptible events.

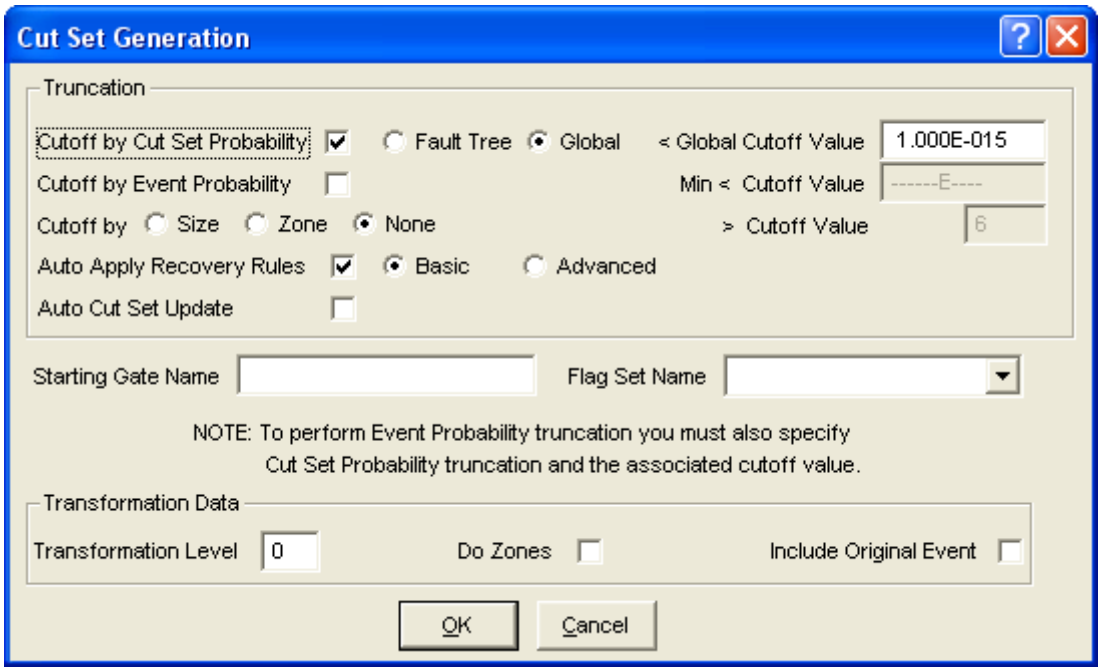
Generating Seismic Cut Sets

Seismic cut set quantification is performed similar to a regular, internal events (i.e., "Analysis Type = random") quantification, with a couple of minor differences.

To generate seismic cut sets, the "Analysis Type" needs to be set to seismic for fault tree and/or sequence cut sets. This tells SAPHIRE to use the seismic cut sets and factor in information such as the hazard curve and any transformations. To generate fault tree cuts, select **Fault Trees** and change the “Analysis Type” to **Seismic**.



Perform the standard steps to generate cut sets (highlight the fault tree, right click and select Solve). To include the “random” basic event failure along with the seismic basic event, mark the “Include Original Event” under “Transformation Data”



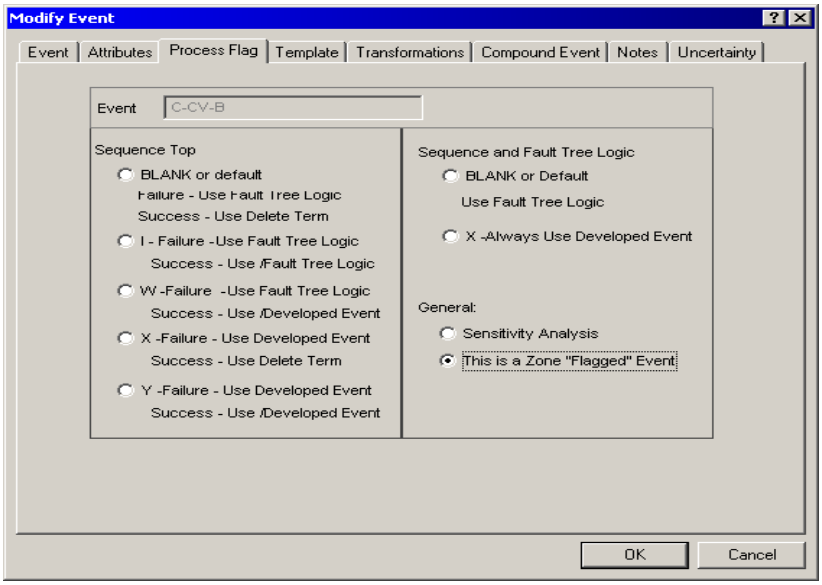
Second, after selecting "Quantification," the user is prompted to choose the "G-Level" for which the quantification is to be performed. The options available include: each g-level bin that contains non-zero data for the hazard histogram identified for use with the current Family, all bins together, and all bins separately. Once the g- level is selected, the quantification proceeds and results are calculated.

An important feature to keep in mind is that only a single cut set list is maintained for each system, sequence, or end state in SAPHIRE. This limit also applies to seismic calculation, which is where the effect of this can cause some confusion. Specifically, when performing a seismic quantification, the cut set lists for each g- level are not maintained. Hence, the user is limited when viewing and reporting quantified cut sets; only the last quantification performed (i.e., that specific g-level) will be available. Numerical results, however, are stored and available for each individual g-level.

B.2 Fire/Flood Analysis

SAPHIRE is capable of performing fire and flood analysis using the internal event analysis fault trees and event trees contained in its data base and executing event transformation that provide the logic to evaluate these events.

The first step in performing a fire or flood analysis is to map out the fire/flood areas and compartments for the associated system(s). After mapping, locations and zones can be assigned using the transformation features of SAPHIRE. Therefore, if a fire occurs in a particular location or zone, all components in that area would be failed (or if a fire in a particular area effects components in other zones. i.e. cable trays). This type of analysis is performed by using the ZOR transformation type and using the "Zone Flagged Event" under the Process Flag Tab under the category "General:".



Once the location transformation are defined, the SAPHIRE software will handle all the necessary analysis details when performing special analysis types such as fire and flooding evaluations.

For more details of Fire/Flood analysis, see the SAPHIRE Technical Reference Guide located under the Help menu option.