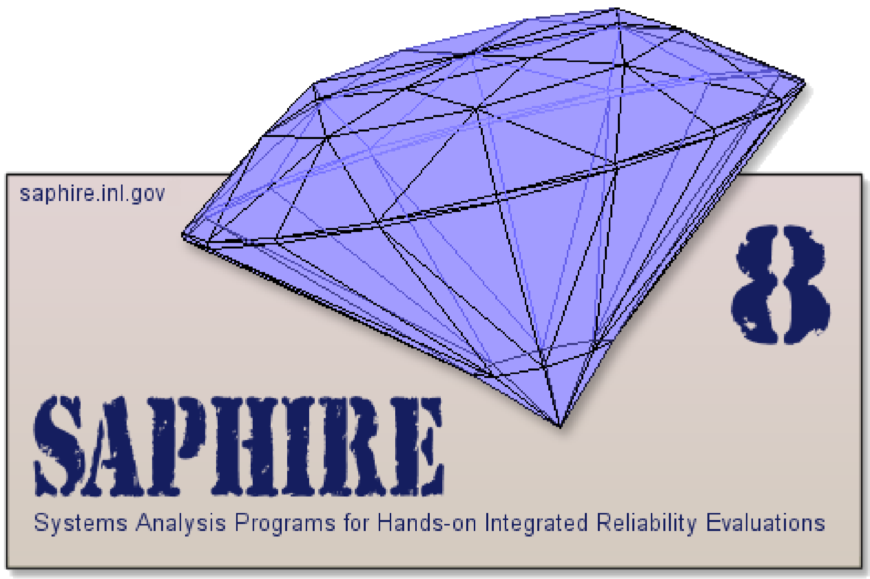
Advanced SAPHIRE 8

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

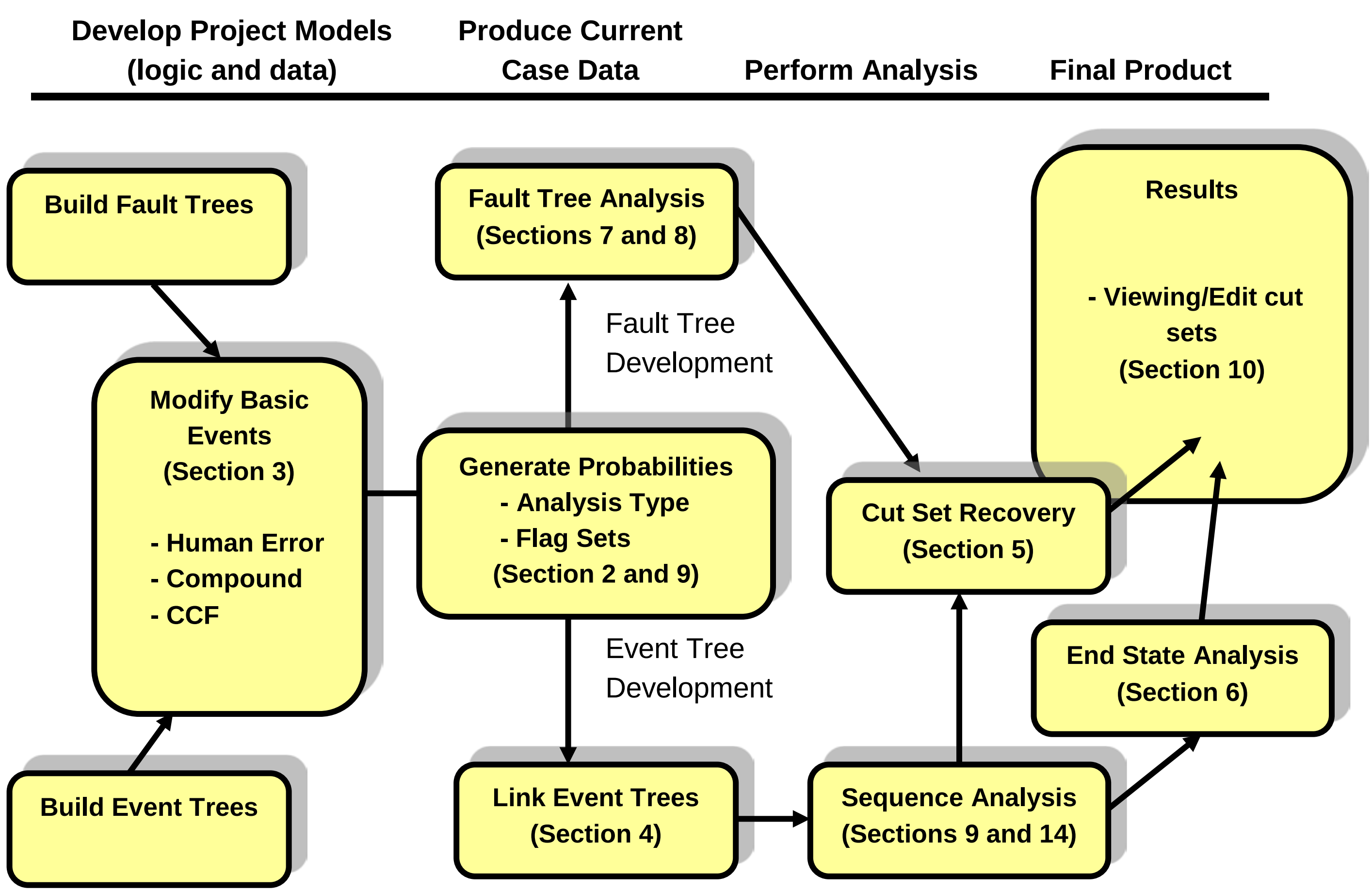


Curtis Smith
James Knudsen
Kurt Vedros
Ted Wood

Idaho National Laboratory

February 2016

SAPHIRE – The "Big Picture"



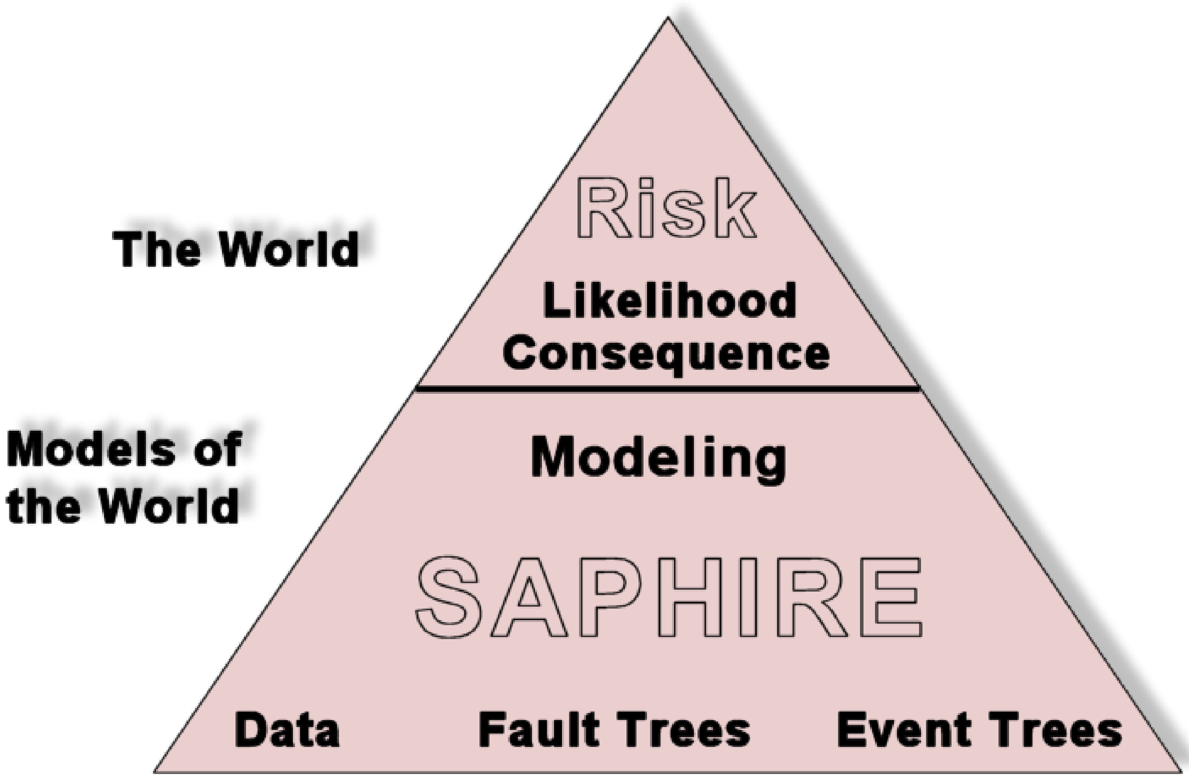
NOTICE

CONTENTS

This page intentionally left blank

| 1 | INTRODUCTION

Section 1 contains an introduction to the SAPHIRE 8 Advanced course material and an overview of the SAPHIRE software.



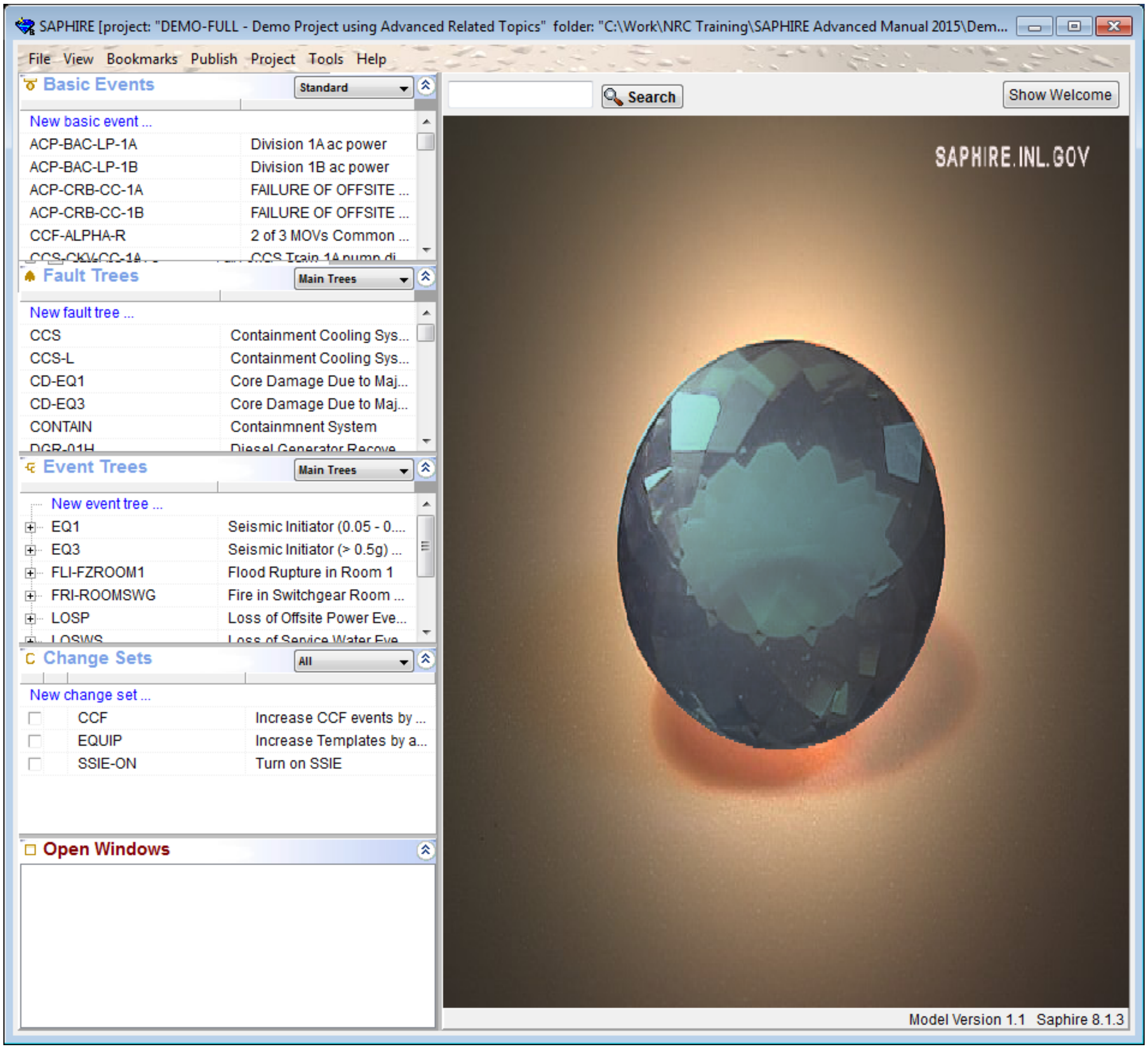
1.1. Overview of the Advanced SAPHIRE Material

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 Model 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 tree logic 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 transfer of data between SAPHIRE data bases and between other PRA codes (MAR-D).

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



The left side of the screen displays a resizable, customizable series of lists containing important PRA elements found in the current project. This series of lists is referred to as the List Panel and can be added to or subtracted from by using View on the main menu.

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 a user the ability to create and analyze fault trees and event trees using a personal computer.

- ♦ Integrated Reliability and Risk Analysis System (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.

- ♦ Creation of 32-bit IRRAS, version 5.0, in 1992 resulted in an order of magnitude decrease in analysis time. New features included: individual codes 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.

- ♦ SAPHIRE for Windows, version 8.0, is released in 2010. The user interface has been completely rewritten. This manual is written for version 8.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 2000 or greater
    - ◊ 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

- ♦ Perform the following steps to install SAPHIRE:

- ♦ Download the current version of Saphire8XXXX.exe from the SAPHIRE website to your pc. Alternatively, use an installation CD and place it into the CD drive.

- ♦ 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:\Saphire8\DEMO).

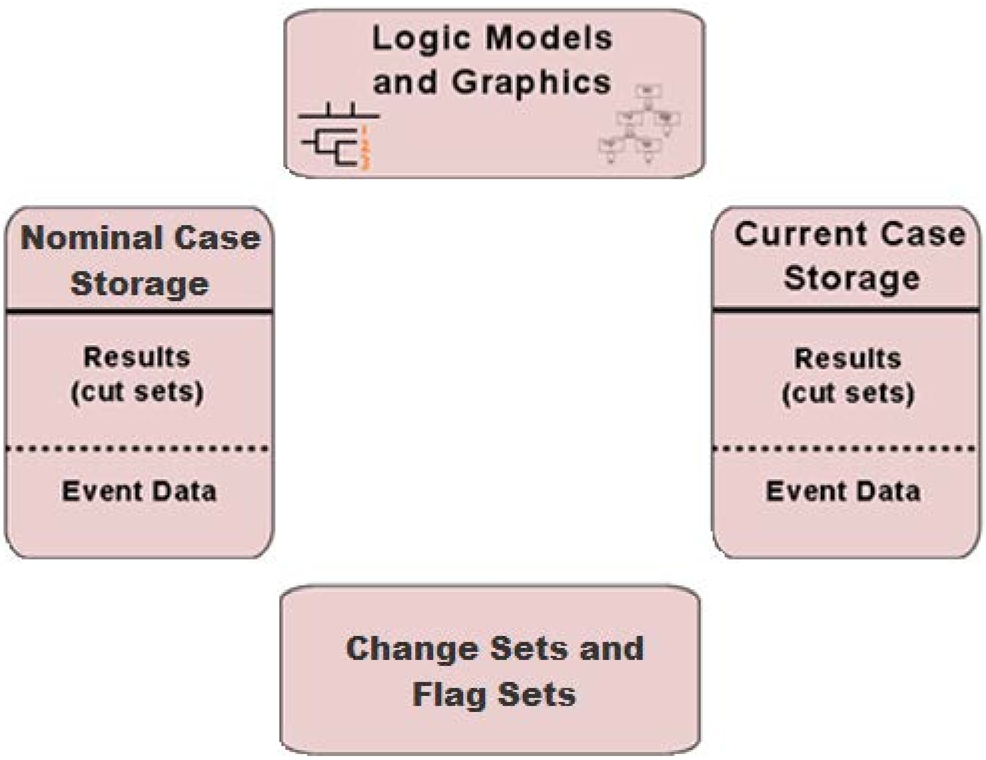
    - ◊ The database subdirectory will contain the relational database files.

        - *.IDX files contain data indices
        - *.BLK files contain variable length data (e.g., cut sets)
        - *.DAT files contain actual data and data pointers.

NOTES

| 2 | DATABASE CONCEPTS

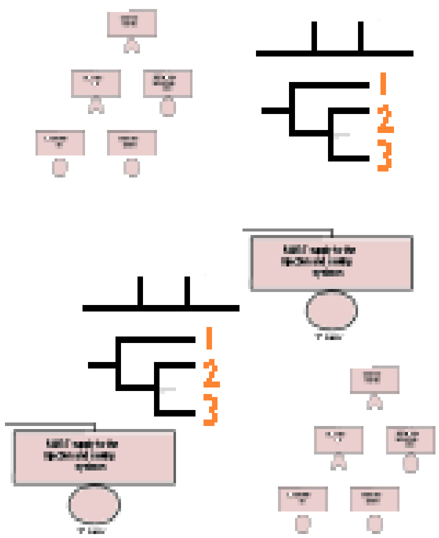
Section 2 presents an overview of the SAPHIRE database structure. Included in this section are discussions of SAPHIRE projects, nominal case versus current case, nominal case updates, flag sets, and change sets. Advanced data base features that are discussed include: (1) Model Types and (2) Flag Sets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



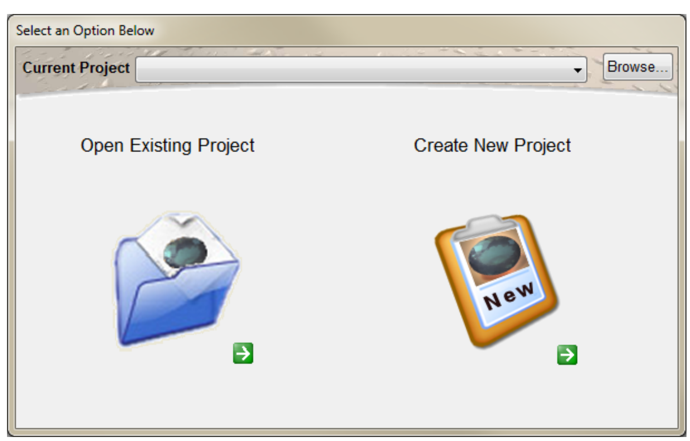
= SAPHIRE Project

Opening a Project

(When SAPHIRE is executed, it loads up the last project that was being utilized.)

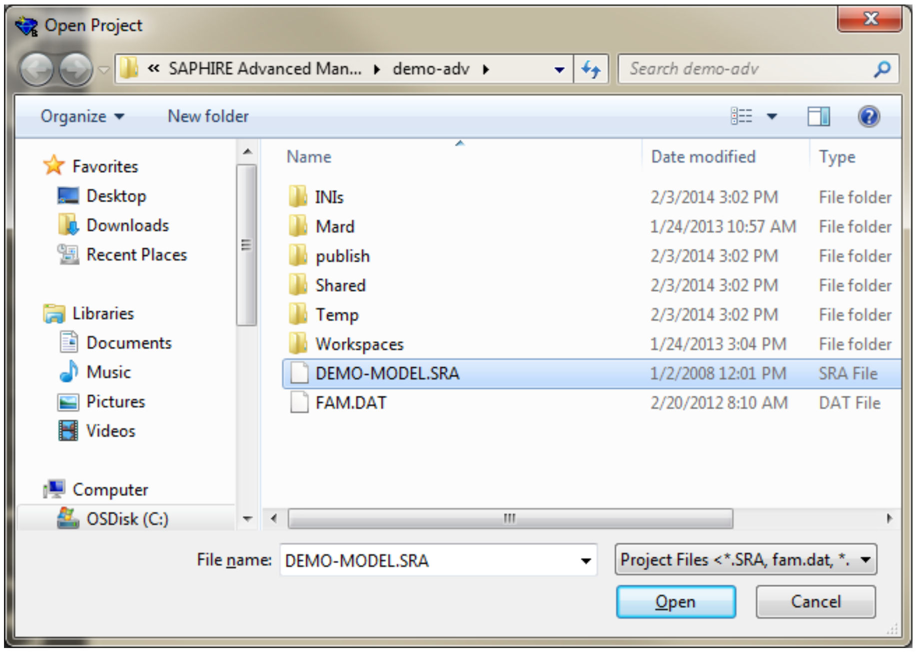
To select an existing project:

If there is not a current project available or you are opening a second session of SAPHIRE on your computer, it will start with an option screen to open an existing project or create a new one.



Existing projects can be found through the browsing function or by clicking the Open Existing Project icon.

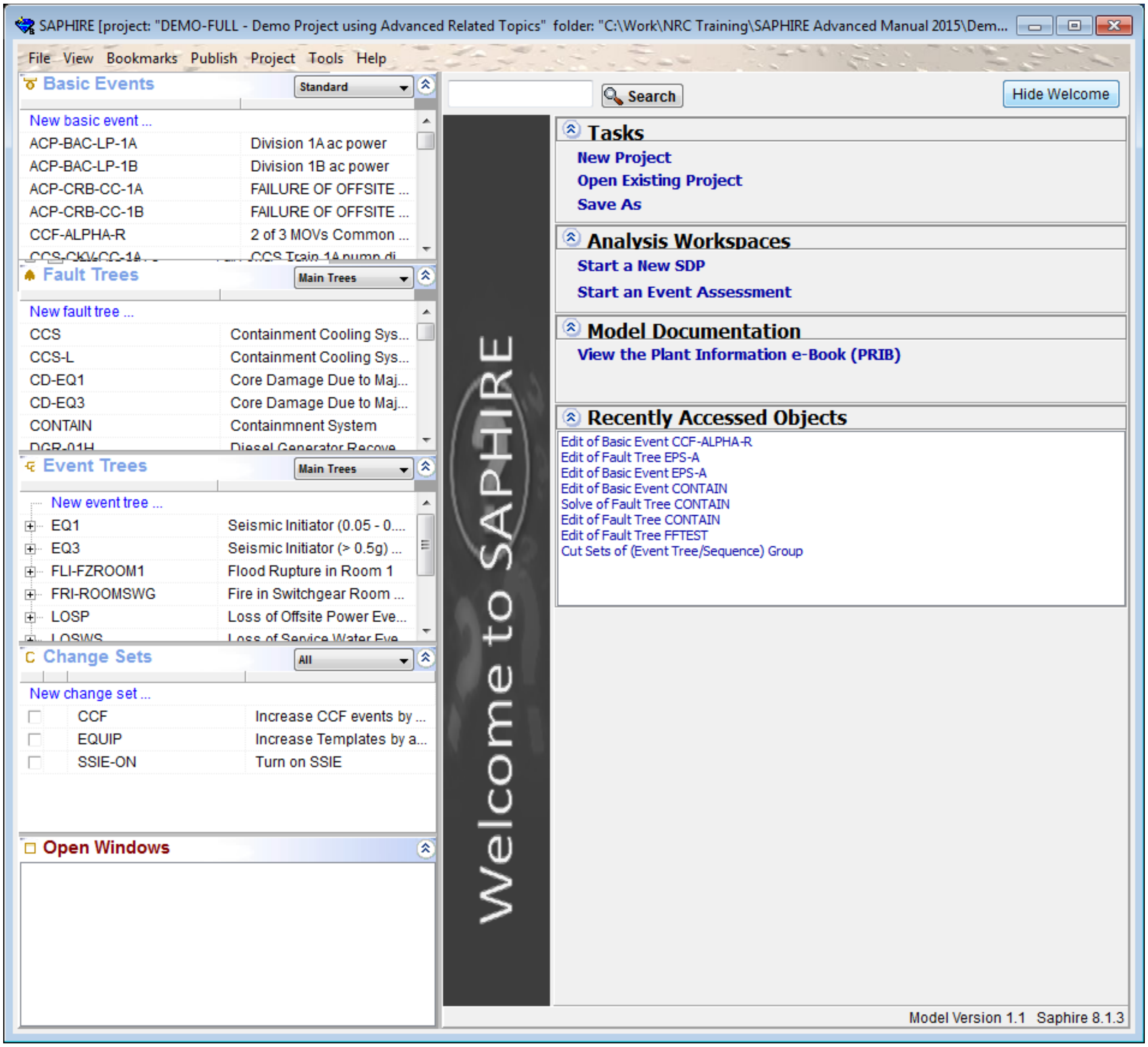
Clicking on Open Existing Project will open up a window to select the project. The user can also use the **File → Open existing project** option on the main menu bar at the top left of the screen.



An “Open Project” window will appear. Use the various Window Explorer options to find the folder containing the existing project.

To open the latest version of the project, select and open the *.SRA or FAM.DAT file; or, to open an archived version of the project, select a *.zip file.

SAPHIRE starts with the last project that was opened. With the “Welcome to SAPHIRE” option shown, SAPHIRE will provide a list of the objects that were recently modified or tasks performed along with Tasks that is designed to help analyst get started. These Tasks are quick links to 1) Create a New Project, 2) Open Existing Project, and 3) Save currently opened project As a new project.

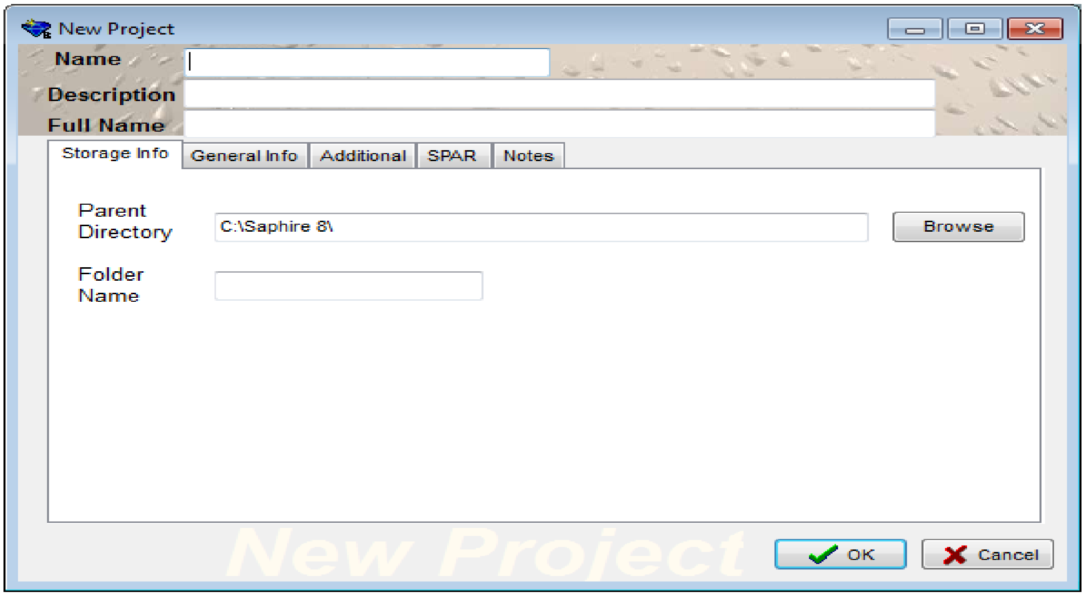


To hide this screen, click the Hide Welcome button in the upper right corner.

To create a new project:

- Select the New icon on the startup screen or the New Project option on the Welcome screen or use the main menu **File** → **New** → **Project** option.

- A "New Project" window will appear. Type the new project name and (optional) description.



- Verify that the Parent Directory is acceptable. If a different parent directory is desired, type in the new parent directory.

- Verify that the Folder Name is acceptable. By default, the Folder Name matches the project Name. (If the Folder Name appears blank, move the cursor out of the project Name field so that the Folder Name can update.)

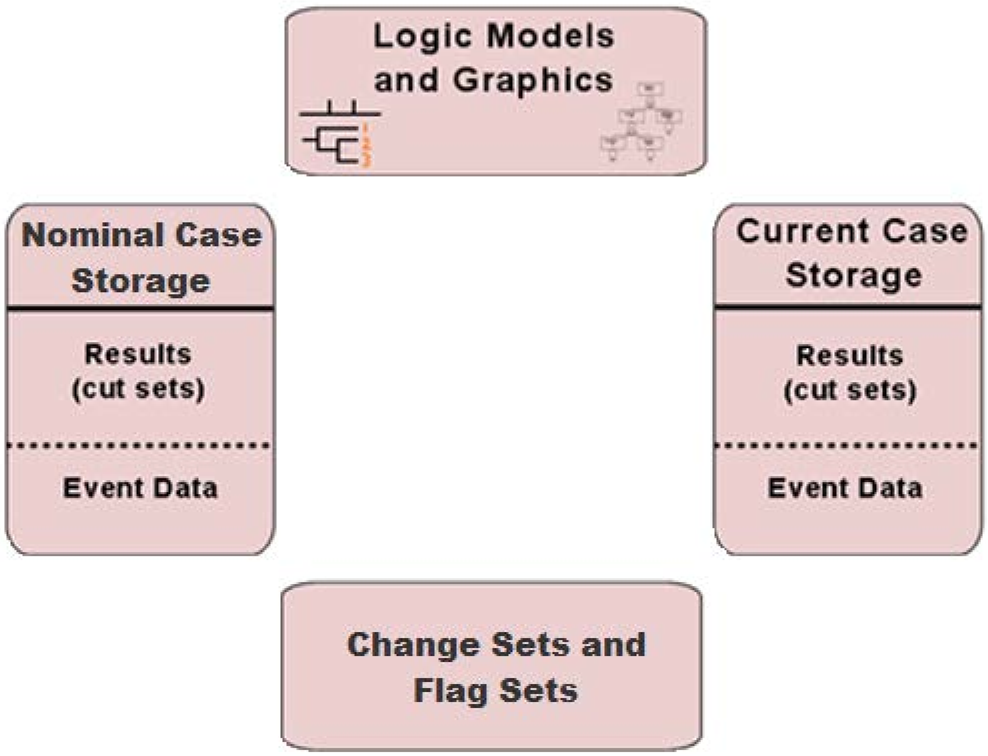
- Click OK. The main SAPHIRE menu will be displayed and the project name and the project location will be shown in the title bar above the menu.

- Modifications to a database (e.g., a new fault tree is developed) are always made to the currently selected project.

- For a given project, only one list is kept for all types of information. Thus, within a project, only a single copy of a particular fault tree, event tree, or basic event is ever stored in the database.

2.1. Nominal Case versus Analysis Case Data

- ♦ Nominal case and analysis cases (prior and current case) are two separate parts of a project database.



◊ **Nominal 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).

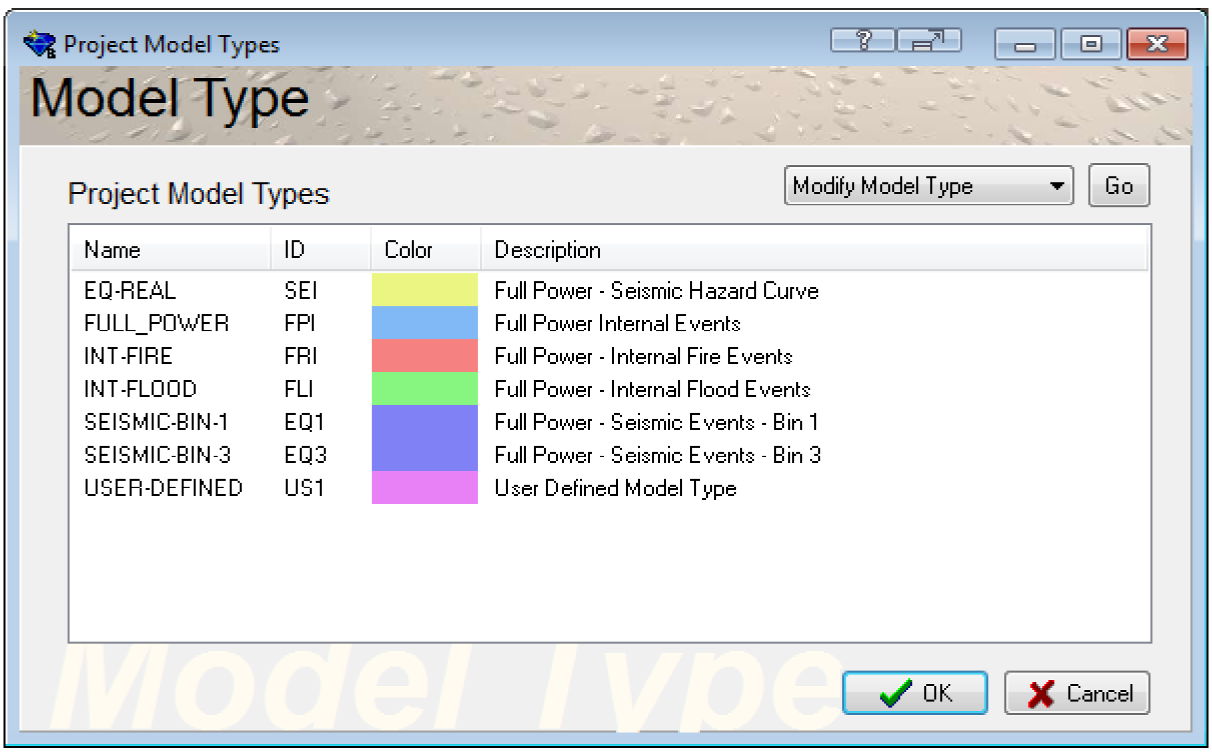
The Current Case is:

- ♦ Created through a Workspace (General Analysis, Significance Determination Process, Event Assessment) or a Change Set
- ♦ Used for sensitivity or event analysis
- ♦ All SAPHIRE calculations use the data stored in the **current** case.
- ♦ When creating or modifying the base model, the current case is the nominal case.

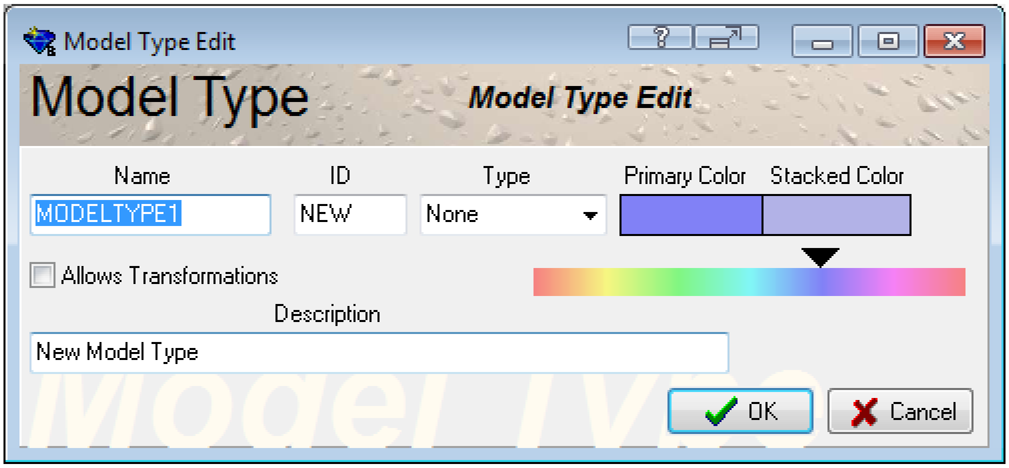
2.2. Model Types

- Versions previous to IRRAS 5.0 only had the capability to handle random analysis (i.e., Level 1, internal events PRA).
- SAPHIRE 8 can set up any number of model type analyses with the default being Random (Level 1, internal events PRA).
- Examples of such analyses include:
  - ◊ SEISMIC (external events related to earthquakes or other ground disturbances).
  - ◊ FIRE (external events related to fires).
  - ◊ FLOOD (external events related to flooding).
  - ◊ USER (User Defined).

To add or modify an existing model type select **Project → Model Types** to open the form below.

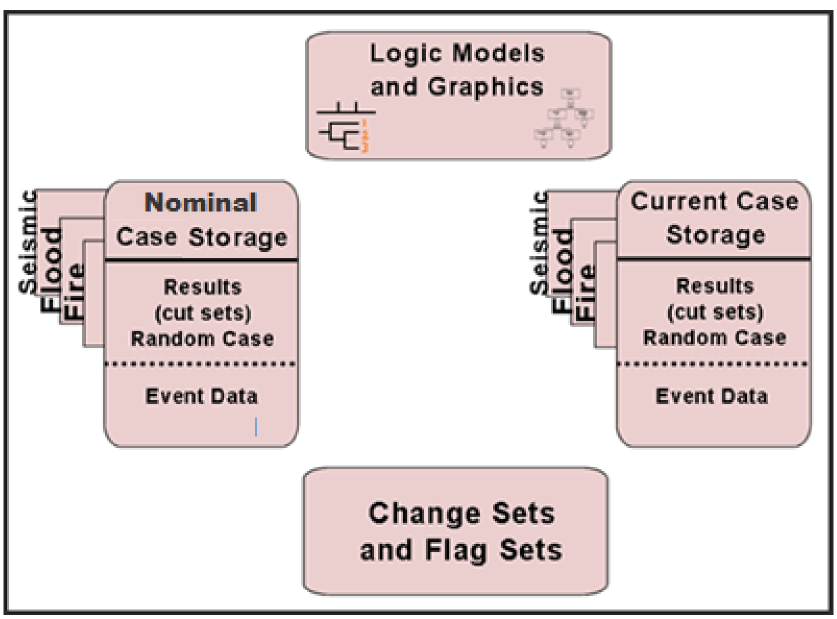


The default Model Type is set to RANDOM and it will be the only model type present when starting a new project. The project must always have at least one model type. To add a model type, select **Add Model Type** and click **Go**.



Enter a name, ID and description for the model type then select a unique primary/stacked color combination by using the slider arrow over the color bar and click **OK.**

The newly created model type will now be available when editing basic events and in analysis options discussed later in this manual. Events can be assigned to multiple model types or just one. If an event uses a model type, the event is prevented from deletion.

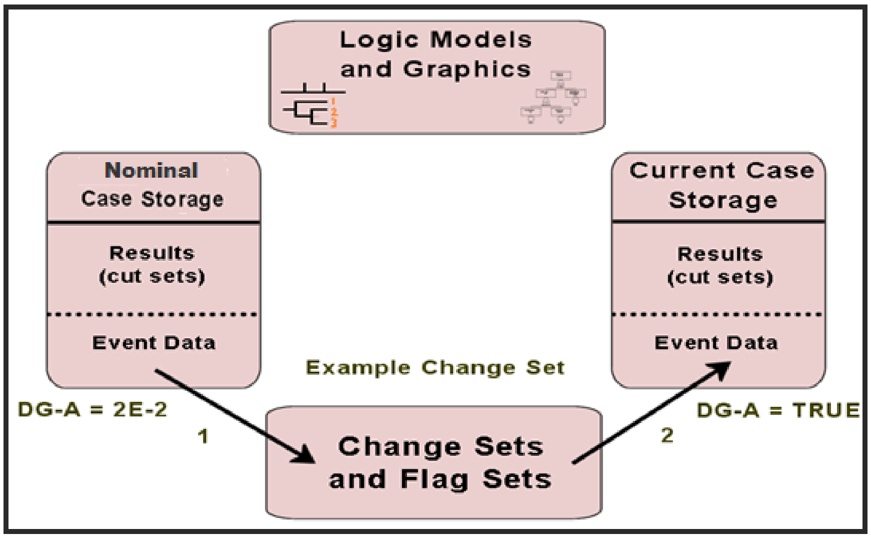


Model Types are discussed further in Section 14.

2.3. Change Sets

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

- Change Sets are created, modified, stored, activated or de-activated by using the Change Set list pane (main menu **View ➔ Change Sets)** or created, modified, and stored through **Project ➔ Change Sets** option.



2.3.1. Rules for Creating and Using Change Sets

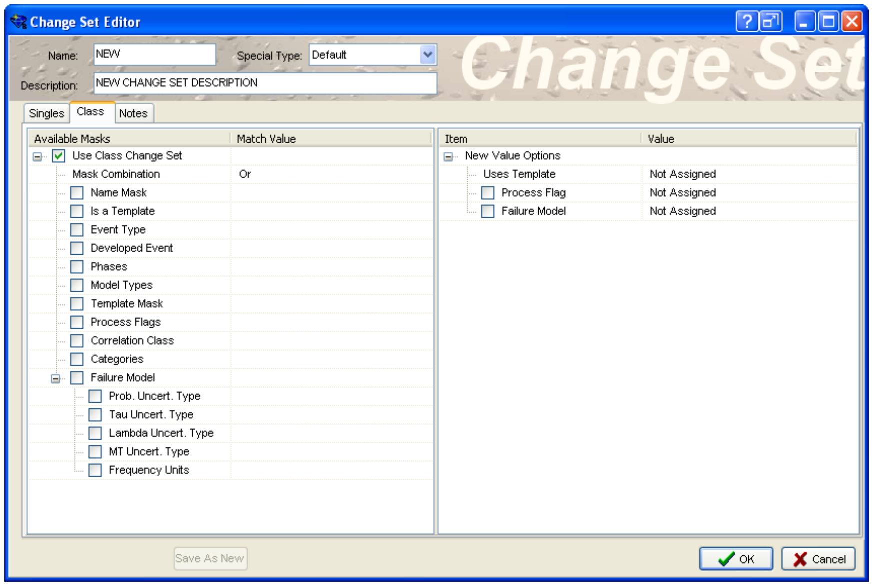
- No limit to the number of change sets that can be added to the data base.

- 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.

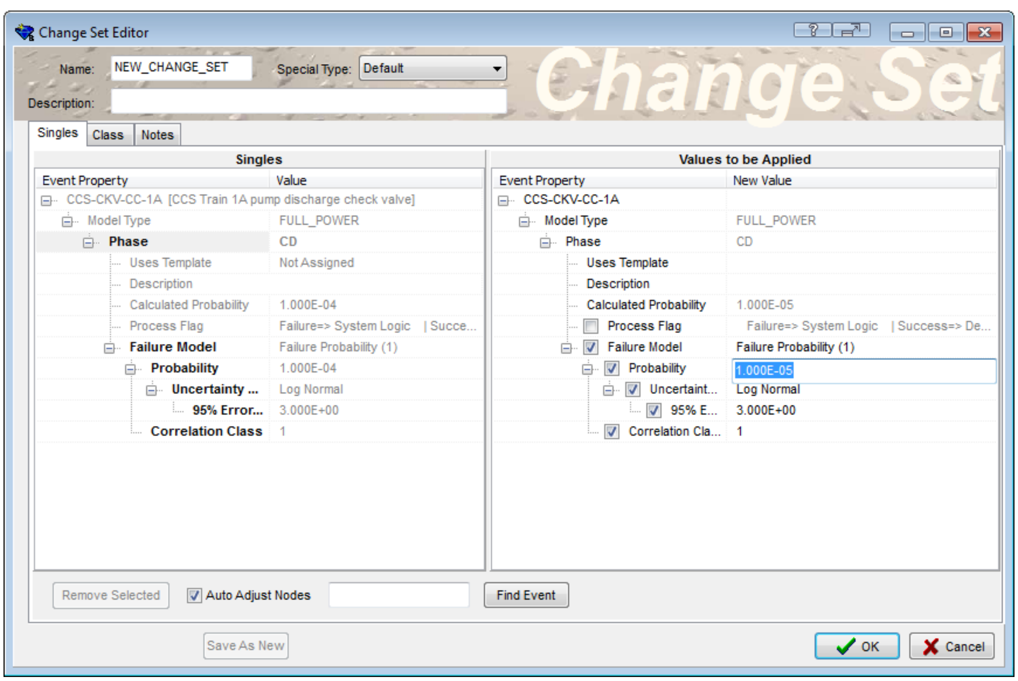
- Multiple change sets can be used in combination to create different sensitivity studies.

2.3.2. 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

2.3.3. Singles Changes



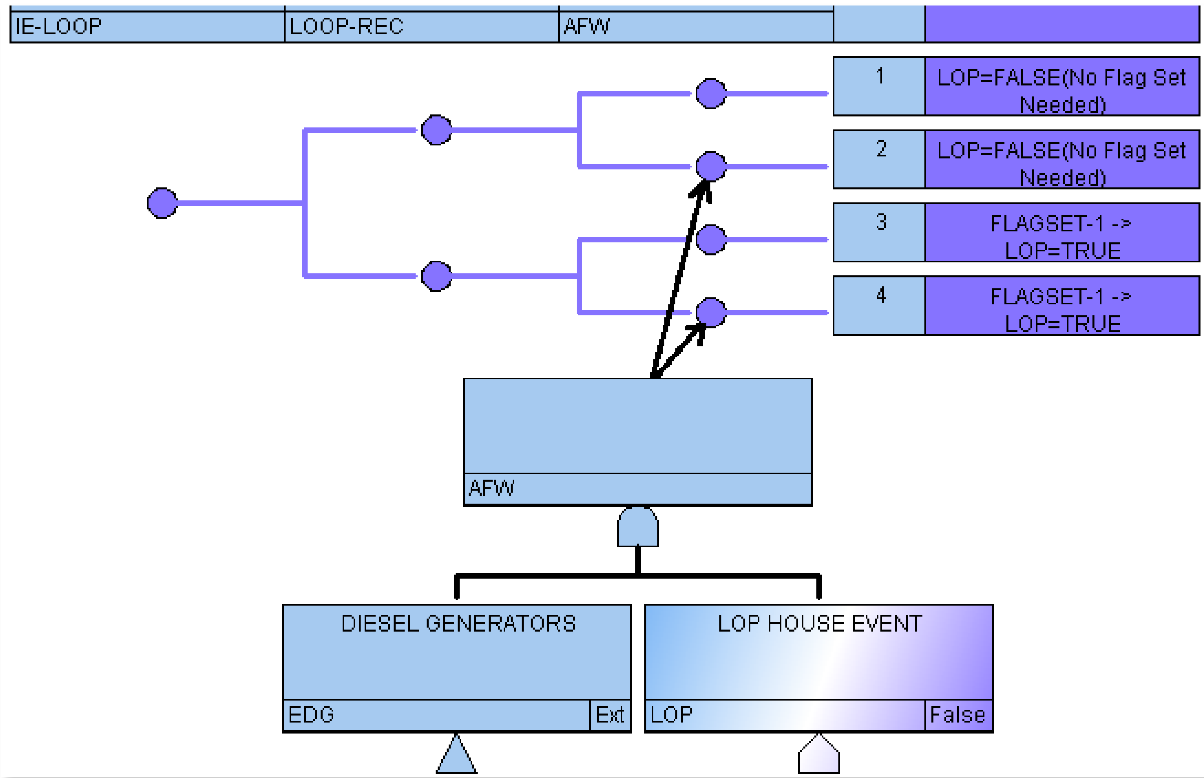
- Singles changes only modify individual, user-identified basic events
- The desired basic event is selected
- The changes to the basic event are then defined
- The order of “marking” a change set is important. (Change sets are marked by checking the check box in front of the change set of interest.)
- 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.

2.4. Flag Sets

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

This 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 turned off (FALSE). The setting of the house event is dependent upon the success or failure of recovering offsite ac power.



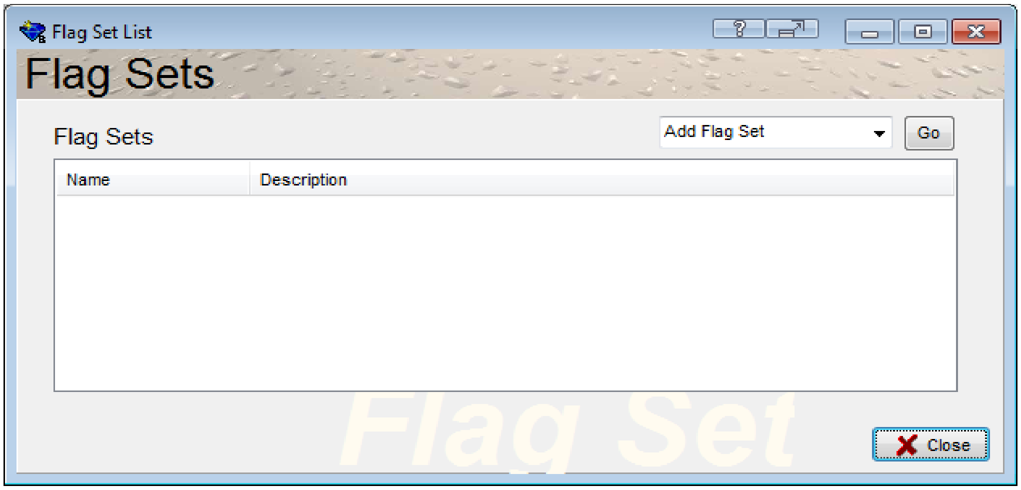
Flag Sets can only contain either house flag changes to the calculation type or process type changes. Consequently, the allowable changes that can be made in a Flag Set are:

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

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

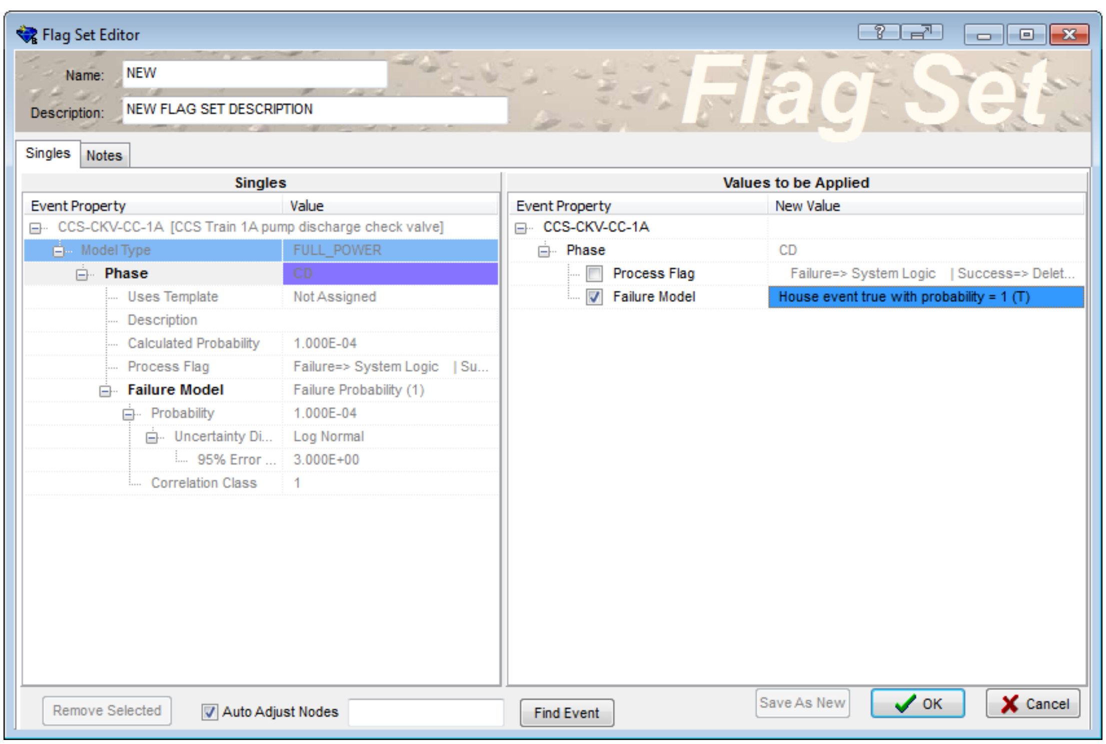
2.4.1. Making a Flag Set

Enter the **Project ➔ Flag Sets** option to open the Flag Set List window.



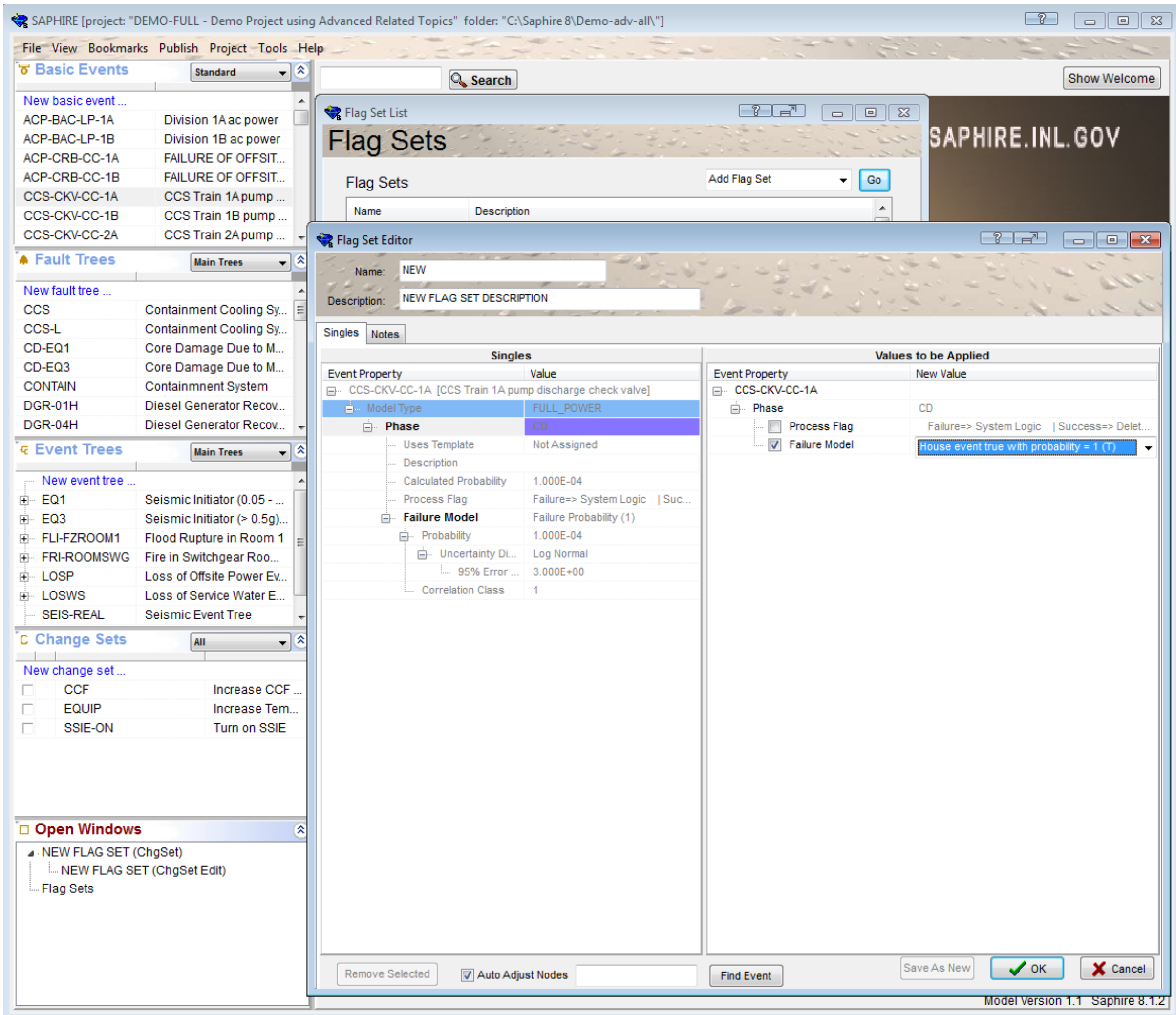
To create a Flag Set, highlight **Add Flag Set** in the pull down menu, click **Go** and enter the Flag Set name and description. Once Flag Sets are created, they can be modified or deleted through the Flag Set List window in the same manner.

With the Flag Set Editor now active, drag and drop the desired Basic Events into the Flag Set and change the desired Set Values.



The figure below shows a Flag Set being created. In this example C-CKV-CC-A has its Set Value changed to House = 1 (T). Values are changed simply by clicking on the

New Value and choosing from a drop down menu. In this illustration, C-CKV-CC-A is currently being dragged to the Flag Set Editor. Note the cursor shape changes from an arrow to a page with arrow when a drag and drop operation is being performed. To remove an event from the Flag Set, highlight the event and click on the button Remove Event.

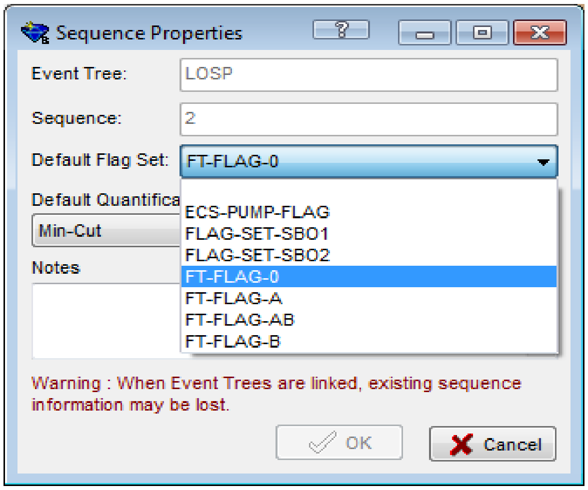


2.4.2. Using the Flag Set

Once the Flag Set has been created, it needs to be assigned to a sequence or fault tree.

- ◊ To add the Flag set to the sequences, highlight the sequence, right mouse click and select **Edit Properties**. (Not recommended, since every time the event tree is saved the sequences are regenerated and the prior sequence information is over-written with the new sequence information.)

◊ Click the drop down box "Default Flag Set:" and select the Flag Set



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

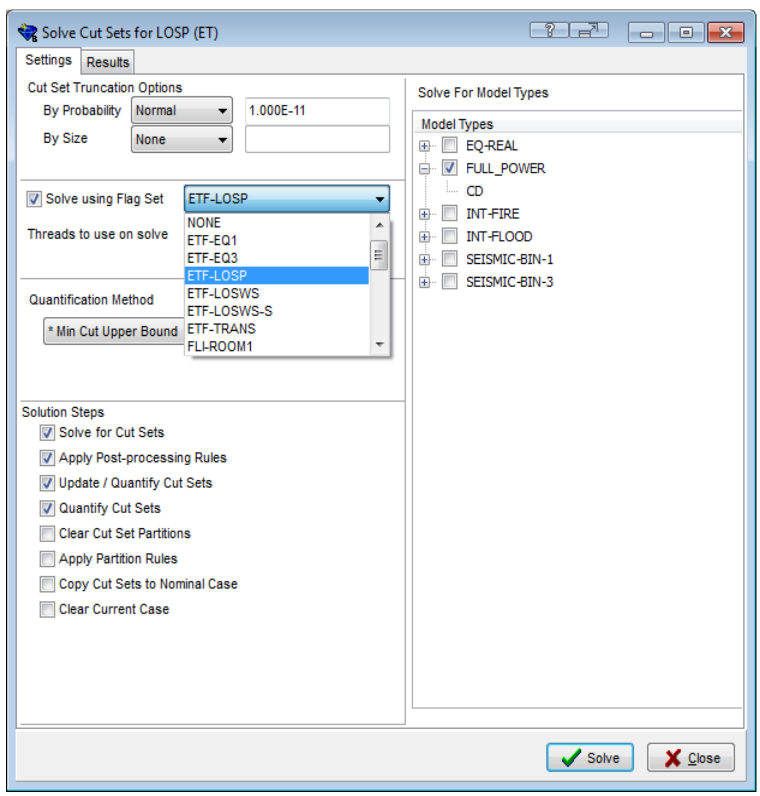
- Check the Solve using Flag Set check box
- The drop down box will now allow the selection of specifying a Flag Set

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

◊ The "Item Defined" option is the same as above (use assigned Flag Set)

◊ The "NONE" option will ignore all assigned Flag Sets.

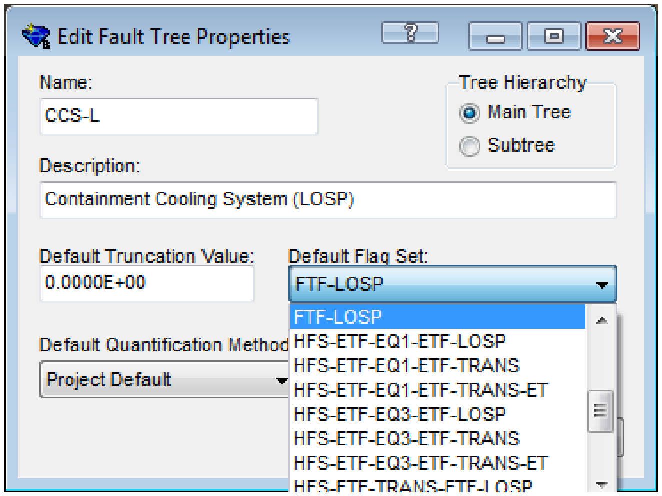
◊ The "Flag Set Name (FLAG-SET-SBO1)" option will use this manually assigned Flag Set with SAPHIRE generates sequence cut sets.



Alternatively, Flag Sets can be assigned to sequences via event tree linking rules (a.k.a. Dynamic Flag Set). The latter is more common and an overview is discussed below.

♦ To add Flag set to Fault Tree:

◊ Highlight the fault tree, right mouse click and select **Edit Properties**.



◊ Click the drop down box “Default Flag Set:” and select the Flag Set

2.5. Dynamic Flag Sets

“Dynamic” Flag Sets are a special type of Flag Set that is 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 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.

Dynamic Flag Sets are the same as Flag Sets, since 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 **Project → Flag Sets** 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.

2.6. Hierarchical Flag Sets

"Hierarchical" Flag Sets are created when SAPHIRE links multiple event trees together to create accident sequences and these event trees contain event tree specific flag sets.

A Hierarchical Flag Set is assigned to a sequence based on the event tree logic and event tree linking rules (Section 4). The hierarchical flag set is created by layering one flag set upon another to create a super flag set (i.e., single flag set that contains all of the information). The hierarchical flag set orders the inputs of each flag set as they appear; therefore, the inputs in the first flag set is loaded first and then the subsequent inputs from the next flag set is loaded, etc.

Advantages of "Hierarchical" Flag Sets are:

- ♦ Flag sets get appended together instead of having to create a single flag manually by looking through all potential sequences and transfers.
- ♦ They are automatically created based on the event tree linking rules (Section 4).

Hierarchical Flag Sets are the same as regular Flag Sets, since only basic event calculation types can be changed (T, F, or I) and the change can only be specified to individual basic events (i.e., no class changes).

The Hierarchical Flag Set name will appear in the Flag Set field under **Project → Flag Sets** after the flag set is created during the event tree linking process. The Hierarchical Flag Set name is assigned by SAPHIRE and is based upon the naming of the flag sets that get grouped together. The user does not have control over the naming process.

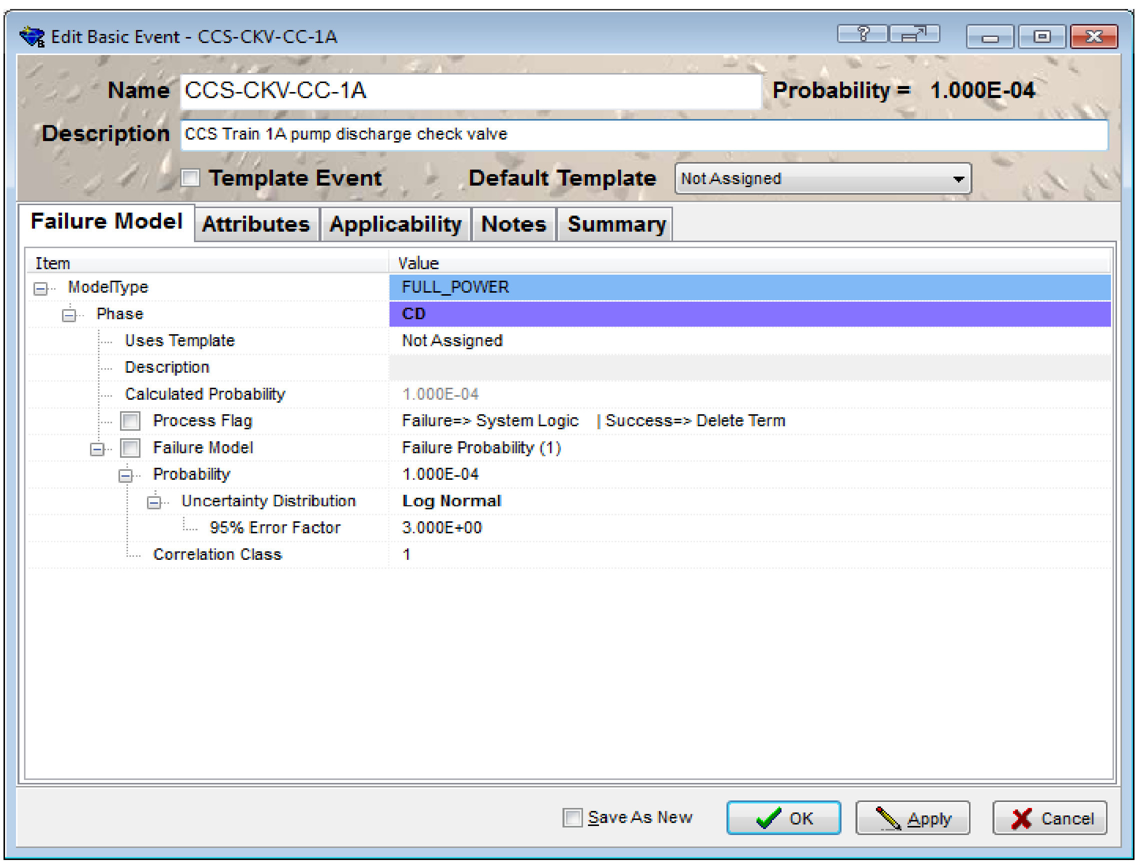
Hierarchical Flag Sets will be discussed in greater detail in Section 9. This section will demonstrate the how to use the Hierarchical Flag Sets and the different features of the Hierarchical Flag Set rules.

| 3 | BASIC EVENT INFORMATION

Section 3 introduces the compound event features, common cause event calculation calculators, and human error worksheets found in SAPHIRE. The compound event allows SAPHIRE to use built-in numerical library to determine a basic event's probability. The common cause event calculations are used to automatically calculate the common cause failure probabilities. The human error worksheets are used to calculate human error probabilities.

3.1. Modify Basic Events

- To modify data for an existing event, double-click on the basic event in the Basic Events window you want to edit or right-click to invoke the pop-up menu and select **Edit Basic Event**.

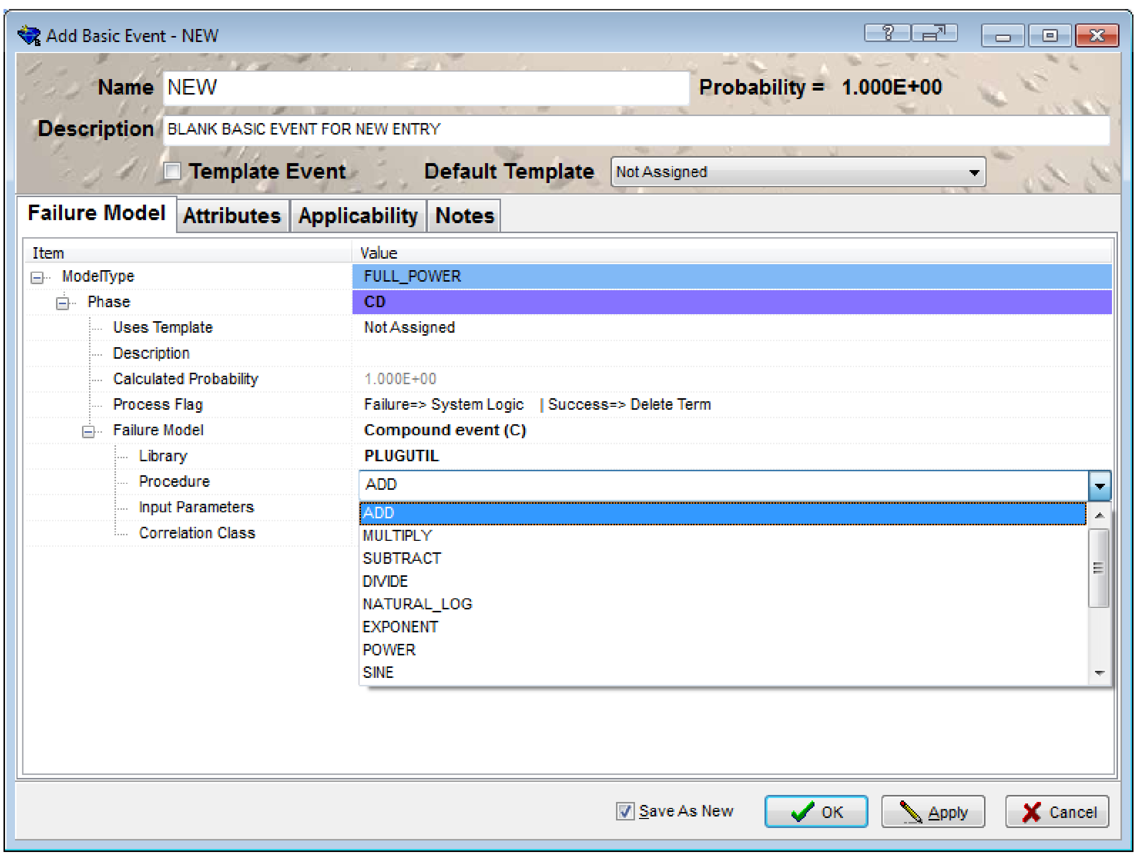


3.2. Compound Events

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 offsite power recovery, common-cause failure, flow accelerated corrosion, etc.

A compound event is generally expressed as a function of other basic events (within the same project). For example, in a “supercomponent” case, one would identify the components (up to 20) that make up the supercomponent (i.e., components that are in series or parallel).

To create a compound event, in the Basic Events list panel, double-click on “New basic event”. This opens the Add Basic Event form:



- Click on the text (Value column) to the right of Failure Model to activate the drop-down list.
- Choose “Compound event (C)”.

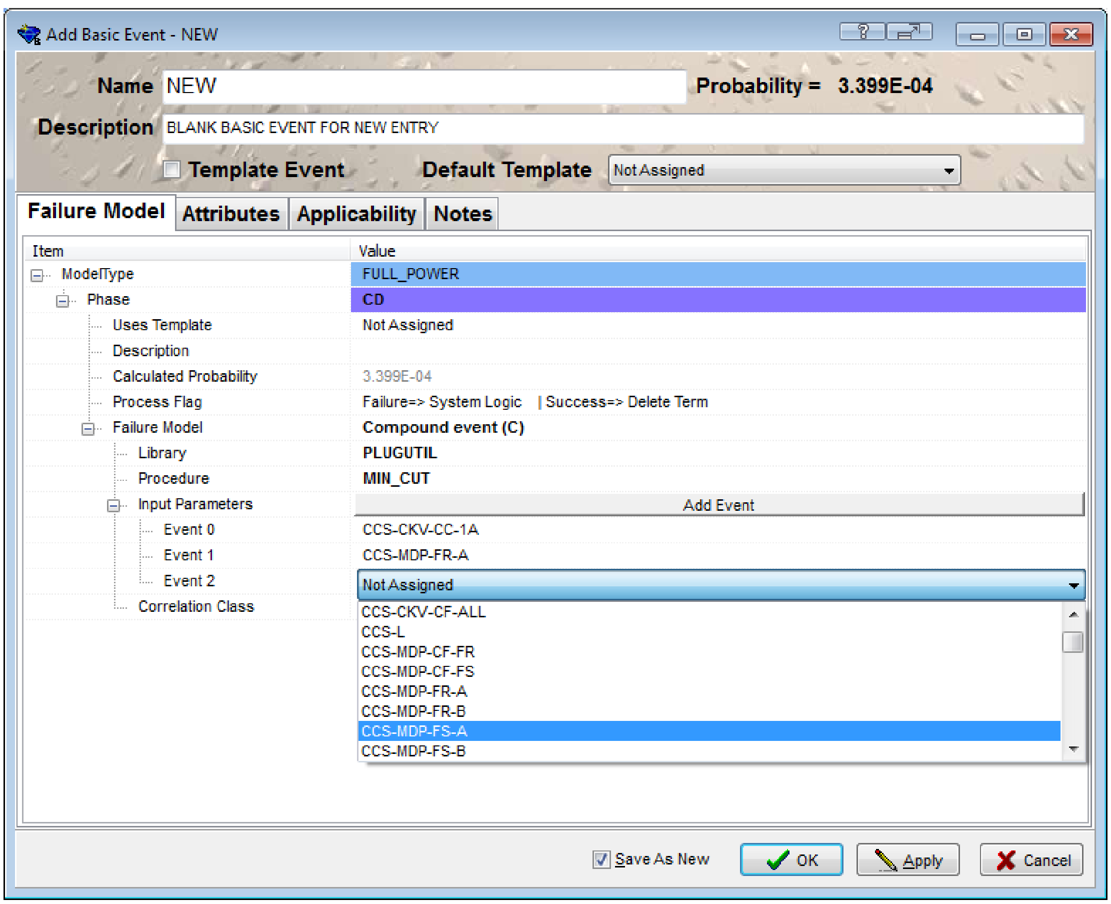
◊ The “Library” drop down option lists the different modules available to the analyst. To create a supercomponent type event, the PLUGUTIL.DLL library is selected.

◊ Click the “Procedure” option text field, 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 upper bound approximation to determine the probability.

◊ Click on the “Add Event” button to the right of “Input Parameters” to add the number of Sub Events desired.

◊ Click on the event field to the right of each Sub Event to activate the drop-down menu of all events in the project. Select an event by clicking on it.



As events are added, SAPHIRE 8 calculates the probability that will be used for the compound event and displays it to the right of the Name of the event.

3.3. 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.

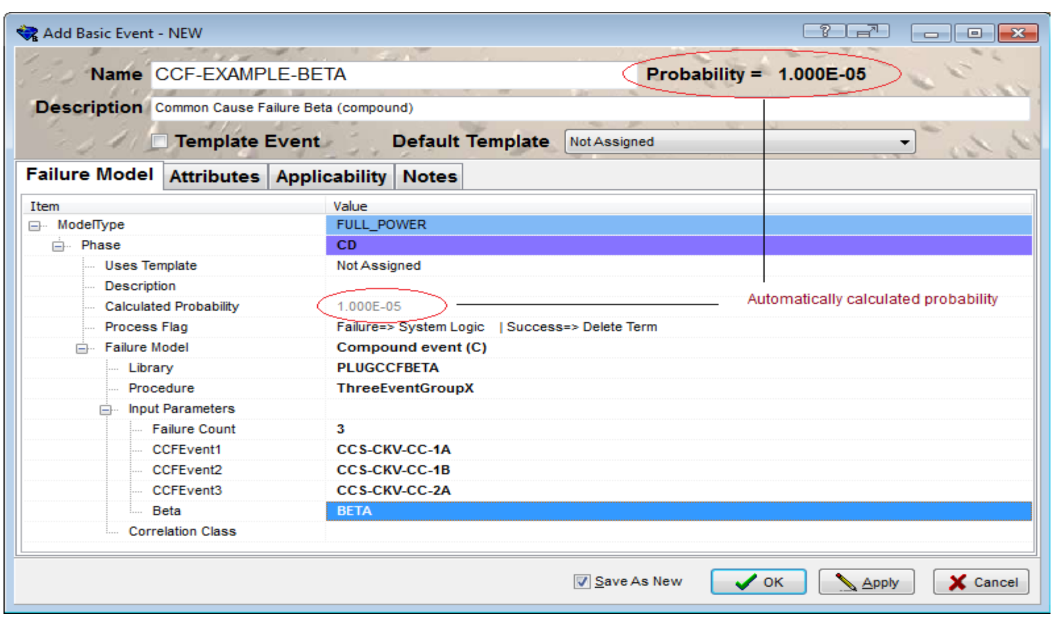
Within SAPHIRE, there are three basic types of common-cause models. The first is known as the Beta factor model. The second is known as the Multiple Greek Letter (MGL) model. The third is the Alpha Factor model. The alpha factor model is calculated based on testing schemes. The testing scheme calculation types are 1) staggered testing and 2) non-staggered testing. Both methods can also be calculated using the Risk Assessment Standardization Project (RASP) type calculation [R-calc]. The [R-calc] uses the first two approaches and uses a conditional probability calculation approach to automatically adjust for changes such as sequence flag sets and event assessments.

3.3.1. Beta Factor Model

The beta factor model can be implemented into SAPHIRE using the compound CCF event calculation. The following will discuss the development of the CCF event using the beta model as a Compound Event (C).

To use the Compound event (C) beta factor model in SAPHIRE, under the Basic Events list panel, double-click on **New basic event ….** Using the drop down menus:

◊ In the dropdown list to the right of **Failure Model**, select **Compound event (C)**.



- ◊ Click the text in the Value column next to “Library” and select **PLUGCCFBETA** operation.
- ◊ Enter the proper number event group for the common cause failure (i.e. ThreeEventGroup for a group of three components failing due to common cause).
- ◊ Enter a failure count equal to the number of component failures that would cause a group failure.
- ◊ Select the basic event from the drop down list to the right of each CCFEvent.
- ◊ Select the Beta Factor basic event for the Beta Factor Model Calculation. This will have to be created as a basic event (value event) to be available for selection.

Use the **New basic event…** as needed to create the Beta Factor basic event needed.

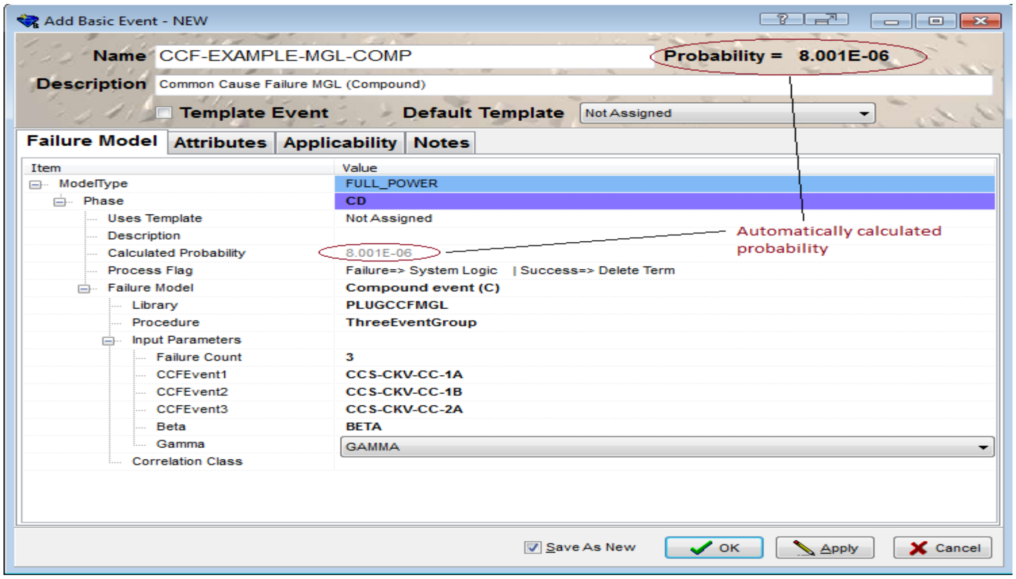
3.3.2. Multiple Greek Letter (MGL) Model

The MGL model can be implemented into SAPHIRE using either the compound CCF event calculation or the full Boolean expression using the R-calculation. The following will discuss the development of the CCF event using the MGL model as a Compound Event (C) and R-calculation (Rolled-Up).

To use the Compound event (C) MGL method in SAPHIRE, under the Basic Events list panel, double-click on **New basic event …**. Using the drop down menus:

- ◊ In the dropdown list to the right of **Failure Model**, select **Compound event (C)**.
- ◊ Click the text in the Value column next to “Library” and select **PLUGCCFMGL** operation.
- ◊ Enter the proper number event group for the common cause failure (i.e. ThreeEventGroup for a group of three components failing due to common cause).
- ◊ Enter a failure count equal to the number of component failures that would cause a group failure.

◊ Select the basic event from the drop down list to the right of each CCFEvent.



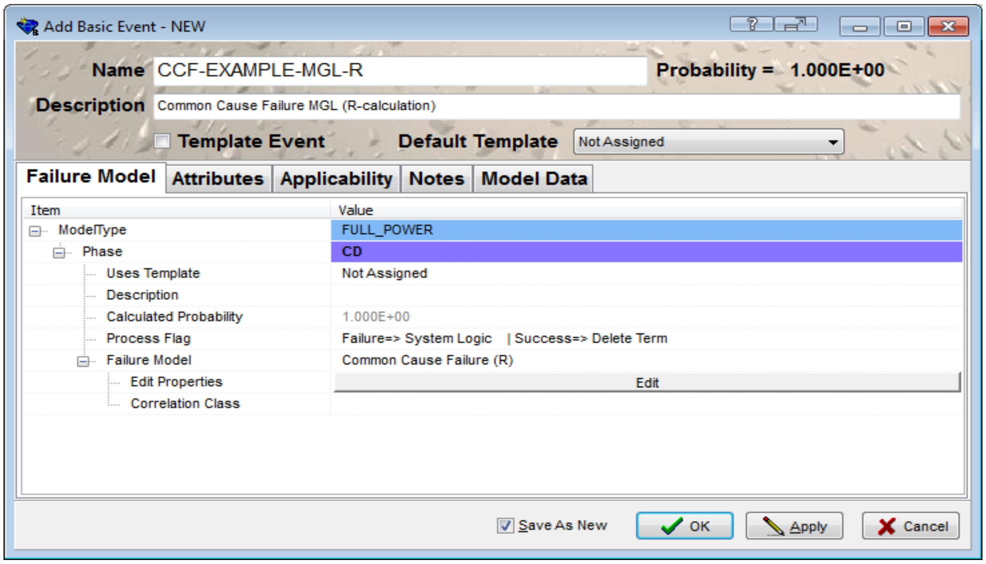
◊ Select the Beta Factor for the MGL Calculation (and others depending upon number of components in group). This may have to be created as a basic event (value event) to be available for selection.

Use the **New basic event…** as needed to create basic events needed such as the Beta Factor event without having to close the CCF event.

To use the R-calculation (R) MGL method in SAPHIRE, under the Basic Events list panel, double-click on **New basic event ….** Using the drop down menus:

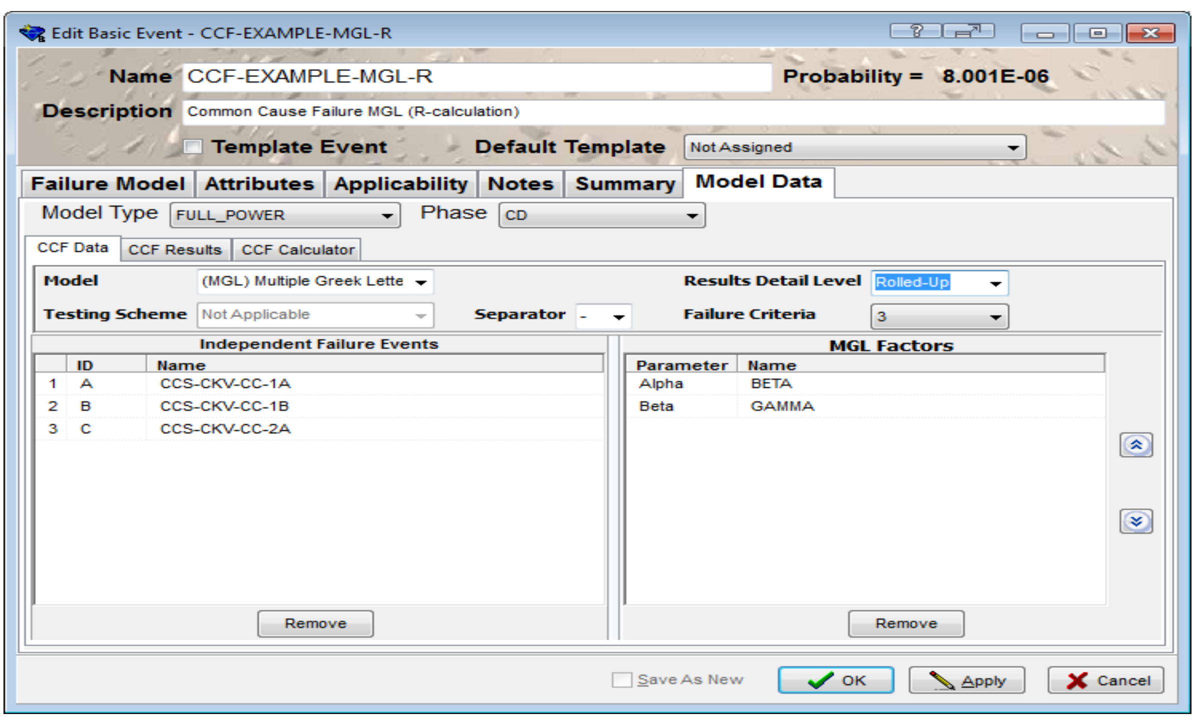
◊ In the dropdown list to the right of **Failure Model**, select **Common Cause Failure (R)**.

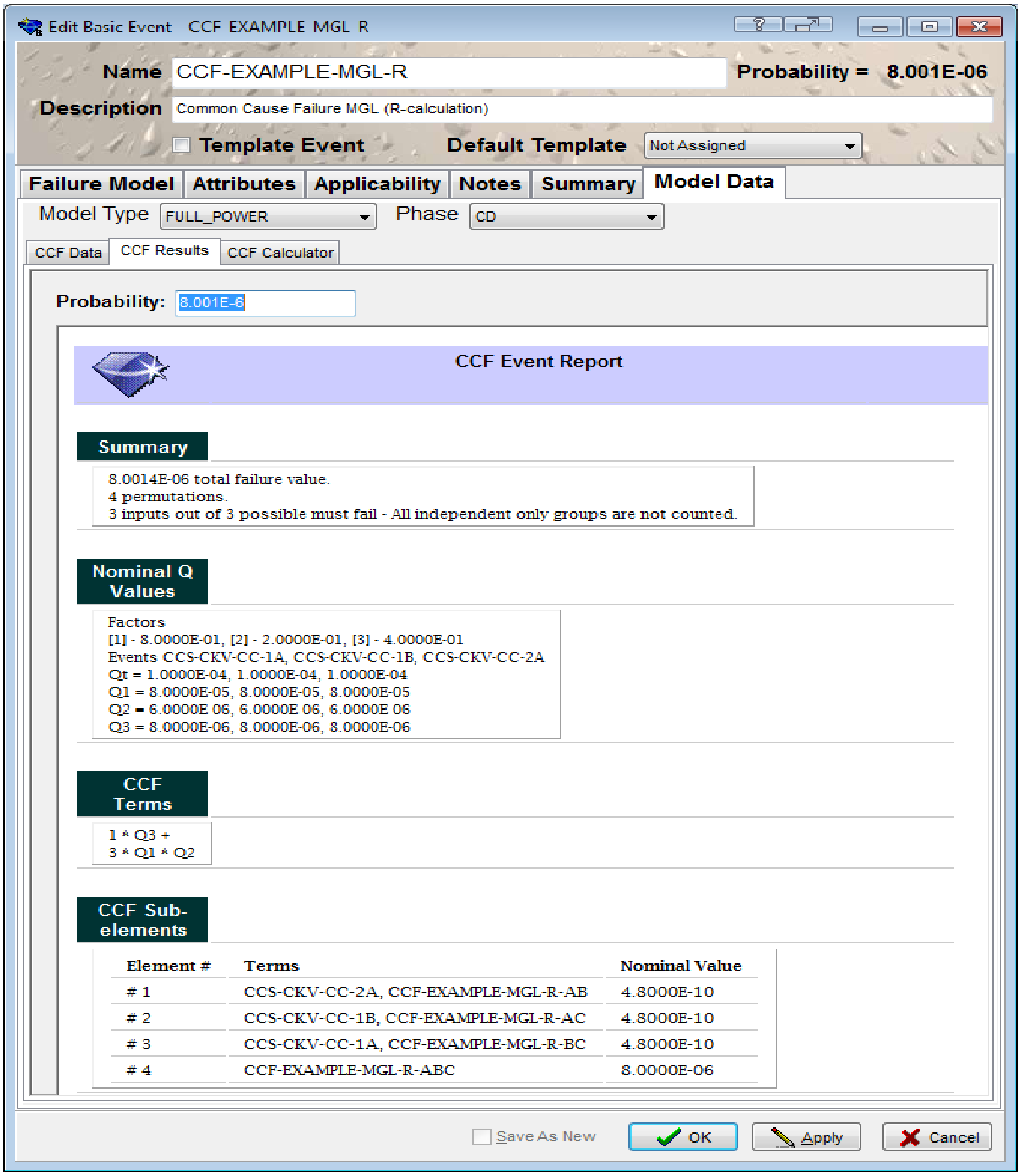
◊ Click the edit bar and SAPHIRE will open up a new window for the analyst to input the components and the MGL parameters (use a drag and drop of the components into the **Independent Failure Events** column and MGL parameters into the **MGL Factors** column.



◊ Select the CCF model type (MGL Multiple Greek Letter) from the drop down option. Based on the number of components, SAPHIRE will automatically default to all components must fail in the Failure Criteria drop down option.

◊ Click on the CCF Results tab and SAPHIRE will display the full calculation used to determine the CCF 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.

3.3.3. Alpha Factor Model

The Alpha Factor model as discussed in this section will use the Compound Event (C) Failure Model option. The Alpha Factor model is incorporated within the R-calculation type (i.e., full Boolean expansion) and will be discussed in next section.

In SAPHIRE, this model uses one of two different equations, depending on the type of testing for the components in question (how data is gathered and quantified).

◊ 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.

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

♦ To use the Alpha Factor model in SAPHIRE, under the Basic Events list panel, double-click on **New basic event** …. Using the drop down menus:

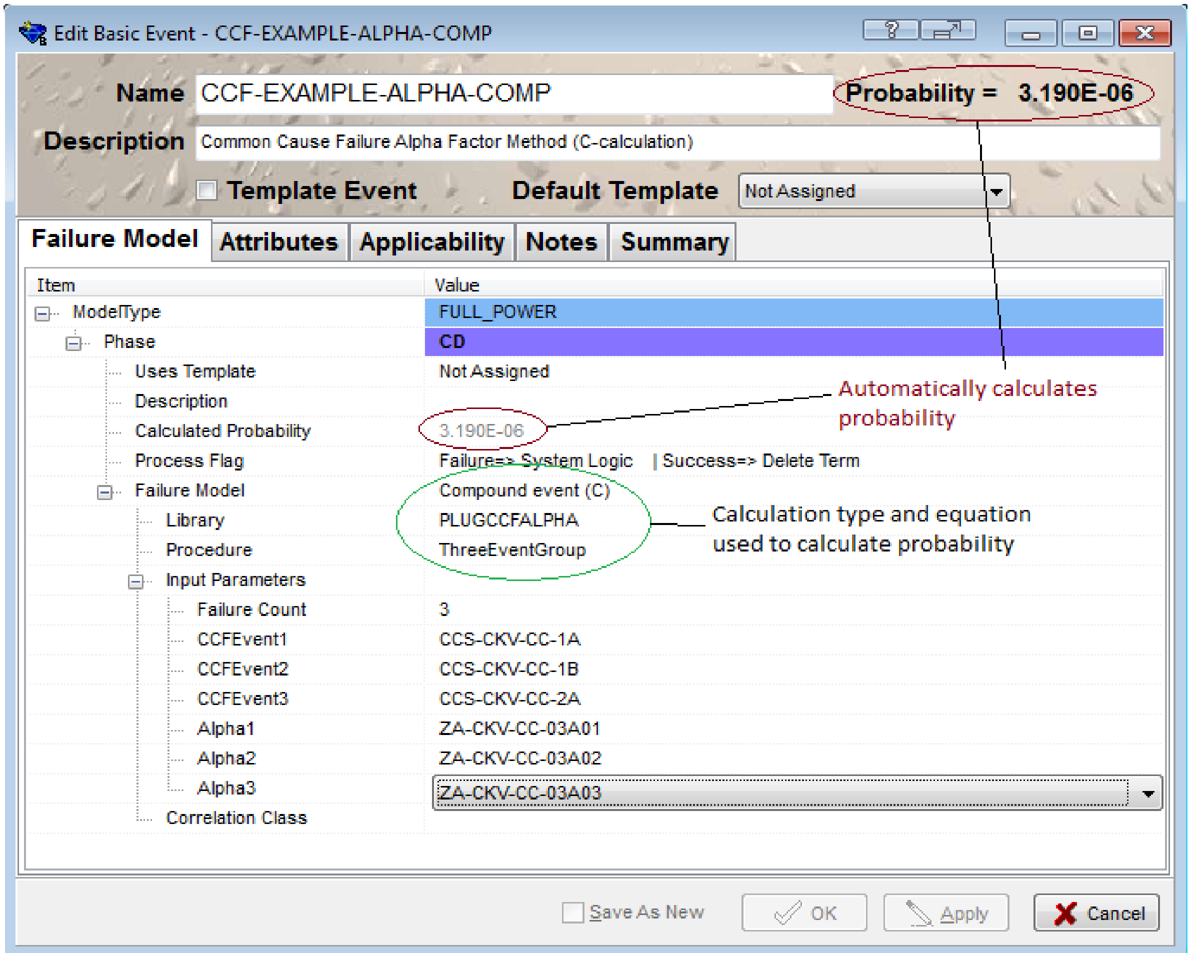
◊ In the Value column of Failure Model, select Compound event (C).

◊ Select PLUGCCFALPHA (or PLUGCCFSTAG if staggered testing) from the drop down list in the Value column of Library.

◊ Enter the proper number event group for the common cause failure for Procedure (i.e. ThreeEventGroup for a group of three components failing due to common cause).

◊ Enter a Failure Count equal to the number of failed components that would cause the system to fail (i.e. 2 if 2 of 3 components must fail due to common cause to cause the system to fail).

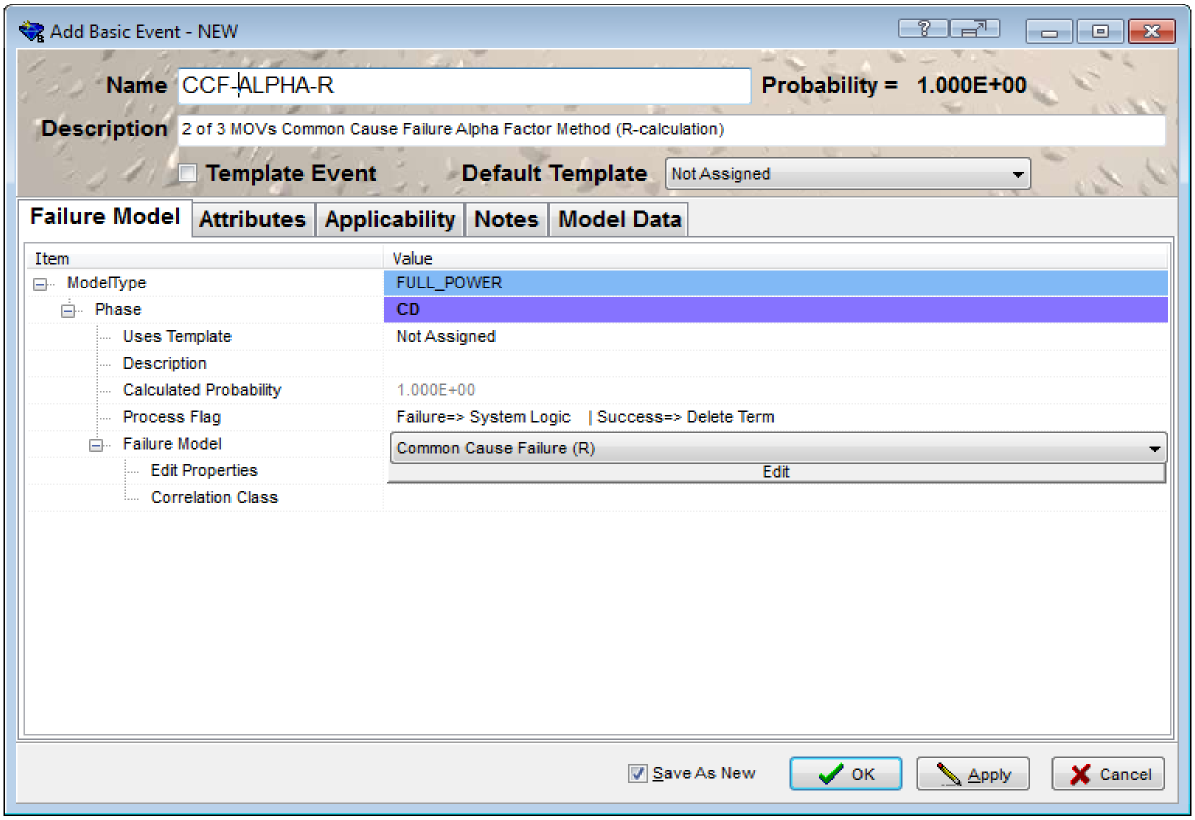
◊ Select the basic event from the drop down list in the Value column of each CCFEvent#.



◊ Select the Alpha Factors for the calculation. These are usually pre-loaded as basic events and correspond to the component number within the group size selected (i.e. Alpha factor 2 in group size 2 for a particular component type).

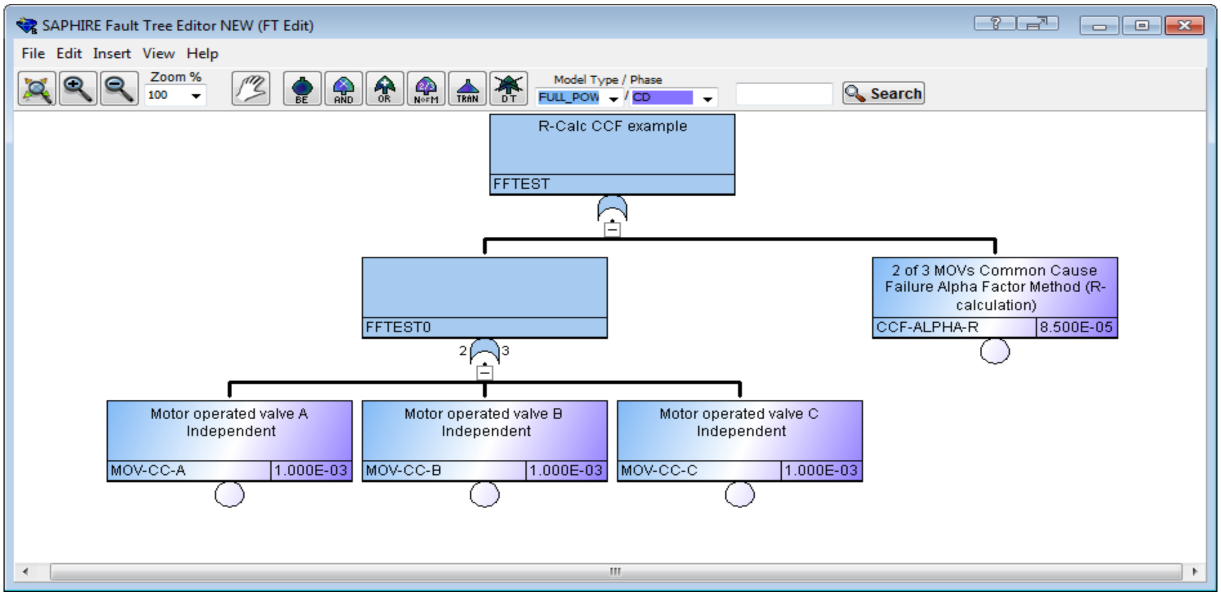
3.3.4. R-calc Common-Cause Failure Module

This CCF module uses Boolean expansion (Basic Parameter Model) to calculate the CCF event probability. The primary difference between this calculation and using the compound event calculation is the ability to have the full detail set of cut sets incorporated into the final result and how the conditional probability is calculated when performing event assessments as a result of setting components' failure modes to 1, T, 0, or F.



This common cause method is a new calculation type to SAPHIRE 8. The common cause failure type "R" can be found in the basic events under the "Failure Model" option.

To demonstrate this CCF module, a simple fault tree will be created with three redundant motor operated valves (MOVs) and a new basic event representing the CCF of all three components. The three MOVs are represented by basic events MOV-CC-A, MOV-CC-B, and MOV-CC-C, and the CCF basic event is represented by CCF-ALPHA-R. The success criterion for this system is that two components have to work for the system to work. Consequently, if two (of three) components fail, the system is failed. The fault tree representing the system is shown.



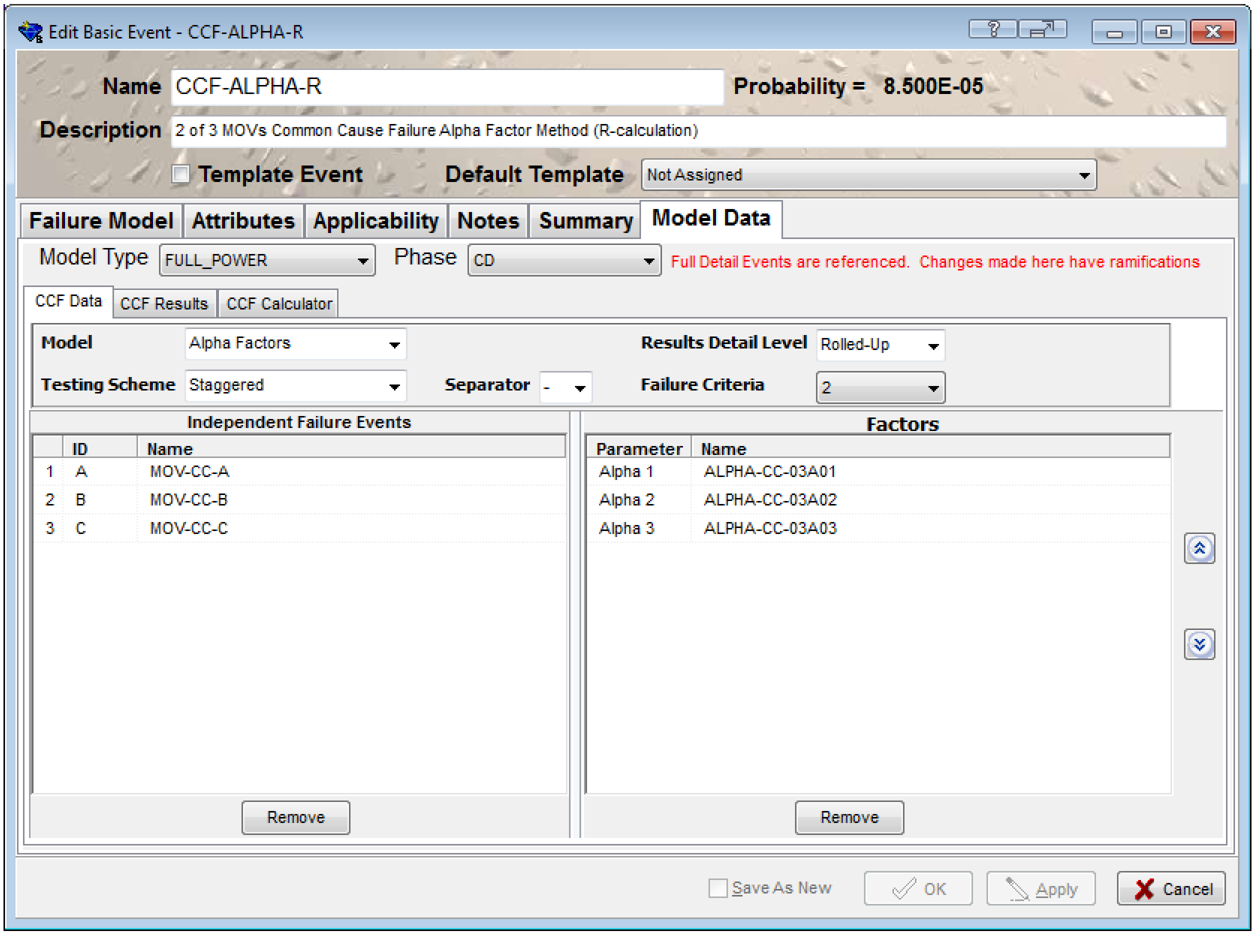
3.3.4.1. *Entering CCF Data into R-calc*

The basic event CCF-ALPHA-R is a part of the fault tree graphic as shown. This basic event needs to be edited.

The following discusses the process to set up the R-calc CCF event:

- ◊ Select the **Common Cause Failure (R)** option in the Failure Model text field.
- ◊ Click the **Edit** bar to open the R-calc CCF form for editing.
- ◊ Choose Alpha factors or MGL as the **Model** type from the pull-down menu.
- ◊ Click and drag the desired Basic Events into the **Independent Failure Events** window
- ◊ Click and drag the MGL or Alpha factors, which are located in the Basic Events list into the **Factors** window. The MGL or Alpha factors need to be entered based on the number of components in the group (3-train, 4-train, and so on) and placed into the correct parameter field (i.e., Alpha 1 must be alpha factor 1).
- ◊ Choose the number of failures that would cause a system failure. In this case it would be 2.

◊ The testing scheme can be chosen as Staggered or Non-Staggered



◊ The Separator character is used by SAPHIRE to determine the “auto generated” event names, as we will see when describing the results of the CCF calculation.

◊ Choose the Results Detail Level

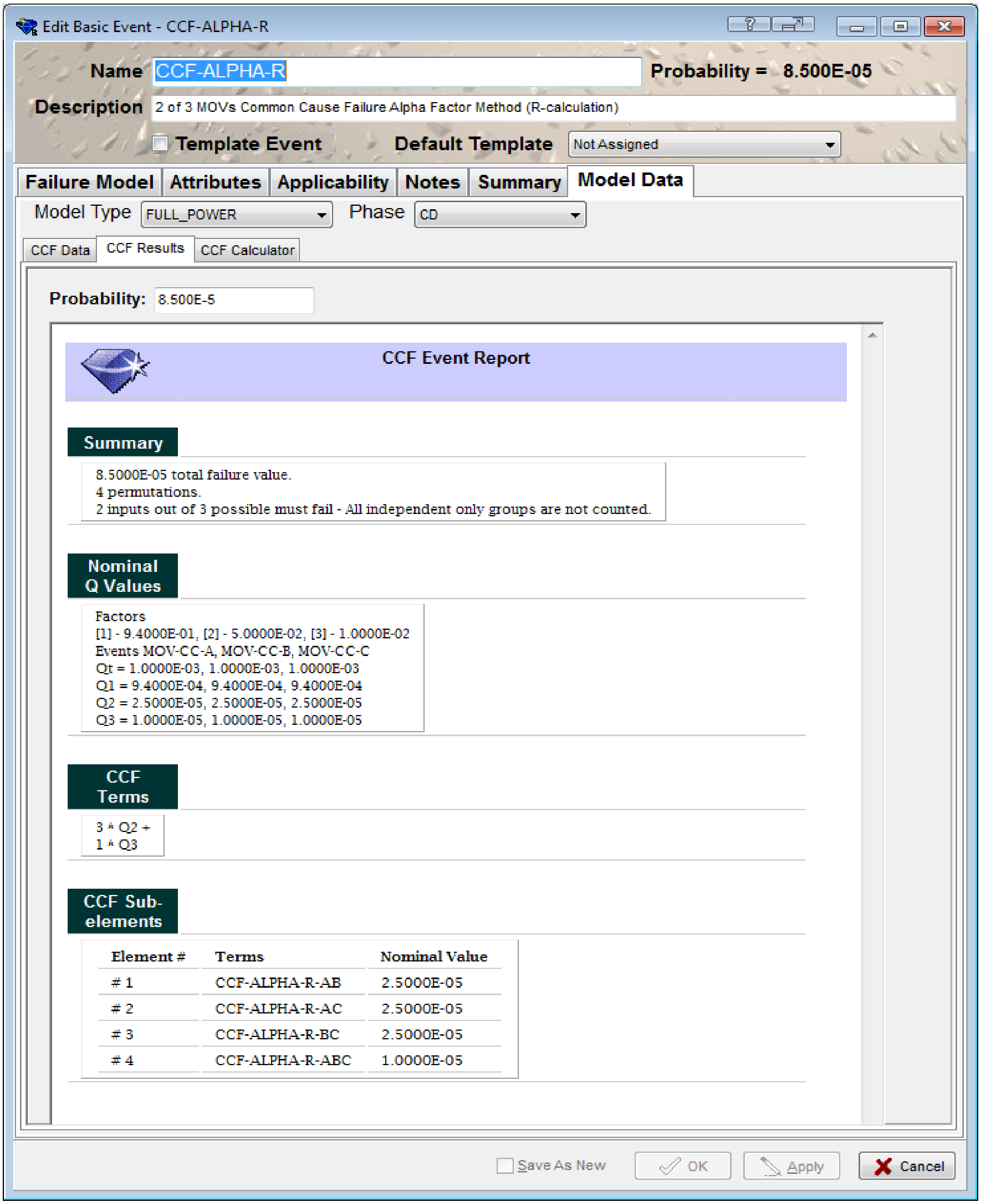
- **Rolled Up** will create one cut set (basic event) with the probability posted.

- **Full Detail** will create all possible cut sets for the components based on the inputs and success criteria (e.g., 2 of 3 failures). These cut sets will be included in the overall fault tree cut sets.

The probability displayed is the overall probability that will be used for this CCF basic event. In this example, the overall CCF probability is 8.500E-5.

3.3.4.2. Viewing RASP Result Details

To view the results for this example, click on the "**CCF Results**" tab. The detailed results appear as:



Evaluating the results in more detail:

Summary

8.5000E-05 total failure value.
4 permutations.
2 inputs out of 3 possible must fail - All independent only groups are not counted.

The Summary block lists the overall calculated CCF failure probability; number of permutations obtained based on the input information, and lastly lists the success criteria. The statement about "ALL independent …" is presented to let the analyst know that the overall probability does not include the independent failure cut sets.

Nominal Q Values

Factors
[1] - 9.4000E-01, [2] - 5.0000E-02, [3] - 1.0000E-02
Events MOV-CC-A, MOV-CC-B, MOV-CC-C
Qt = 1.0000E-03, 1.0000E-03, 1.0000E-03
Q1 = 9.4000E-04, 9.4000E-04, 9.4000E-04
Q2 = 2.5000E-05, 2.5000E-05, 2.5000E-05
Q3 = 1.0000E-05, 1.0000E-05, 1.0000E-05

The nominal Q value block provides the calculated terms based on the Basic Parameter Model (Q_1, Q_2, and Q_3 terms) along with the input values.

CCF Terms

3 * Q2 +
1 * Q3

The CCF Terms block lists the different permutations obtained. For this example, there are three combinations of Q_2 terms, and only one Q_3 term. Consequently, the full expression to determine the CCF probability is given by:

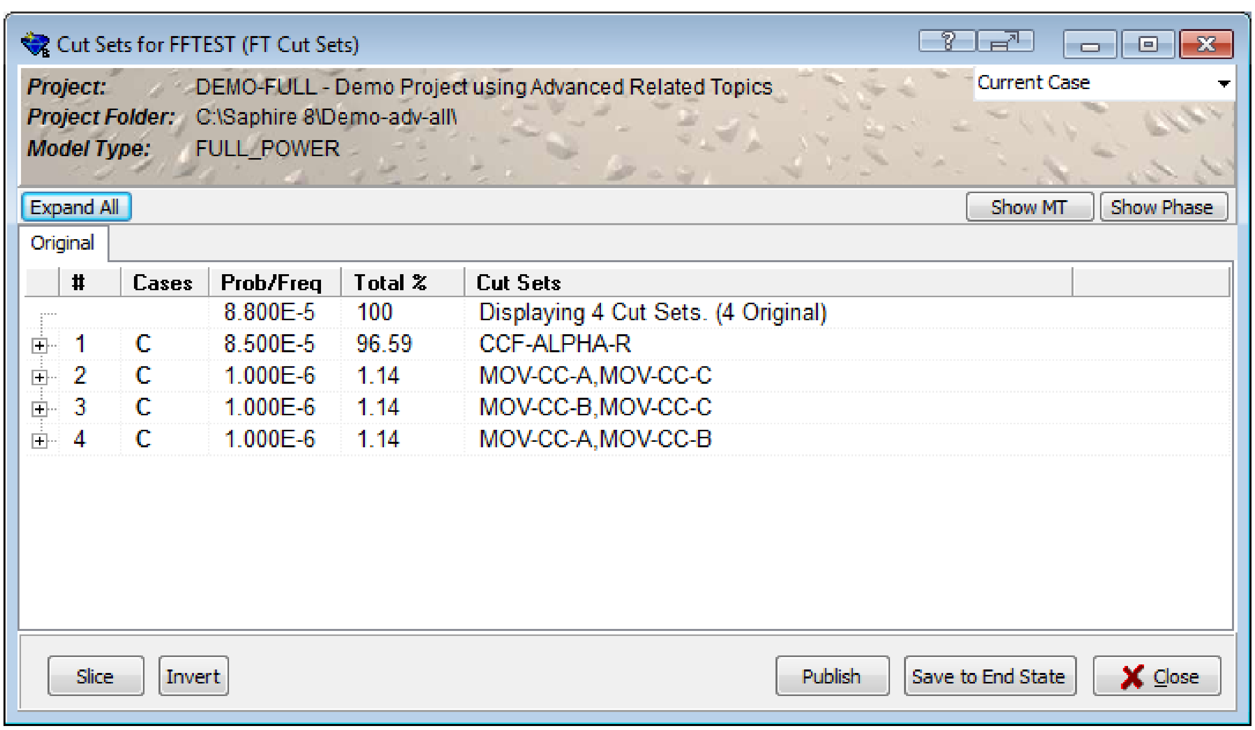
$$P(CCF) = 3 * Q_2 + Q_3 = 3 * 2.50E\text{-}05 + 1.00E\text{-}05 = 8.50E\text{-}05.$$

CCF Sub-elements

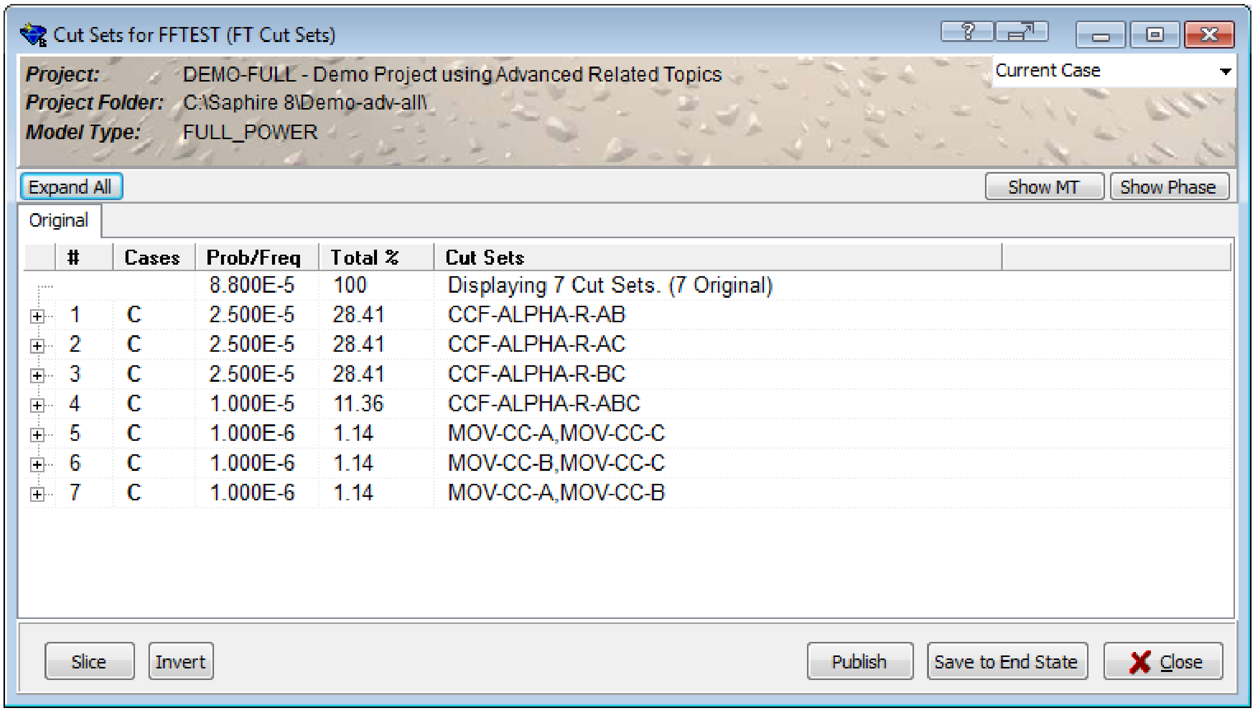
Element #	Terms	Nominal Value
# 1	CCF-ALPHA-R-AB	2.5000E-05
# 2	CCF-ALPHA-R-AC	2.5000E-05
# 3	CCF-ALPHA-R-BC	2.5000E-05
# 4	CCF-ALPHA-R-ABC	1.0000E-05

The CCF Sub-elements box represents the "cut sets" included in the CCF calculation. The first three cut sets represent a CCF of two components, specifically AB or AC or BC. The last line represents CCF of all three components. Their respective probabilities are also listed, for example the probability of seeing A and B fail due to CCF is 2.50E-5. Also note the name of the CCF "events" listed here. For components A and B failing due to CCF, SAPHIRE automatically creates the name "CCF-ALPHA-R-AB" by appending the CCF basic event name with the separator character ("-") and then the identifier for the A and B components (A and B, respectively).

Note that if CCF-ALPHA-R is set to have results "**Rolled up**" then only a basic event with probability of 8.500E-5 will appear in the fault tree cut sets:

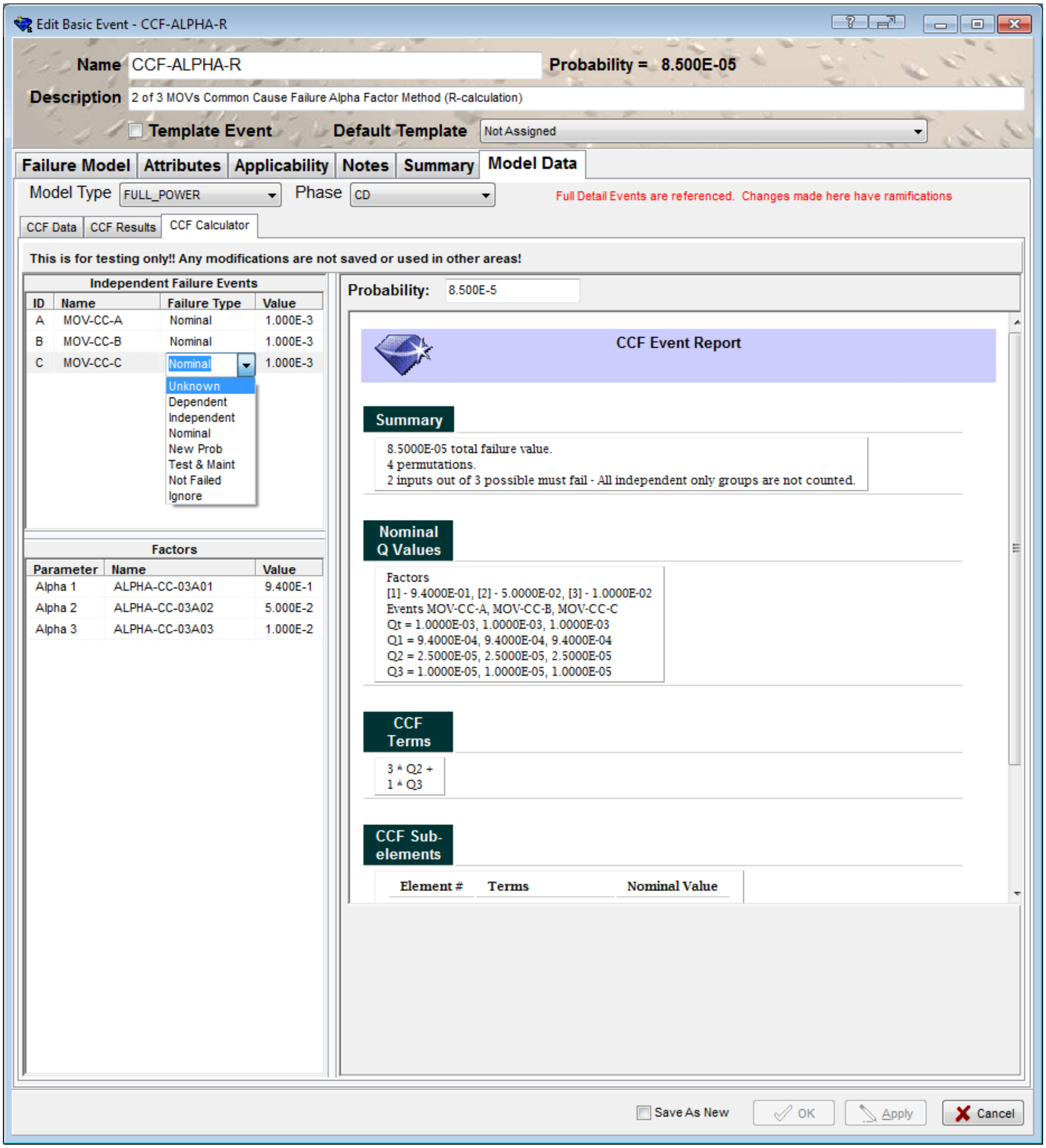


However, if "**Full Detail**" results are selected, the results will show all of the combinations of the CCF terms:

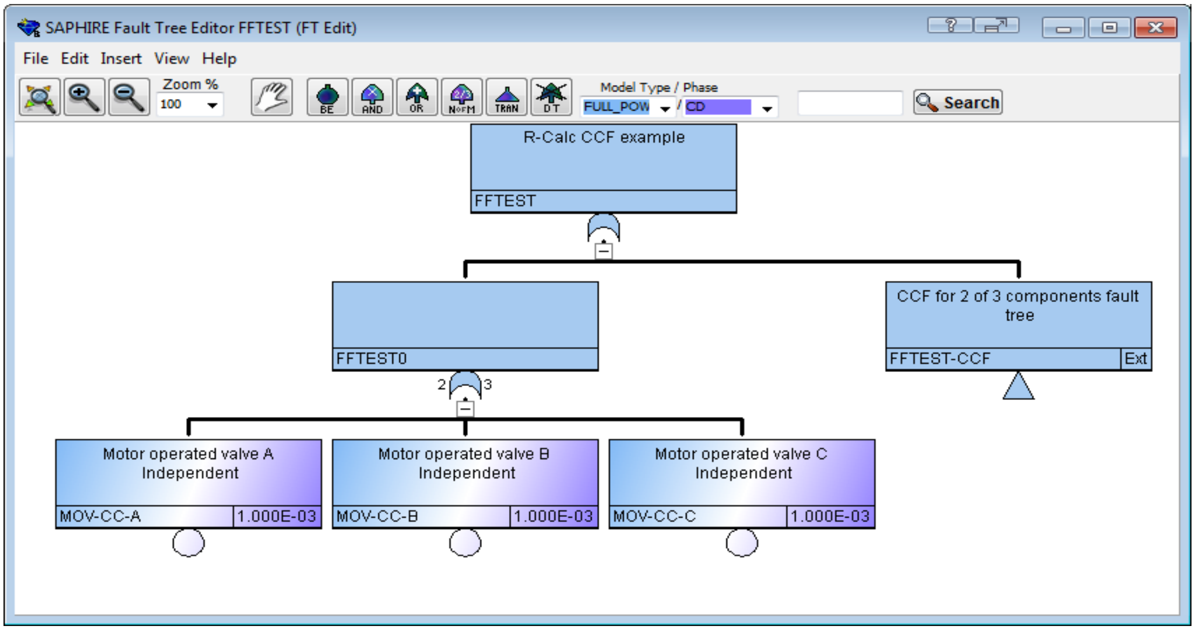


3.3.4.3. RASP CCF Calculator Tool

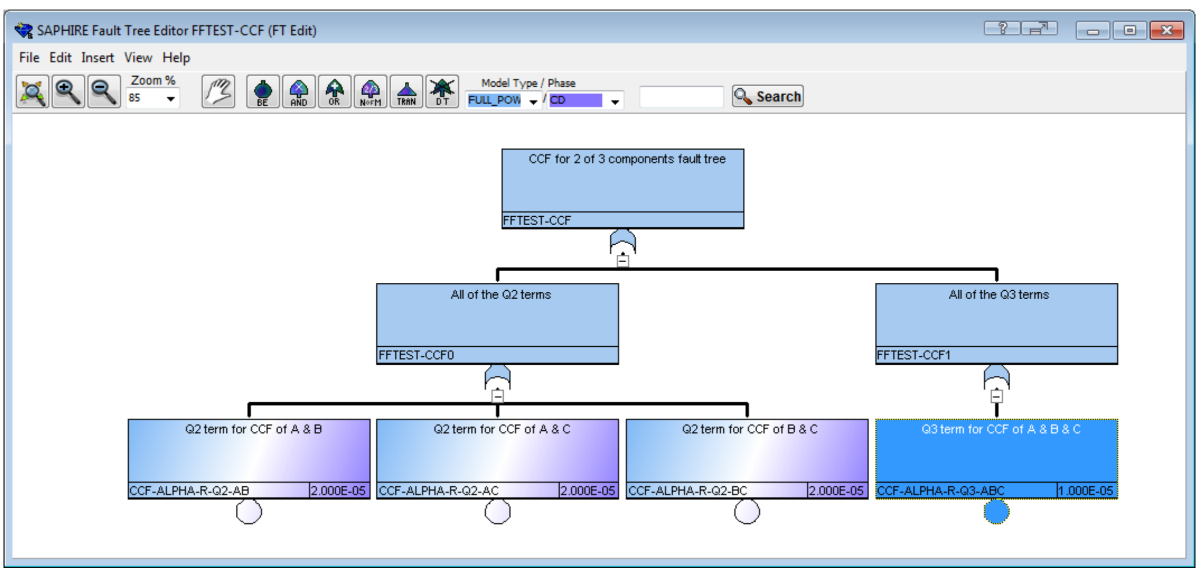
The "**CCF Calculator**" tab allows one to change the make-up of the CCF calculation without saving. The analyst can change the components probability and see the results.



3.3.4.4. RASP CCF Equivalent Fault Tree Logic



By having all of the detail from the CCF, the fault tree calculation is equivalent to having the fault tree logic as shown in the fault trees below:

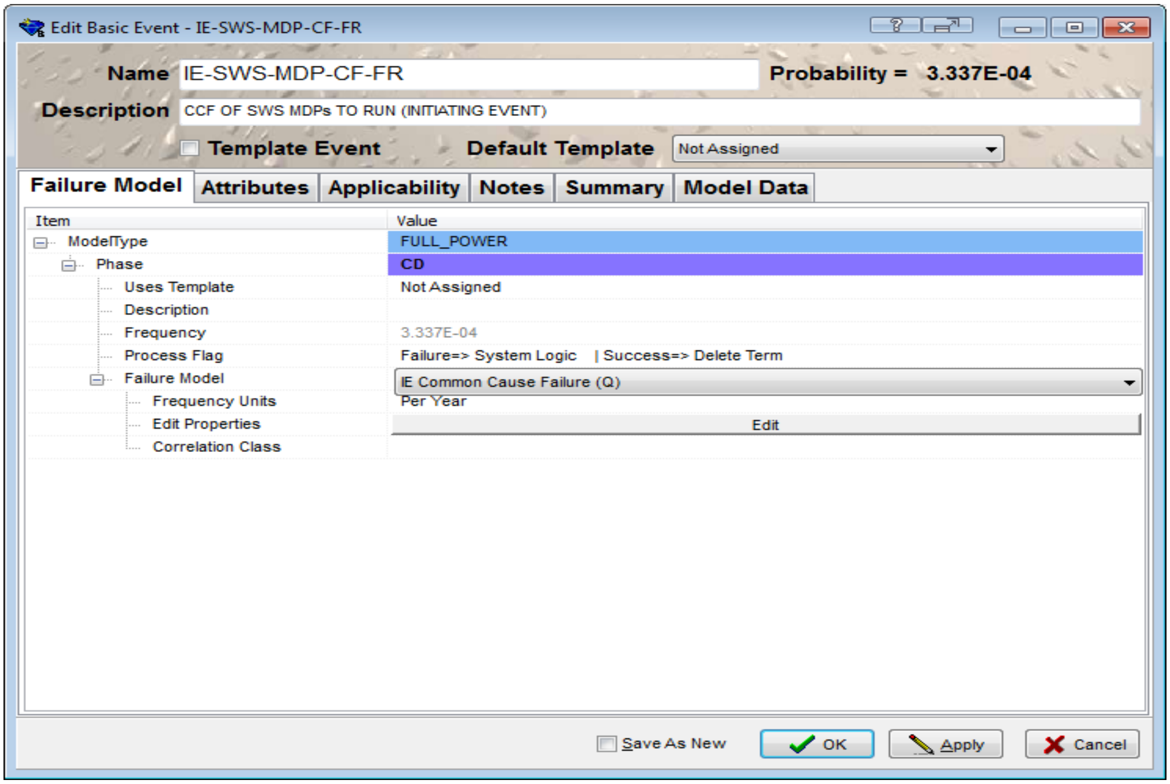


3.3.5. IE Common Cause Failure (Q)

This failure model is used to model common cause failure of multiple trains that are used to calculate an initiating event frequency. This failure model is created the same as the R-calc failure model with the exception that there is no cross-products that get calculated; the independent failure “ANDed” with a CCF event (i.e., Q1 * Q2, since this becomes /yr^2).

3.3.5.1. Entering CCF Data into the Q-failure model

A basic event, IE-SWS-MDPS-CF, which would be part of the fault tree logic model used to calculate an initiating event frequency will be developed. This basic event needs to be edited.



The following discusses the process to set up the Q-type CCF event:

◊ Select the **IE Common Cause Failure (Q)** option in the Failure Model text field.

◊ Click the **Edit** bar to open the Q-calc CCF form for editing.

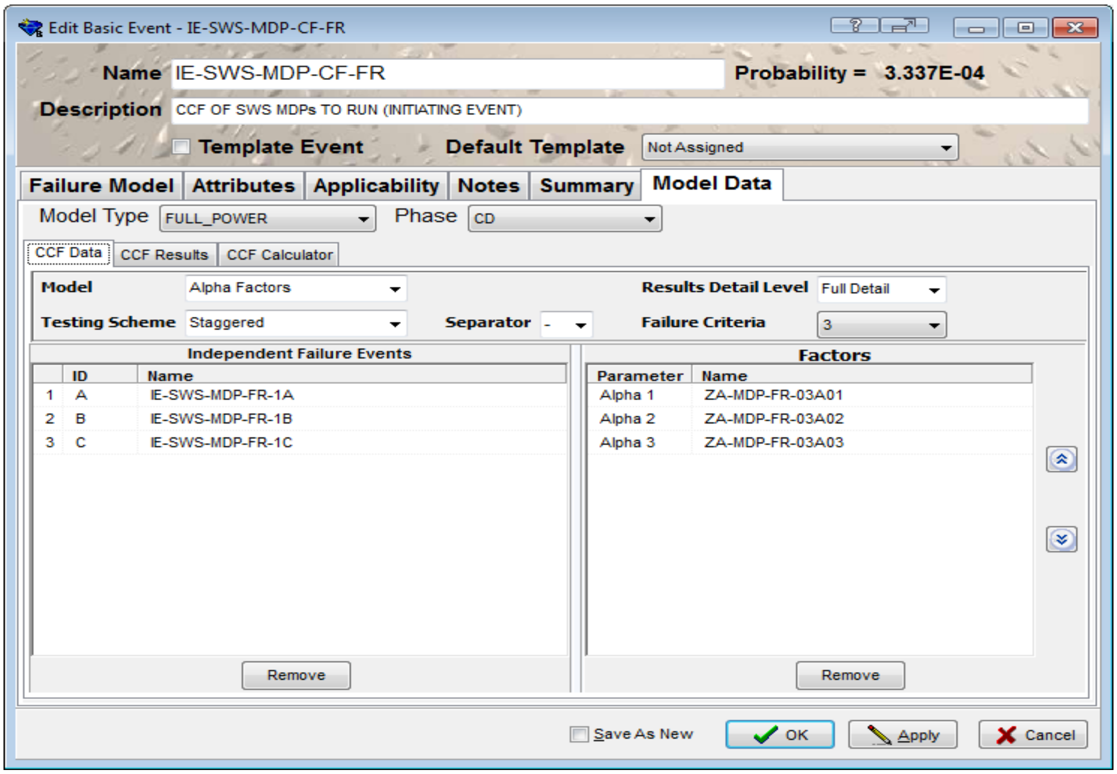
◊ Choose Alpha factors as the **Model** type from the pull-down menu.

◊ Click and drag the desired Basic Events into the **Independent Failure Events** window

◊ Click and drag the Alpha factors, which are located in the Basic Events list into the **Factors** window. The Alpha factors need to be entered based on the number of components in the group (3-train, 4-train, and so on) and placed into the correct parameter field (i.e., Alpha 1 must be alpha factor 1).

◊ Choose the number of failures that would cause a system failure.

◊ The testing scheme can be chosen as Staggered or Non-Staggered



◊ The Separator character is used by SAPHIRE to determine the "auto generated" event names, as we will see when describing the results of the CCF calculation.

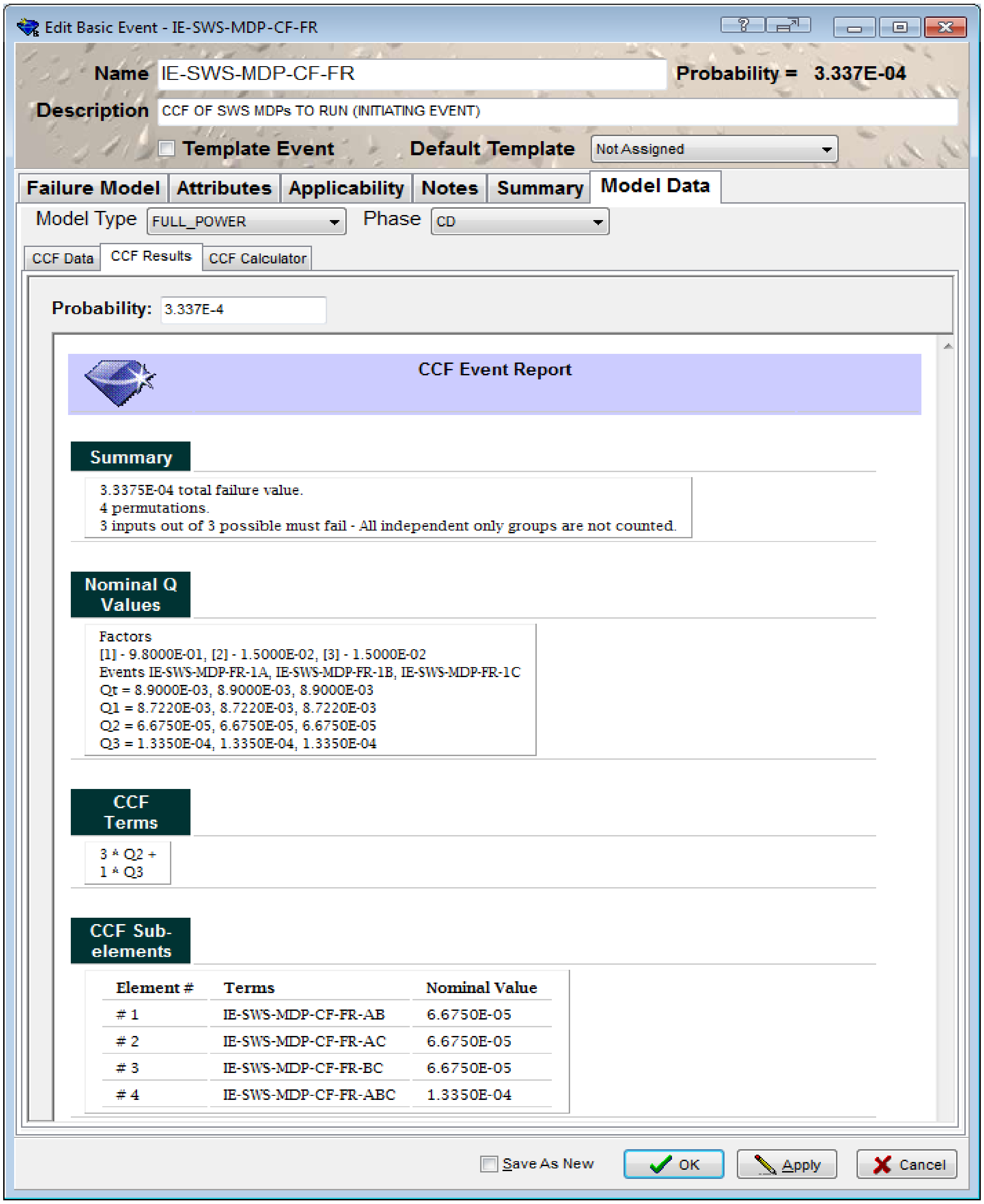
◊ When using the Q-calc option, only choose the Full Detail Level

- **Full Detail** will create a basic event representing each of the CCF terms (Q-terms) depending upon the number of trains. These system generated basic events (i.e., IE-SWS-MDP-CF-FR-ABC) will have to be used in the fault tree logic. The reason that the system generated basic event is required to be used in the fault tree logic is due to the conditional calculation that may be required given an event assessment.

The probability displayed is the overall probability (for a three train system it will be the summation of the Q2 terms and the Q3 term). This is the frequency that will be used for this initiating event CCF basic event.

3.3.5.2. Viewing IE CCF Result Details

To view the results for this example, click on the "**CCF Results**" tab. The detailed results appear as:



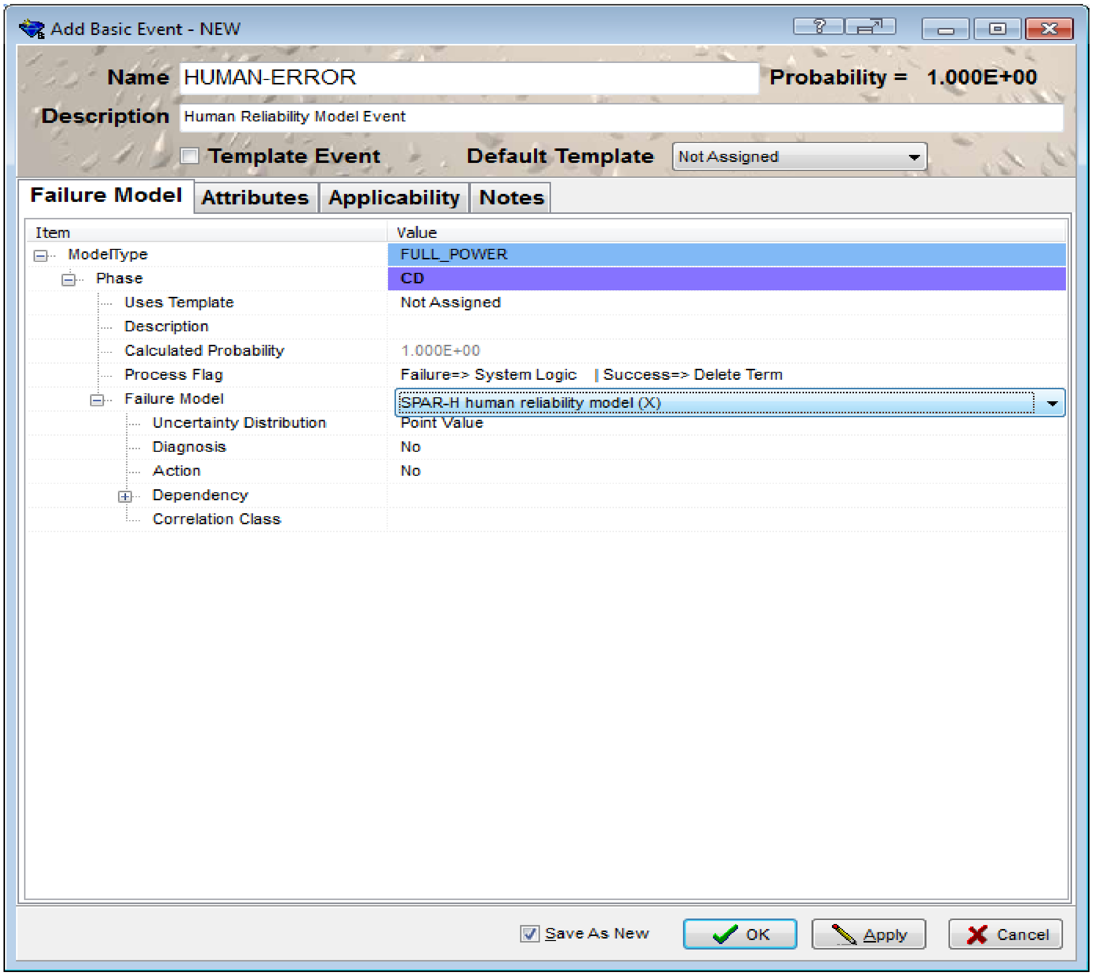
(Notice the CCF sub-elements only lists the Q2 and Q3 terms and no cross-products since the frequency cannot be squared.)

3.4. 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 worksheet that SAPHIRE uses to calculate the HEP. For more information about this HRA, refer to reference 3-1.

- ♦ To access the SPAR-H worksheet, highlight the basic event in the Basic Event list panel, double click the basic event (or right click and select Edit)
- ♦ Select the **SPAR-H human reliability model (X)** in the Value column for the Failure Model in the Basic Event editing window



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

After selection of the SPAR-H failure model, the entries for Diagnosis, Action, and Dependency will become available

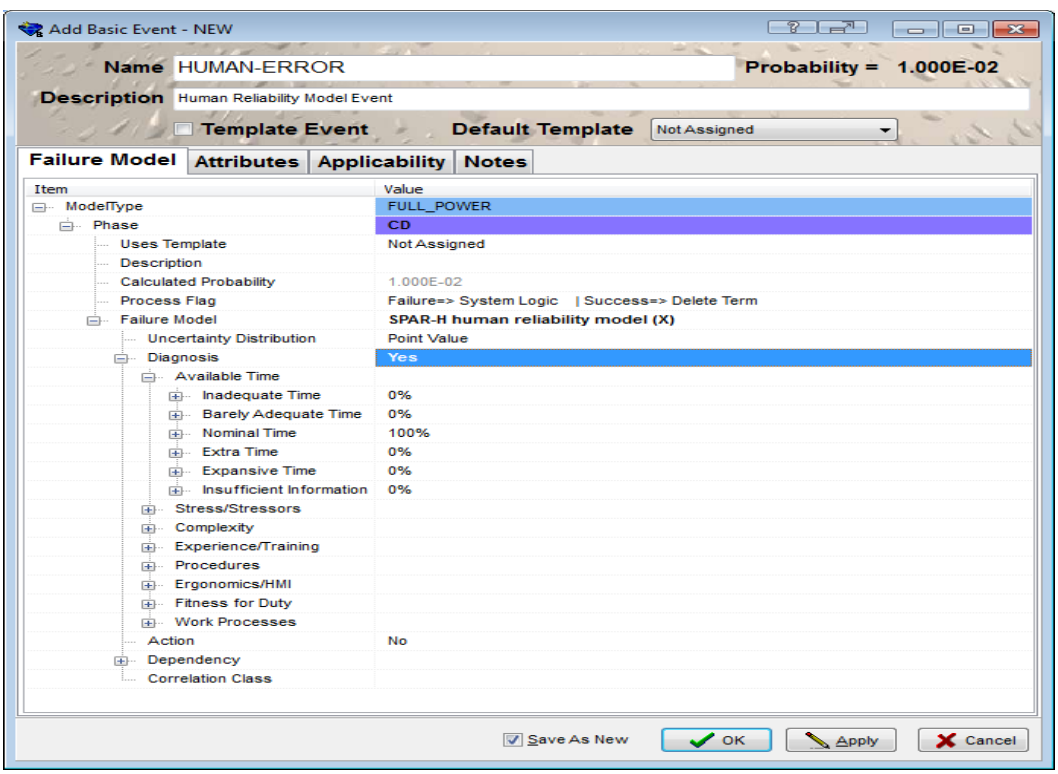
- Diagnosis

  ◊ If the operator is required to perform some type of diagnosis prior to performing an action, select **Yes** in the drop down menu next to **Diagnosis**.

  ◊ 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. A default of 100% is specified for the Nominal time field. 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 a percentage greater than zero is entered into a time field other than the Nominal time field, the form updates Nominal time field to sum the totals to 100%. A Notes field is available for support of entries, if so desired.

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

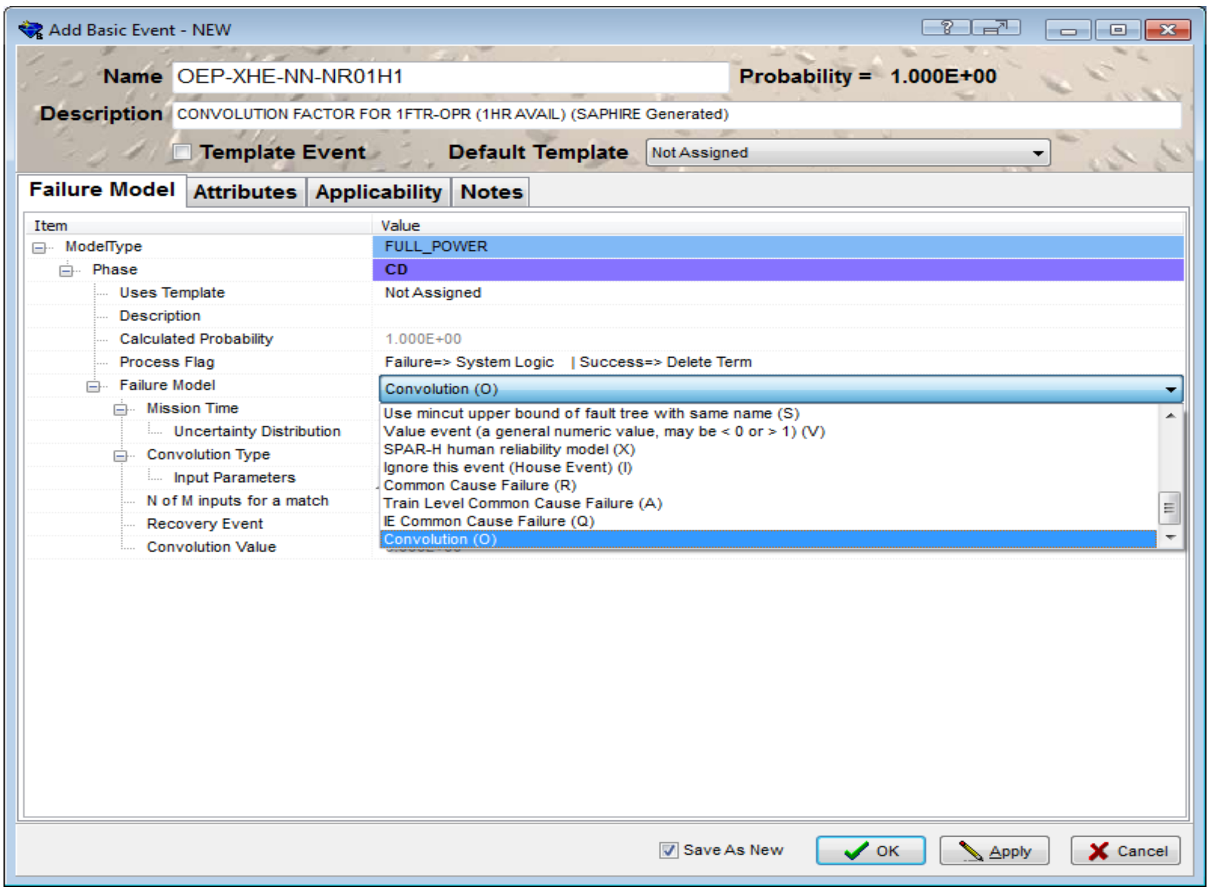


- ♦ Action
  - ◊ If the operator is required to perform some type of action, select **Yes** in the drop down menu next to **Action**.
  - ◊ As with Diagnosis above, each of the shaping factors can be modified individually. These shaping factors are used to modify the nominal probability for diagnosis (1.0E-3 for action).
- ♦ Dependency
  - ◊ Model Dependency
    - Select **Yes** in **Model Dependency** to model the situation when a dependent operator action follows a previous operator action (i.e., more than one operator action failing to perform the function within a sequence).
  - ◊ Different Crew
    - Is the crew performing the action different from the one performing the prior action?
  - ◊ Not Close in Time
    - Is the current action not close in time compared to the prior action?
  - ◊ Different Location
    - Is the current action being performed in a different location from the prior action?
  - ◊ Additional Cues
    - Are additional cues related to the current task available?

3.5. Convolution Basic Events

Convolution basic events are special type of basic events that are used to more accurately represent "time related" events within a cut set. These basic events are more of an adjustment factor for accident sequence cut sets when time related basic events fail. All PRAs assume that failure of an operating component occurs at time equal to zero. This special basic event is designed to convolve multiple time related basic events into an adjustment factor to account for when these time related basic events can fail and recovery should be applied (see Convolution Event User Manual for more information).

- ♦ To create a convolution basic event, in the Basic Events list panel, double-click on "New basic event". This opens the Add Basic Event form.



- ♦ Click on the text (Value column) to the right of Failure Model to activate the drop-down list.

- ♦ Choose "Convolution (O)".

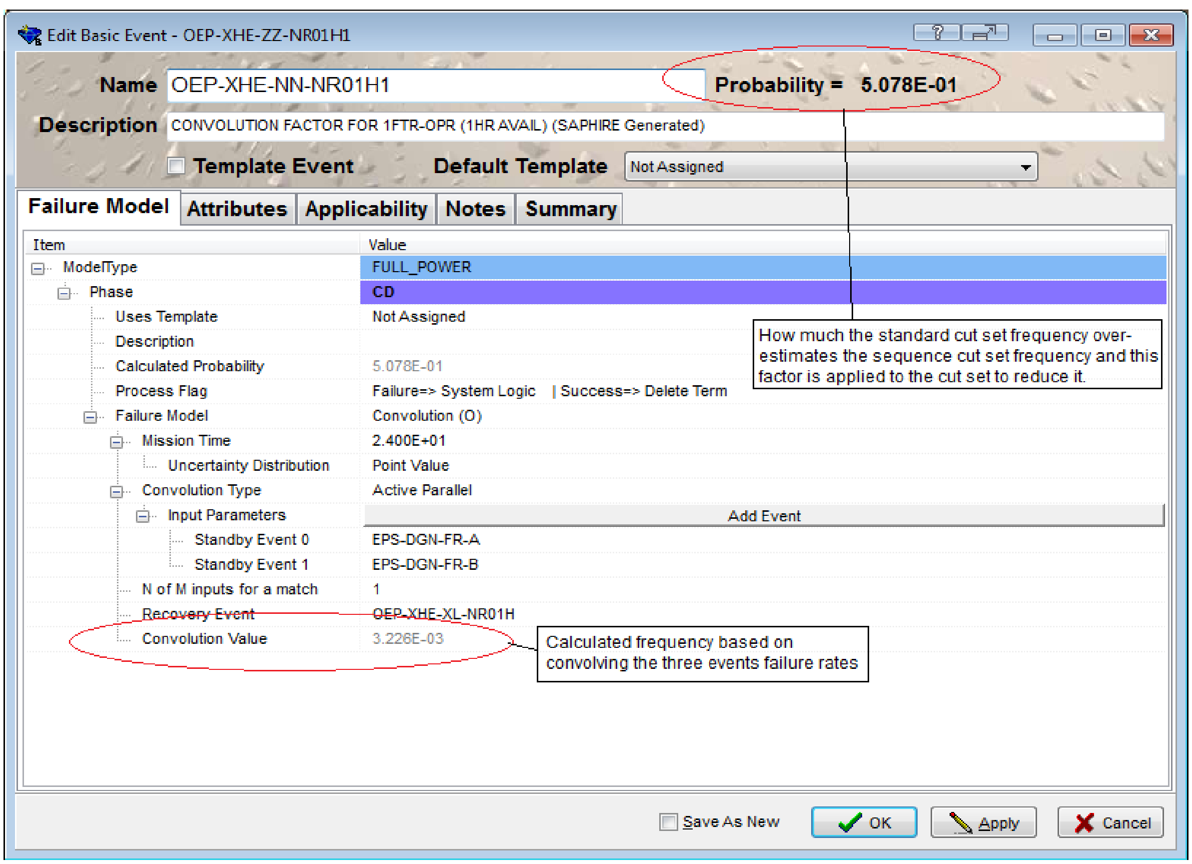
◊ Select the mission time that the cut set will require to define failure (i.e., recovery time based on core uncover or battery depletion time)

◊ Click the AddEvent tab to specify the time related basic events that need to be convolved together (fill in the number of Standby Events that will be found in a single cut set) (i.e., EPS-DGN-FR-A, EPS-DGN-FR-B).

◊ Select how the inputs should be found in the cut sets (N of M) (e.g., if only one of the basic events are to be convolved per cut set use 1; EPS-DGN-FR-A only in a cut set or EPS-DGN-FR-B).

◊ Select the recovery event that will be part of the convolution (i.e., the recovery basic event based on the sequence cut set). This is the basic event related to the sequence time to core uncovery (i.e., 1 hour, 2 hours)

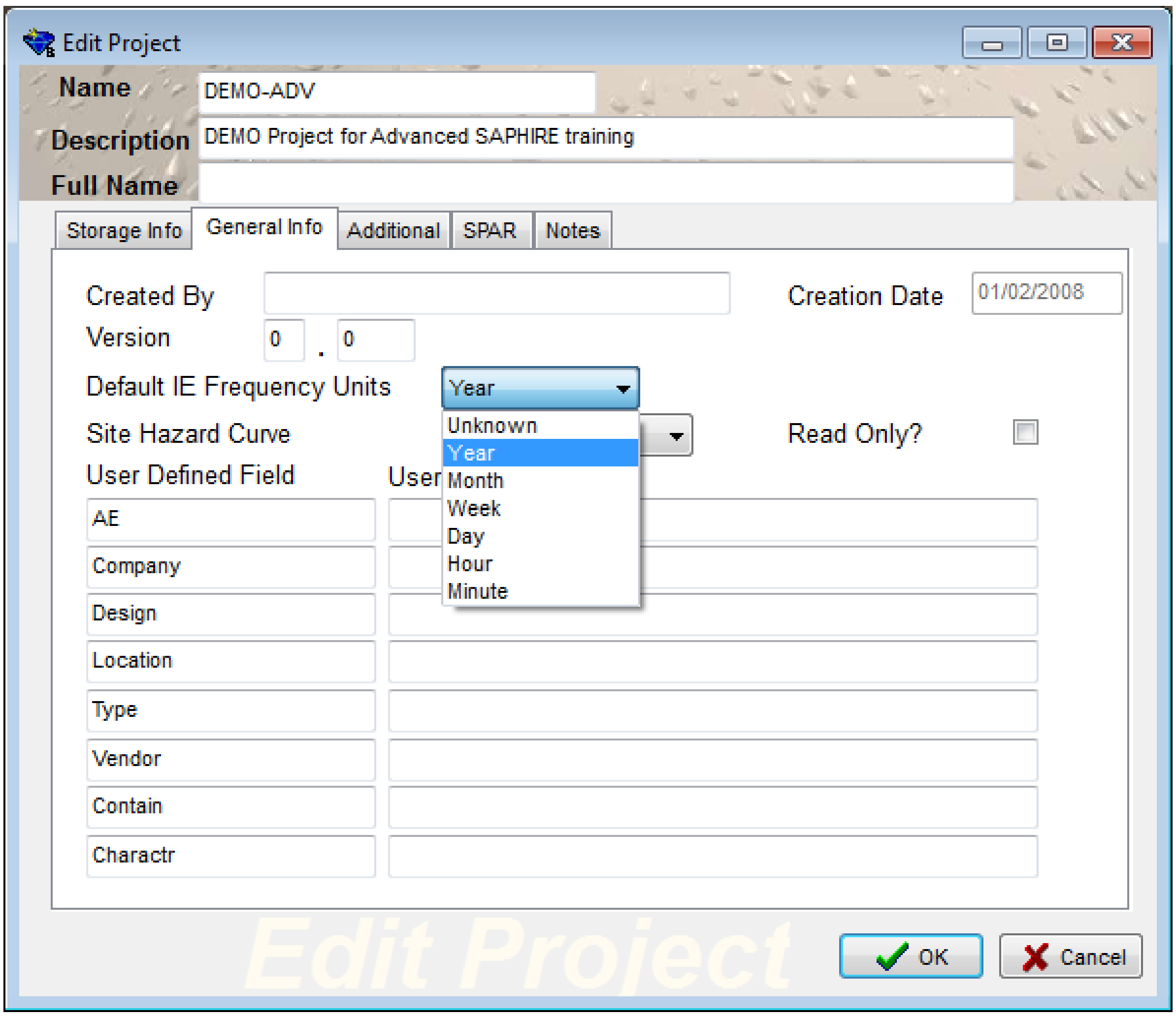
◊ SAPHIRE will calculate what the cut set frequency should be and it what the cut set frequency is based on t=0 and the mission time of the diesel generators to run of 24 hours and divides the two together to calculate the convolution factor. This is how much the sequence should be reduced.



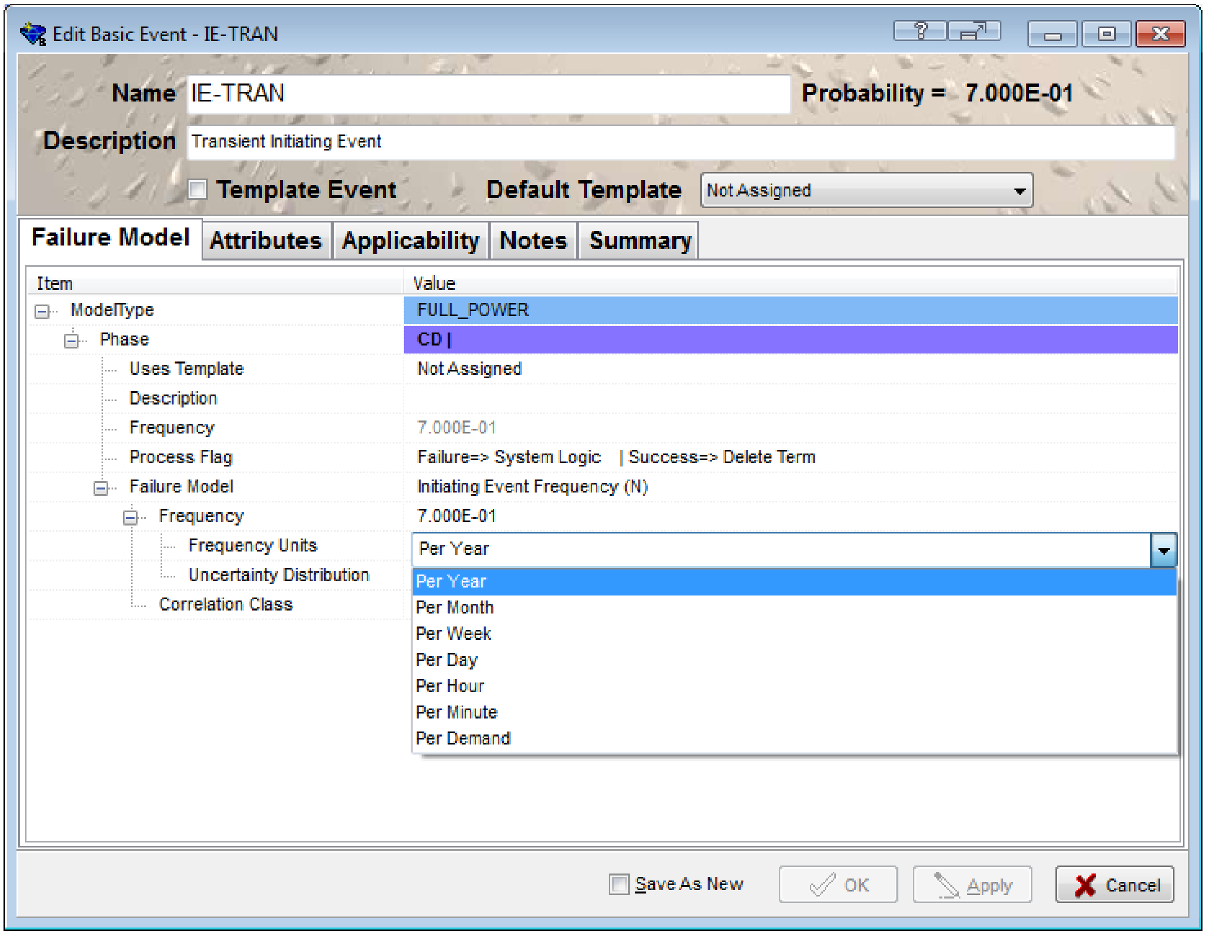
◊ Post-Processing rules are then used to apply this basic event

3.6. Base Units used in Data

SAPHIRE 8 is designed to handle different frequency units and make proper conversions when necessary. The default frequency units for Initiating Events is set up in the New Project form and can be changed for an existing project by accessing the **Project ➔ Modify** option.



In the Basic Event Failure Model tab, when an Initiating Event is selected, a Frequency Units textbox becomes available. If nothing is entered in the textbox, the default value is used.



- The different frequency units that can be defined in SAPHIRE are:
    - ◊ Unknown
    - ◊ per year
    - ◊ per month
    - ◊ per week
    - ◊ per day
    - ◊ per hour
    - ◊ per minute
    - ◊ per demand

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

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

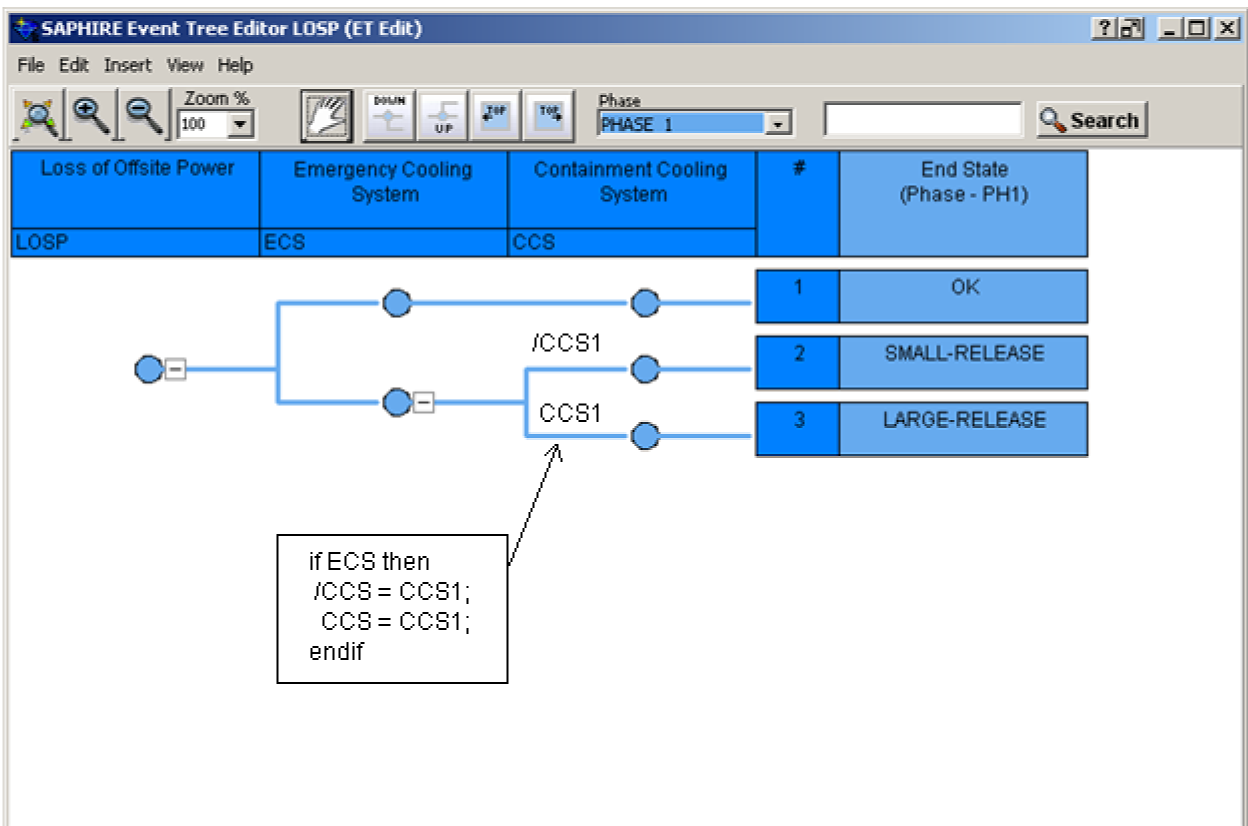
- ♦ If the “Unknown” frequency units are specified in the **Project → Modify**, then the units specified for the initiating events will be used. (One must be careful, since different units can be specified and the overall summation of the sequences won’t be correct).

3.7. Reference

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

| 4 | EVENT TREE LINKAGE RULES EDITOR

Section 4 introduces the rule editor for the **Edit Linkage Rules** editor used to link event tree sequences. This 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.

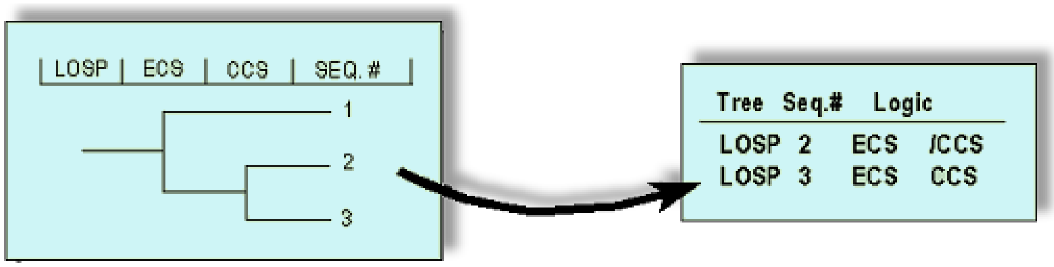


4.1. Linking Event Trees

This section introduces the edit linkage rules editor used to link event tree sequences. This 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.

4.1.1. Linking

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



4.1.2. Menus and Options for Linking Event Tree Sequences

- Linking is performed automatically when the event tree is saved
- Or, it can be initiated by highlighting the event tree, right clicking the mouse and selecting “**Link**”.

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 "Edit Linkage Rules" Rule Editor

The link event tree rule editor provided in SAPHIRE allows rules to be written that are used when sequence logic is generated. To access the rule editor, highlight the event tree in the Event Tree list panel, right click the mouse and select **Edit Linkage Rules** to open the linkage rules editor.

- 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 other rule editors in SAPHIRE have different functions.

- Section 5 describes the cut set rules editor which sets up *post-processing rules* that are used to modify existing cut sets.
- Section 6 describes the *partitioning rules* that are used to bin cut sets into end states 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.

4.2.1. “Link Event Trees" Rules Nomenclature and Structure

This rule editor is used when the “Link Trees” operation is performed when the event tree is saved from the event tree editor 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
;	End of replacement line	??	Wildcard of length in string of “?”, (in this case 2)

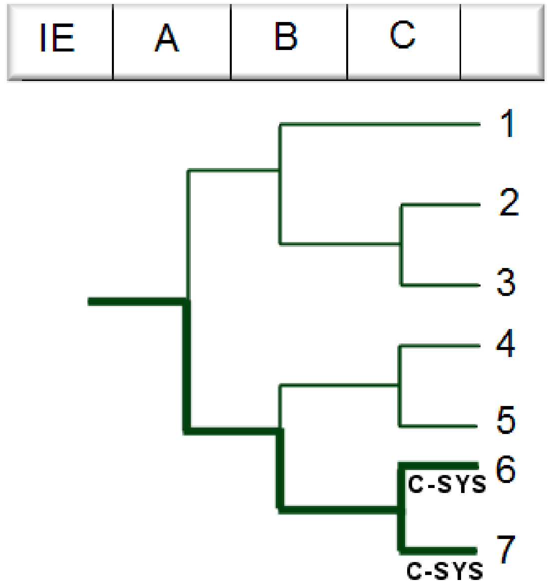
Search Criteria

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
/A	Success of top event or fault tree A (**/** 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.

4.2.2. Link Event Trees Rules Examples

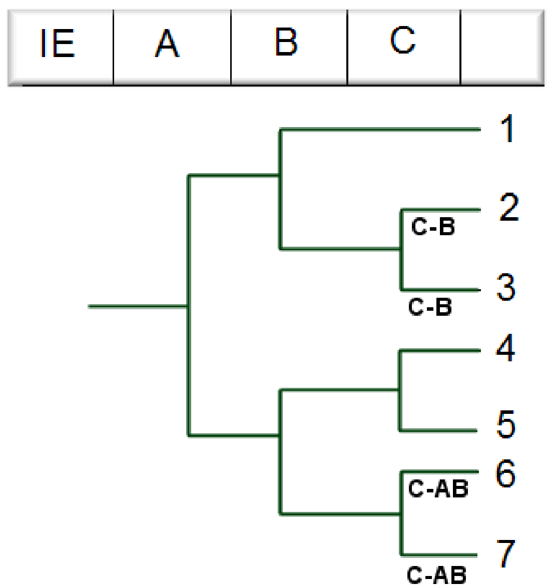
Linking Rule Structure (Example 1 – If -Then)



| 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

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

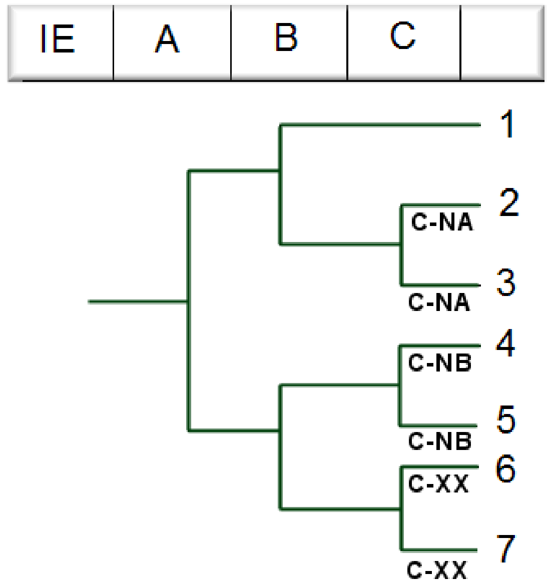


| The "if-then-elsif" Structure:
| This rule replaces C with C-AB if
| A and B are failed, and replaces C
| with C-B if only B is failed.

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)

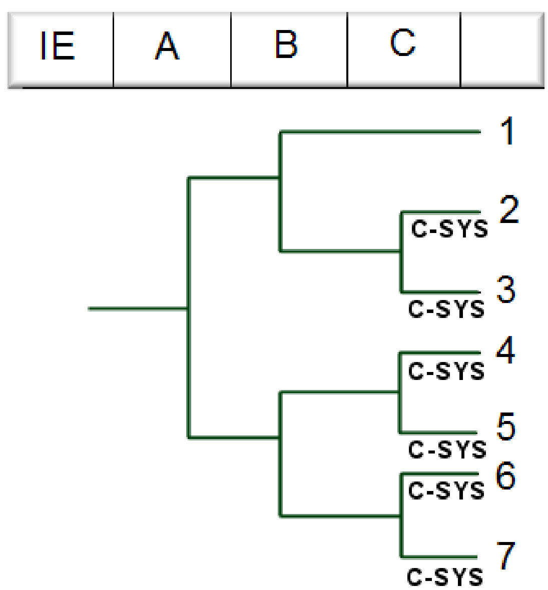
```
| The "if-then-elsif-else" Structure:
| This rule:
|   Replaces C with C-NA when A is
|   successful
|   Replaces C with C-NB if B is
|   successful
|   Replaces C with C-XX in any other
|   case
|
if /A  then
  /C = C-NA;
   C = C-NA;
elsif /B then
  /C = C-NB;
   C = C-NB;
else
  /C = C-XX;
  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

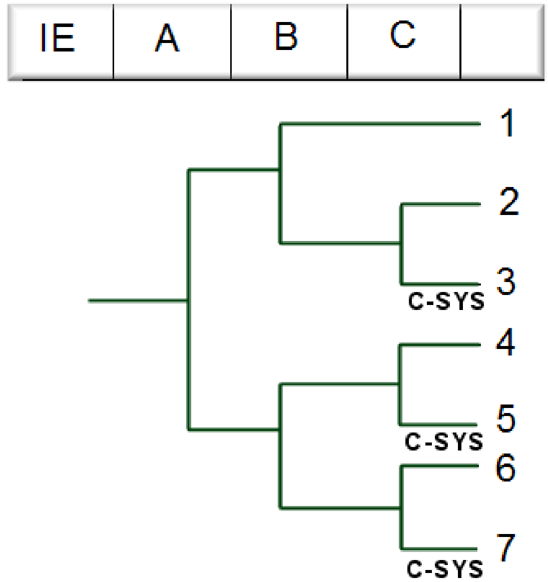


Linking Rule Structure (Example 5 – If –Then with wildcards)

| 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.



IMPORTANT NOTES AND REMINDERS:

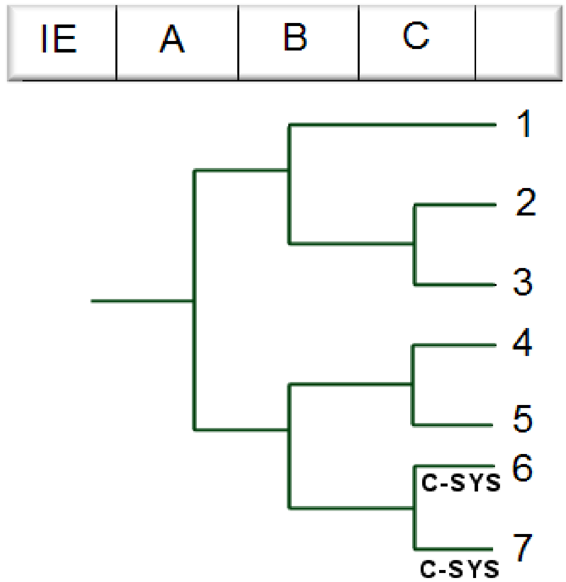
- Each replacement line must end with a semicolon.
- There is no limit to the number of replacement lines that can be used in a rule.
- In the **ELSIF** rule structure, only the first substitution that applies for every applicable branch is made. Subsequent possible substitutions are ignored.
- In fact, the "Event Tree Linking" rules as a whole works 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. Examples are 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
```

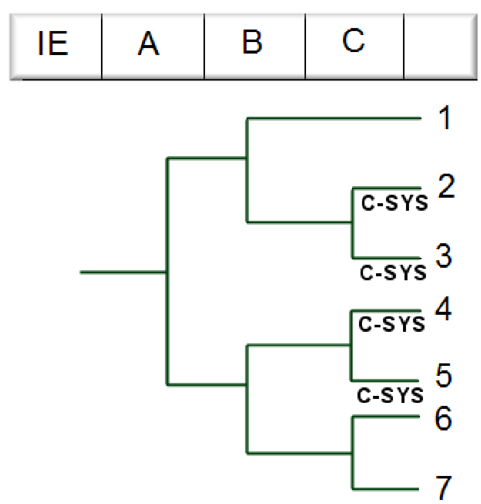


If you are creating a rule where the events in the macro do not occur, use the ~ (i.e., not present) symbol.

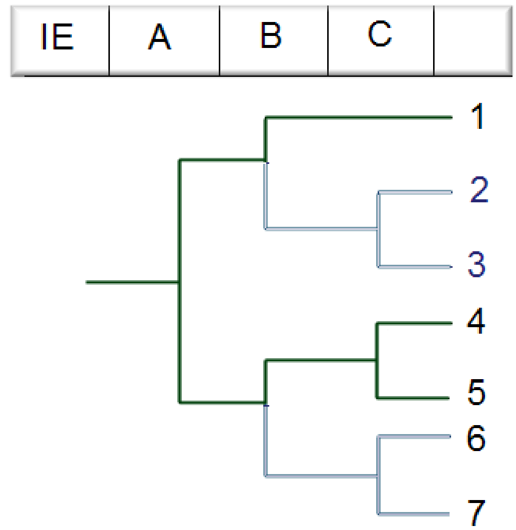
If looking for success events, do not "complement" the macro. Instead, complement the events of interest. For example, if looking for success of A, use A-MACRO = /A;. Do not try A-MACRO = A; if /A-MACRO then…

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.

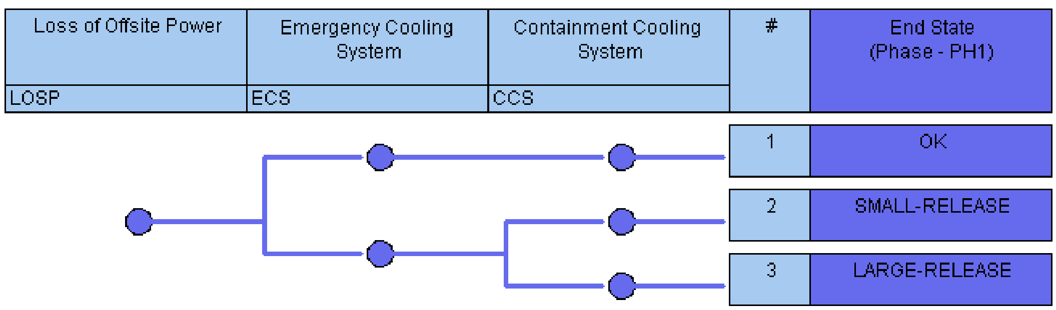
A new key word was added to the same thing but more straightforward. This key word is SkipSequence and performs the same function as above using the rule shown below:

```
if B then
  SkipSequence;
endif
```

After applying the rule only sequences 1, 4, and 5 get generated.

Linking Rule Structure (Example 9 – changing end states)

| The "End state" keyword:
| This keyword provides the ability to change the end state names
| created on the event tree logic to a different end state (or add an already
| created end state for the sequence cut sets that meet the specified search criteria).
| The new end state must exist in the project.



if /CCS **then**
  eventree(LOSP) = Endstate(S-R);
endif

This rule will change the end state for sequence 2 from SMALL-RELEASE to S-R when the sequence logic is created.

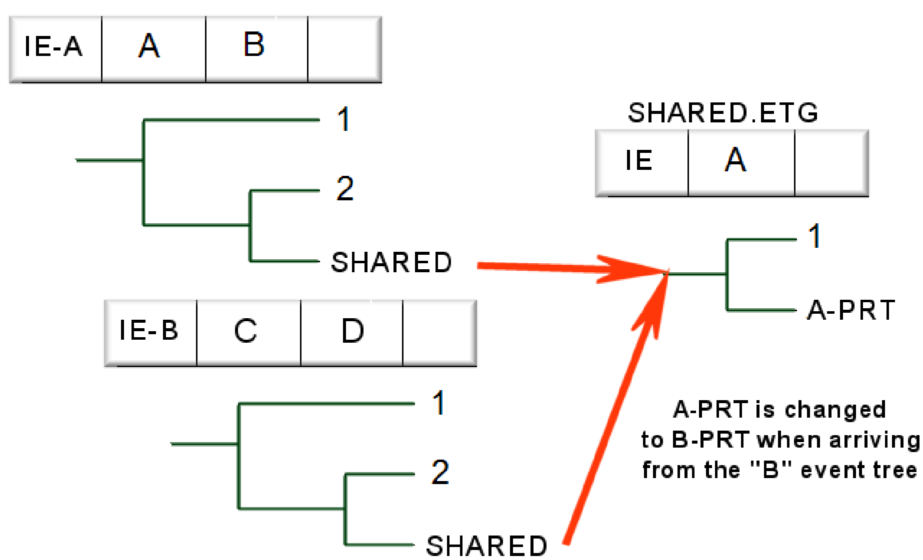
4.3. Changing Transfers Trees using Link Rules

4.3.1. Using Eventree() option

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 10 – 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
```



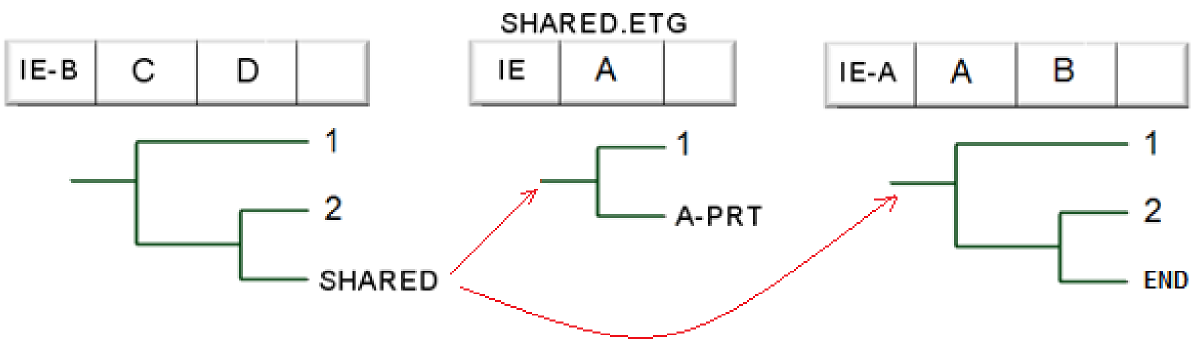
4.3.2. Using SeqTransfer() option

The following discusses another option to control the transfer process from one event tree to a sub tree. This ability brings in the use of a new keyword, **SeqTransfer()**. See the example below.

Linking Rule Structure (Example 11 – changing the transfer tree via SeqTransfer)

|
| This rule is for a transfer tree named SHARED.
| The SHARED event tree contains specific logic that may not be required under certain
| conditions; therefore, the event tree IE-B can change the transfer from SHARED to
| IE-A under that condition. This option will change the transfer sub-event tree from
| SHARED to IE-A and that is the sequence logic that will be generated.
|
|

if C * D **then**
  SeqTransfer(SHARED) = eventree(IE-A);
endif



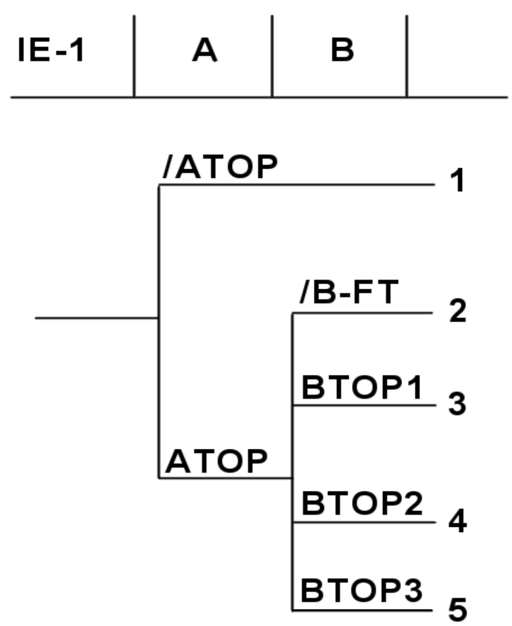
The resultant sequence logic would be: [IE-B * C * D * A * B]

4.4. Rules for Binary and Multiple-Split Branches

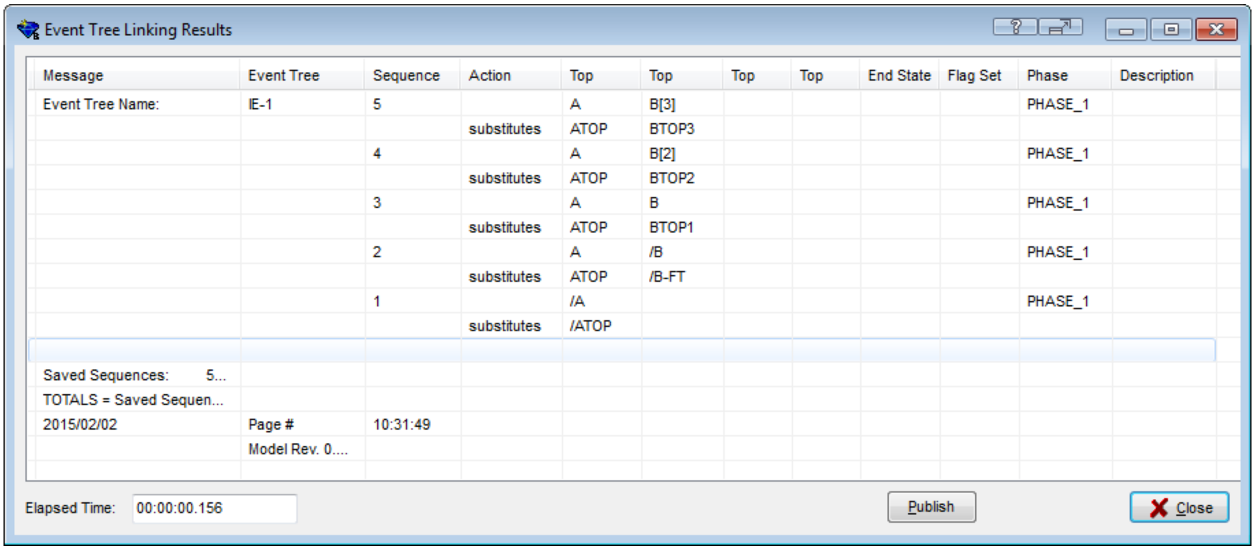
- Event tree branches are normally binary (one up, indicating success, and one down, indicating failure). But, in general, there may be more than two “splits” underneath a single top event.
- SAPHIRE addresses multiple branches by way of the event tree link rules. Several important modeling conventions 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 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
```



If you print a logic report to the screen during the link process (Right Click on Event Tree **IE-1** in list window select **Link,** and then check the **"Create Report"** box before clicking on **OK**) it will look like:



Event Tree Linking Results

Message	Event Tree	Sequence	Action	Top	Top	Top	Top	End State	Flag Set	Phase	Description
Event Tree Name:	IE-1	5		A	B[3]					PHASE_1	
			substitutes	ATOP	BTOP3						
		4		A	B[2]					PHASE_1	
			substitutes	ATOP	BTOP2						
		3		A	B					PHASE_1	
			substitutes	ATOP	BTOP1						
		2		A	/B					PHASE_1	
			substitutes	ATOP	/B-FT						
		1		/A						PHASE_1	
			substitutes	/ATOP							
Saved Sequences: 5...											
TOTALS = Saved Sequen...											
2015/02/02	Page #	10:31:49									
	Model Rev. 0....										

Elapsed Time: 00:00:00.156 Publish Close

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.

4.5. 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 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	Definition	Example Usage
~	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.
/	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 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

Keyword or symbol	Definition	Example Usage

endstate	Keyword to assign an end state to a sequence (based upon sequence logic). [This can also change the existing end state to a new one.]	If “search criteria” then
		eventree(ET-NAME) = endstate(ES-NAME);
		endif

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

SeqTransfer( )	Keyword to indicate a change in the sequence transfer name.	if "search criteria" then
		SeqTransfer(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

SkipSequence
SkipSystem
	Keywords with the same function as Skip( ), however it provides further delineation between sequence and system with the use of categories.	if “search criteria” then
		SkipSequence;
		endif

End_Rule_Section	Provides a break from one rule group to the next rule group in layered rules.	First set of rules **End_Rule_Section** Second set of rules.

LerfFactor	Keyword to assign LERF factors to sequences that meet the search criteria. These factors are used for SPAR models when performing SDP analyses.	if init(event tree) then
		lerffactor = 1.0; |numerical value between 0 < 1
		elsif RPS then
		endif

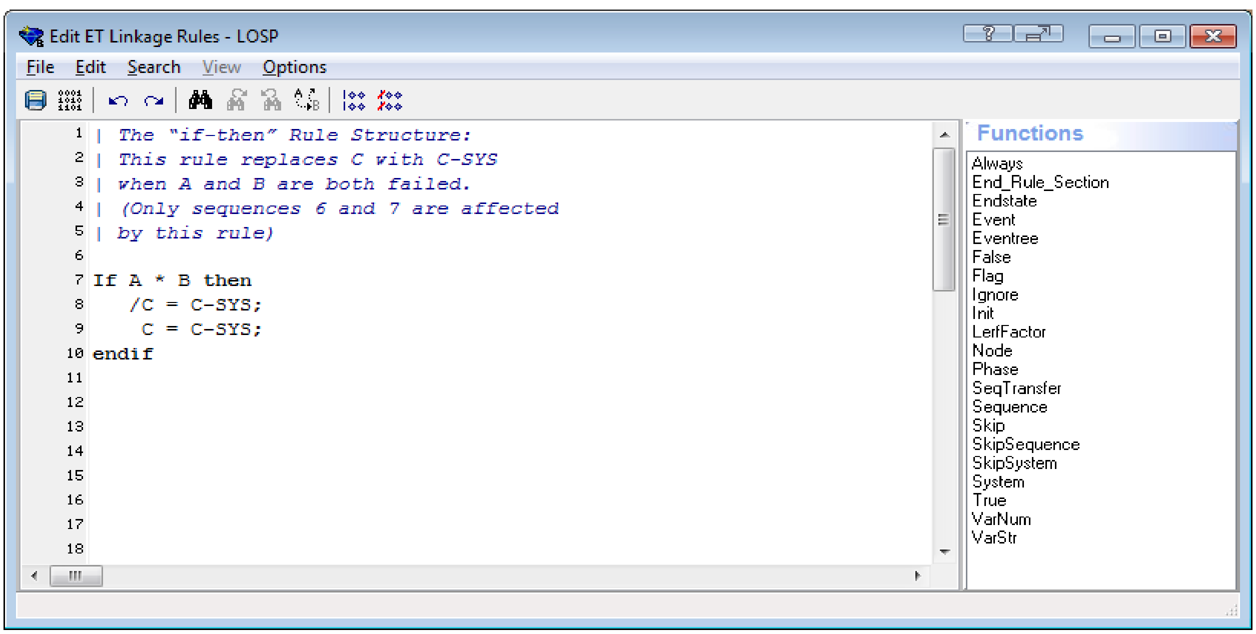
[ ]	Keyword to indicate the number of the event tree branch for multiple-split branch points. The first branch under the top branch is designated as 1. The second is designated as 2, etc.	if "search criteria" then
		/ET-FT = NEW-TREE-NAME1;
		ET-FT[1] = NEW-TREE-NAME2;
		ET-FT[2] = NEW-TREE-NAME3;
	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

Keyword or symbol	Definition	Example Usage

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 cannot define a macro in event tree
		|"A" and expect to use it in event tree "B."

4.6. "Edit Linkage Rules" Event Tree Editor

♦ To use the event tree linkage rule editor, highlight the event tree, right click the mouse and select the "Edit Linkage Rules" editor.



Rules Editor Toolbar:

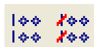
 Save

 Compile

 Undo or Redo action

 Find, Find Next, and Find Previous

 Replace

 Comment highlighted block, Un-Comment highlighted block

Rules Editor Menu Bar:

File – The file drop down menu allows the user to compile, import or export text, and save or both save and exit the rule editor. A check of the script status is performed on an exit of the rule editor. If there is a problem found (i.e. script is not compiled or has errors in the compile) the user is prompted to optionally fix the problem without exiting the editor.

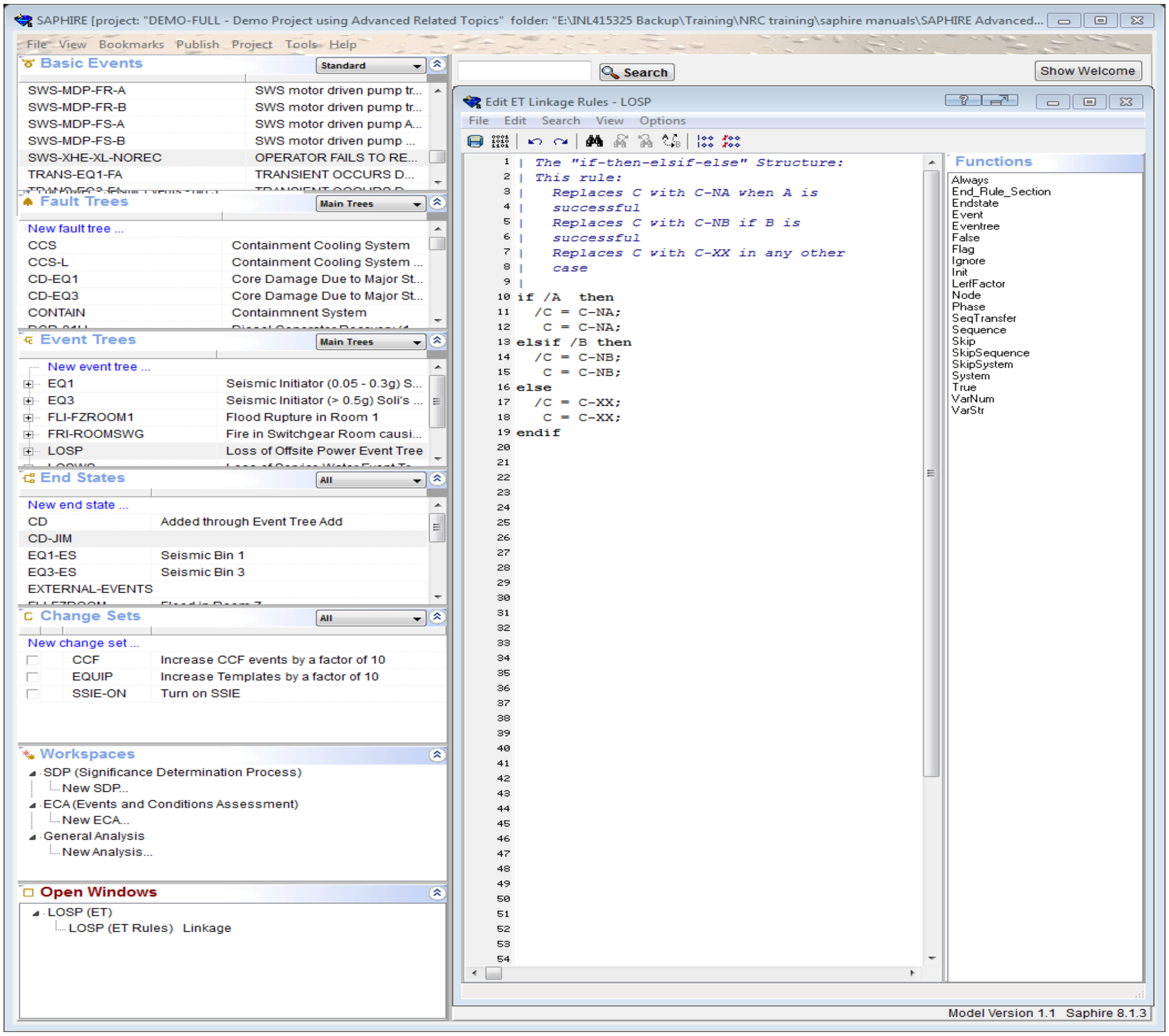
Edit – The edit drop down menu allows the user to undo, redo, cut, copy, paste insert indents or spaces, comment or uncomment any portions of the existing rules,

Search – The search drop down menu allows the user the ability to search the rules, find the next or previous, or replaces character strings.

View – The view drop down menu is used in concurrence with **Options → Show Item Lists** and will show or hide items in the list pane opened to the left of the editing pane.

Options – The options drop down menu controls what panes are visible (Show Function List and Show Item List), what is displayed in the list pane (Header and Description), and can save the current configuration as the default configuration for the rules editors in the project.

The following image shows the rules editor including the list pane.



♦ Editing Pane

◊ Items can be dragged and dropped to the editing pane from the main list panes outside of the editor, the list pane within the editor, and the function pane within the editor.

◊ Items can be inserted where the cursor is in the editing pane by double-clicking on a list item or function within the editor.

- Double-clicking on a main list item outside the editor opens that list item for editing.

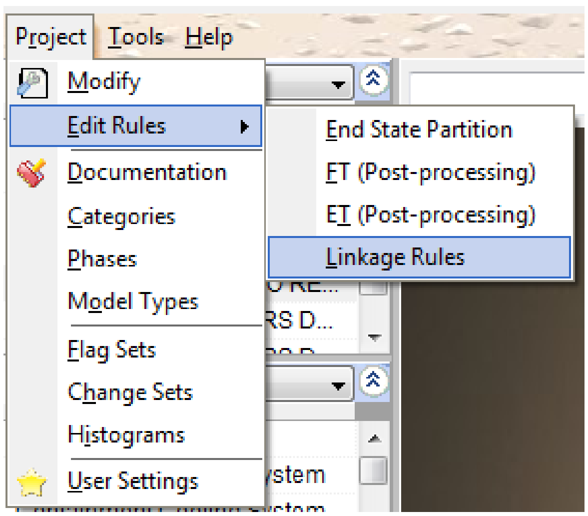
Note:

Functions are **not** case sensitive, (i.e. AddEvent is the same as addevent) but variables **are** case sensitive (DG-A is different than DG-a).

4.7. “Project Linkage Rules” Editor

A new rule editor was added to give the analyst the ability to create linkage rules that will be applied to all event trees. These rules get applied when the event tree accident sequences are generated via “link” or saving the event tree(s).

- To use the “Project” linkage rule editor, under the **Project → Edit Rules** option, select “Linkage Rules” editor.

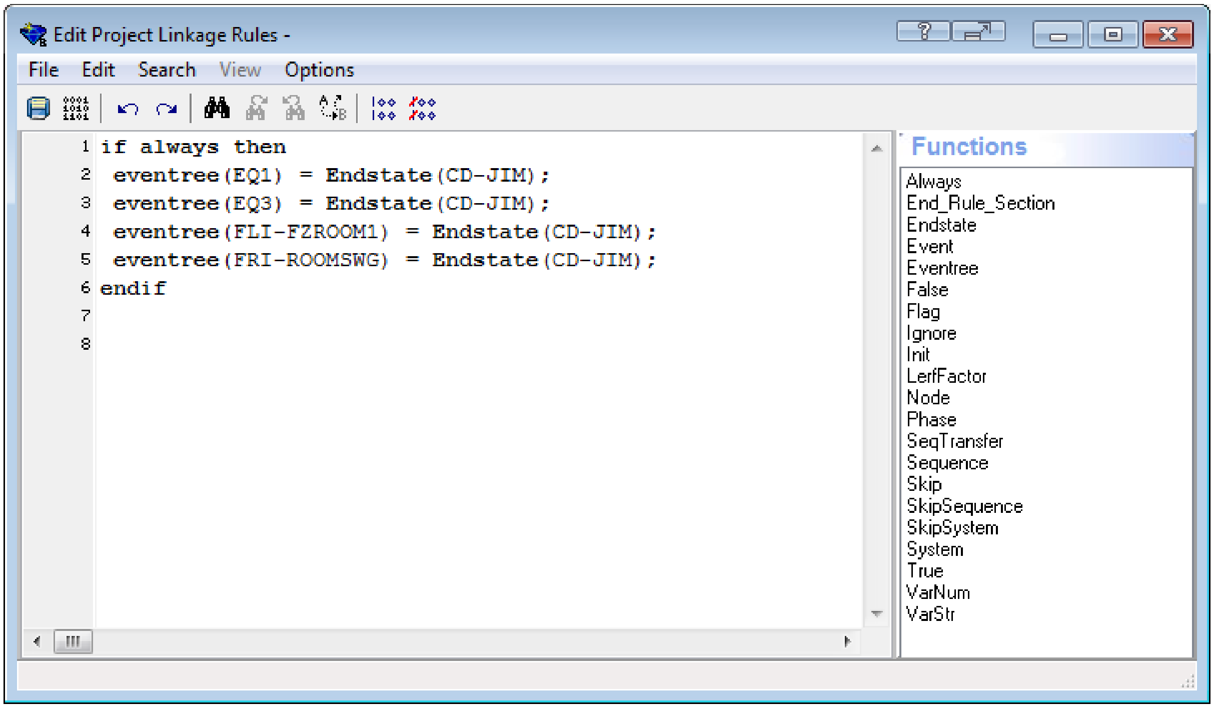


- The rules are created the same as the Link Event Tree rule.

  ◊ Items can be dragged and dropped to the editing pane from the main list panes outside of the editor, the list pane within the editor, and the function pane within the editor.

  ◊ Items can be inserted where the cursor is in the editing pane by double-clicking on a list item or function within the editor.

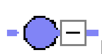
    - Double-clicking on a main list item outside the editor opens that list item for editing.



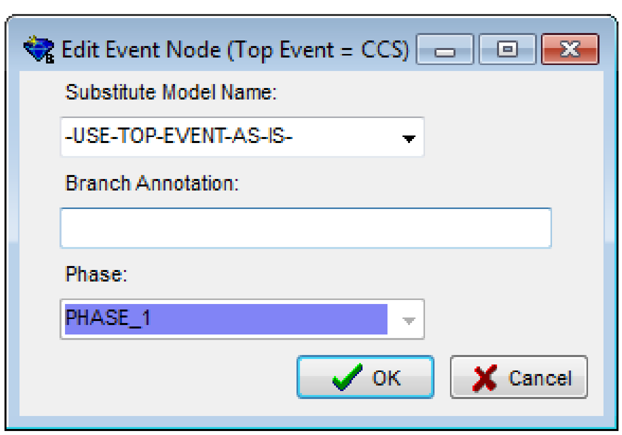
4.8. "Top Event Substitution" Event Tree Graphic

Default top events can be replaced directly from the event tree graphics. This new option creates the linking rules discussed above behind the scenes based on the substitution specified.

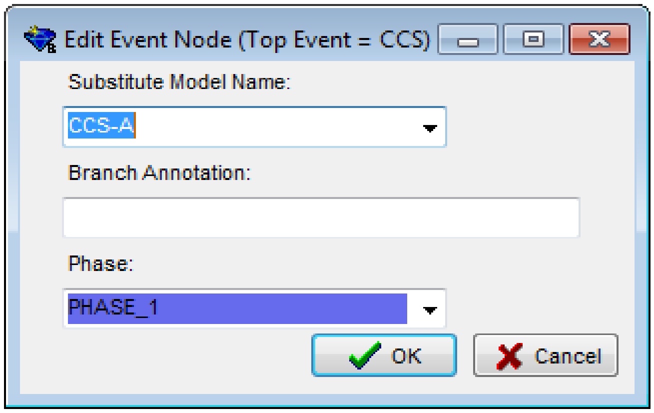
♦ To use this option, open up the event tree of interest via double-clicking or highlight event tree, right-mouse click and select **Edit Logic**.

♦ Select the node  beneath the default top event that needs to be changed to a new top event based on success criteria, etc.

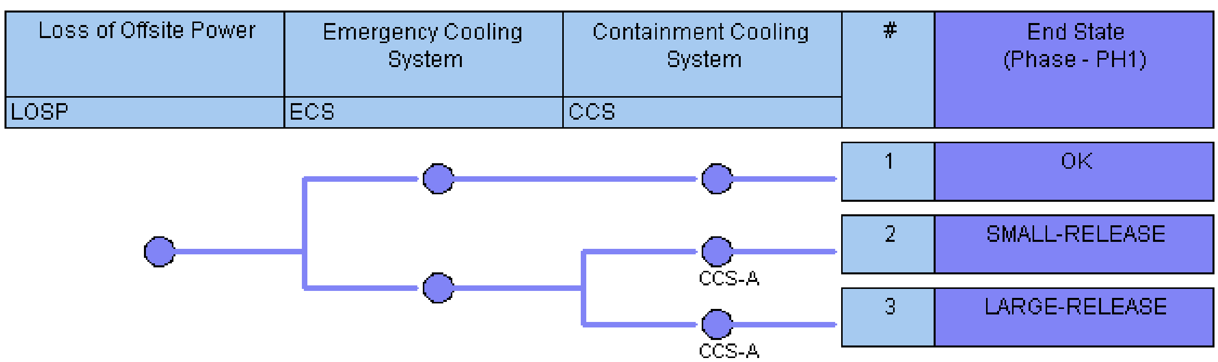
◊ Right-click the mouse and select **Edit**



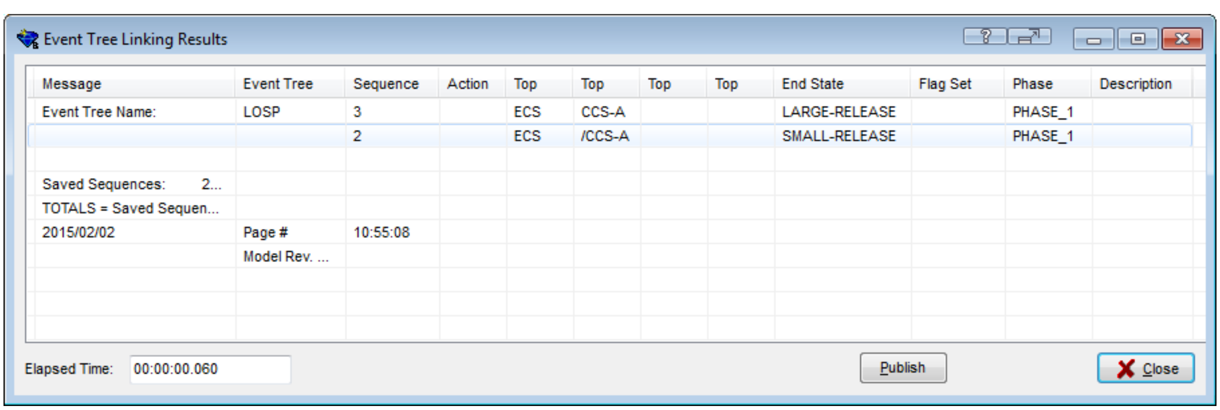
◊ Click the drop down option under "Substitute Model Name:" and then select the new top event from the list of top events (fault trees) that is going to be used for this specific sequence.



◊ If the same fault tree is going to be used for both the success and failure, which in most cases will be true, then both success and failure branch nodes need to specify the same top event (fault tree).



◊ Now save the event tree and the new sequence logic based on the top event substitution will be used when solving the sequences for minimal cut sets.



| 5 | POST PROCESSING RULES

Section 5 presents the "Post Processing Rules" editors. These editors allow you to create rules that affect existing cut sets in a "post-processing" fashion. The rule-based editors are available for both fault tree and sequence cut sets.

5.1. Post Processing Rules Editor Introduction

The SAPHIRE Post Processing Rules are "free-form" logic rules that allow for the alteration or deletion of fault tree or sequence cut sets. These post-processing rules can be applied to all event trees and fault trees via (**Project ➔ Edit Rules ➔ ET(Post Processing) or FT(Post Processing)** or applied to individual event trees and fault trees via highlighting the event tree or fault tree, right mouse click and selecting **Edit Post-processing Rules**.

> **Note:**
> Formerly called "recovery rules," post processing rules have evolved from the simple inclusion of recovery events into a powerful rule-based system for cut set manipulation.

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

- The automated inclusion of sequence recovery events
- The inclusion of dependent operator actions
- 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 single event tree, or all sequences.

Item	Menu Path	Name of rule(s)
Specific fault tree	Fault Tree ➔ Edit Post-processing Rules	“Fault Tree” Rule Level
All fault trees	Project ➔ Edit Rules ➔ FT	“Project” Rule Level
Single event tree	Event Tree ➔ Edit Post-processing Rules	“Event Tree” Rule Level
All sequences	Project ➔ Edit Rules ➔ ET	“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 Post Processing Rules could result in non-minimal cut sets. Thus, the typical steps in performing an analysis using the Post Processing Rules are:

- Finalize logic models and data changes by saving any modified trees
- Solve fault tree or sequence cut sets
- Apply Post processing 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. Post Processing Rules Nomenclature and Structure

These examples apply to all the Post Processing rules editors. This 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

5.2.1. Post Processing Rules Examples

Post Processing 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
  AddEvent = BUSREC;          |This keyword line must end with a semicolon.
endif
```

Post Processing 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

Post Processing 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 Post Processing Rules Option.

| The rule attaches the recovery action NRAC-RMCOOL to every cut set for a
| particular sequence (or all sequences).
| This rule would probably be typed as a project rule to be applied to all event trees;
| however, it can be typed into the event tree sequence rule editor for the sequence
| of interest.
| A rule to apply NRAC-RMCOOL recovery event to all cut sets in the sequence.

if RM-A-FAN * RM-B-FAN then
  recovery = NRAC-RMCOOL;
endif

Post Processing 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 Post Processing 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
```

Post Processing Rule Structure (Example 5 – including dependent operator action events)

The example below shows how the rules could be used to add dependent operator action events to the cut sets via the Post Processing Rules Option.

- The usefulness of the Post Processing Rules for dependent operator action modeling is limited by the fact that the cut sets containing the independent operator actions must exist for the search criteria to work.
- If a probability truncation is specified when generating fault tree or sequence cut sets, the independent operator action cut sets may be truncated.

```
| The search criterion identifies the failure combination of two operator actions in
|series.  If these two basic events are found in a cut set then one of the operator
|actions will be removed and replaced with a new dependent operator action
|performing the same operation.
| 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
| RCI-XHE-XM-ERROR and HCS-XHE-XM-ERROR.

XHE-DEP = RCI-XHE-XM-ERROR * HCS-XHE-XM-ERROR;

| Search for the operator actions
if XHE-DEP **then**
| Now remove the independent operator action that is dependent upon the failure
| of the previous operator action.
DeleteEvent = HCS-XHE-XM-ERROR;
| Now add the dependent operator action
AddEvent = HCS-XHE-XM-ERROR1;
endif

The new cut set would be RCI-XHE-XM-ERROR * HCS-XHE-XM-ERROR1.

Post Processing 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 Post Processing 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. End Section Post Processing Rule Keyword

End section rule keyword was developed to break a post-processing rule into separate rules that are applied to cut sets. The keyword allows one rule to manipulate the cut sets and stops at that point and then these manipulated cut sets are processed by the next rule. This continues until all rules have been applied.

♦ The following will provide an example of the structure and application of this keyword to post-processing rules.

◊ Assume the following cut sets are generated via the logic model

EPS-DGN-FR-A5HR * OEP-XHE-XL-1HR +
EPS-DGN-FR-B5HR * OEP-XHE-XL-1HR;

◊ Now based on the structure or some other determining factor the 1 hour recovery event (OEP-XHE-XL-1HR needs to be replaced by a 5 hour recovery event). The following rule can be developed.

```
if OEP-XHE-XL-1HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
  DeleteEvent = OEP-XHE-XL-1HR;
  AddEvent = OEP-XHE-XL-5HR;
endif
```

◊ The new cut sets after the post processing rule is applied is:

EPS-DGN-FR-A5HR * OEP-XHE-XL-5HR +
EPS-DGN-FR-B5HR * OEP-XHE-XL-5HR;

◊ Now if there is a subsequent rule that has the following search criteria:

```
if OEP-XHE-XL-1HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
  AddEvent = OEP-XHE-XX-1HR;
endif

if OEP-XHE-XL-5HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
  AddEvent = OEP-XHE-XX-5HR;
endif
```

this rule will be applied to the original cut sets (generated via logic)
[EPS-DGN-FR-A5HR * OEP-XHE-XL-1HR +
EPS-DGN-FR-B5HR * OEP-XHE-XL-1HR]

◊ The new cut sets would look like (given the substitution from earlier rule):

EPS-DGN-FR-A5HR * OEP-XHE-XL-5HR * **OEP-XHE-XX-1HR** +
EPS-DGN-FR-B5HR * OEP-XHE-XL-5HR * **OEP-XHE-XX-1HR**;

The final outcome is not the correct resultant cut sets. This is due to how SAPHIRE handles the cut sets and rules. SAPHIRE loads the cut sets up and looks for all rules that need to be applied to this group of cut sets and then applies these rules separately.

◊ There are two ways to get the correct cut sets:
1. Use the elsif option as discussed above, or
2. Use the End_Rule_Section keyword

◊ The End_Rule_Section rule option will be illustrated below and would use the following rule structure. The resultant cut sets are also provided.

```
| Post-processing rules applied in the first pass through the cut sets.
    if OEP-XHE-XL-1HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
      DeleteEvent = OEP-XHE-XL-1HR;
      AddEvent = OEP-XHE-XL-5HR;
    endif

| Use the key word End_Rule_Section to separate the first set of post-processing rules
| with the new set of post-processing rules.

    End_Rule_Section

| These rules will be applied to the resultant cut sets that are created (modified) from the
| first set of rules (look at this as the second pass).

    if OEP-XHE-XL-1HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
      AddEvent = OEP-XHE-XX-1HR;
    endif

    if OEP-XHE-XL-5HR * (EPS-DGN-FR-A5HR + EPS-DGN-FR-B5HR) then
      AddEvent = OEP-XHE-XX-5HR;
    endif
```

The resultant cut sets after rules applied:

EPS-DGN-FR-A5HR * OEP-XHE-XL-5HR * **OEP-XHE-XX-5HR** +
EPS-DGN-FR-B5HR * OEP-XHE-XL-5HR * **OEP-XHE-XX-5HR**;

The End_Rule_Section takes the group of cut sets that have been manipulated by the first set of post processing rules and then applies the next set of post processing rules, and continues if more section breaks are developed.

5.4. Convolution Post Processing Rule Keyword

Convolution keyword was developed to append the specially created basic event to adjust sequence cut set frequencies based on time related basic events failing within a cut set. This rule will search through the cut sets and when the cut set has been found the special basic will be appended to the cut set in order to reduce its frequency based on what the convolved sequence frequency should be.

- The following will provide an example of the structure and application of this keyword to post-processing rules.

    - Assume the following sequence cut set is generated via the logic model

    IE-LOOP * EPS-DGN-FR-A * EPS-DGN-FS-B * OEP-XHE-XL-NR01H;
    The frequency (assume) is 1.09E-5/yr

    - Since, the failure of this cut set is assumed to start at time = 0.0 and the mission time is 24 hours for the diesel generator; this cut set frequency over-estimates the fact that the failure actually occurred sooner and the operators only have 1 hour to recover offsite power.

    - The correction factor needs to be applied. The rule structure is simple, since the keyword knows exactly what combinations to search for (developed when the basic event was created).

```
if Convolve_Cut_Sets then
  AddConvolEvent;
endif
```

    - The above rule will search through the cut sets and where the cut set combinations occur the convolution basic event is applied. For the simple Demo project created the cut set of interest is:

    IE-LOOP * EPS-DGN-FR-A * EPS-DGN-FS-B * OEP-XHE-XL-NR01H;

Since EPS-DGN-FR-A * OEP-XHE-XL-NR01H shows up together is a cut set; the convolved basic event created in Section 3 will be appended to this cut set to reduce the sequence cut set frequency

IE-LOOP * EPS-DGN-FR-A * EPS-DGN-FS-B * OEP-XHE-XL-NR01H * OEP-XHE-NN-NR01H1;
The new frequency (correct) is 1.53E-6/yr (a factor of 0.139)

5.5. Post Processing Rule Keywords and Nomenclature

Each of the "rules" in SAPHIRE (linking, post processing, and partition) has their own nomenclature. The table below lists the keywords available for post processing 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 Post Processing 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 Post Processing 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

Keyword or symbol	Definition	Usage

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

End_Rule_Section	Provides a break from one rule group to the next rule group in layered rules.	First set of rules **End_Rule_Section** Second set of rules.
~	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.
/	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.

Keyword or symbol	Definition	Usage
+	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

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

Keyword or symbol	Definition	Usage

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

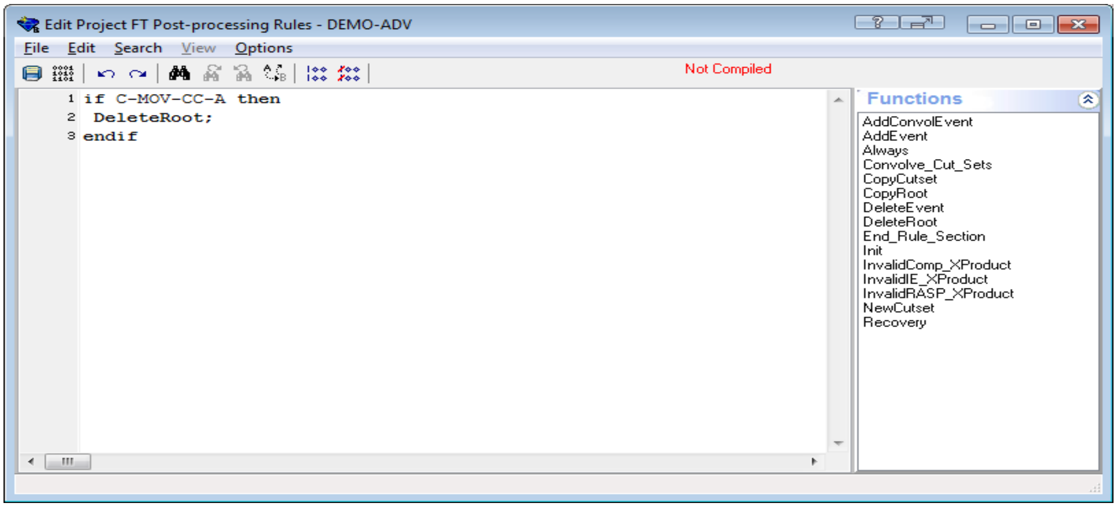
Convolve_Cut_Sets	Keyword that indicates the original cut set (i.e., that cut set that satisfied the search criteria) will be adjusted by a correction factor. The keyword for the correction factor in the rule is **AddConvolEvent**.	If Convolve_Cut_Sets then
		AddConvolEvent;
		endif

InvalidRASP_XProduct	Keyword that indicates the original cut set (i.e., that cut set that satisfied the search criteria) will be removed from the cut set. This is used when fully expanded R-Type CCF events are used in the PRA model	If InvalidRASP_XProduct then
		DeleteRoot;
		endif

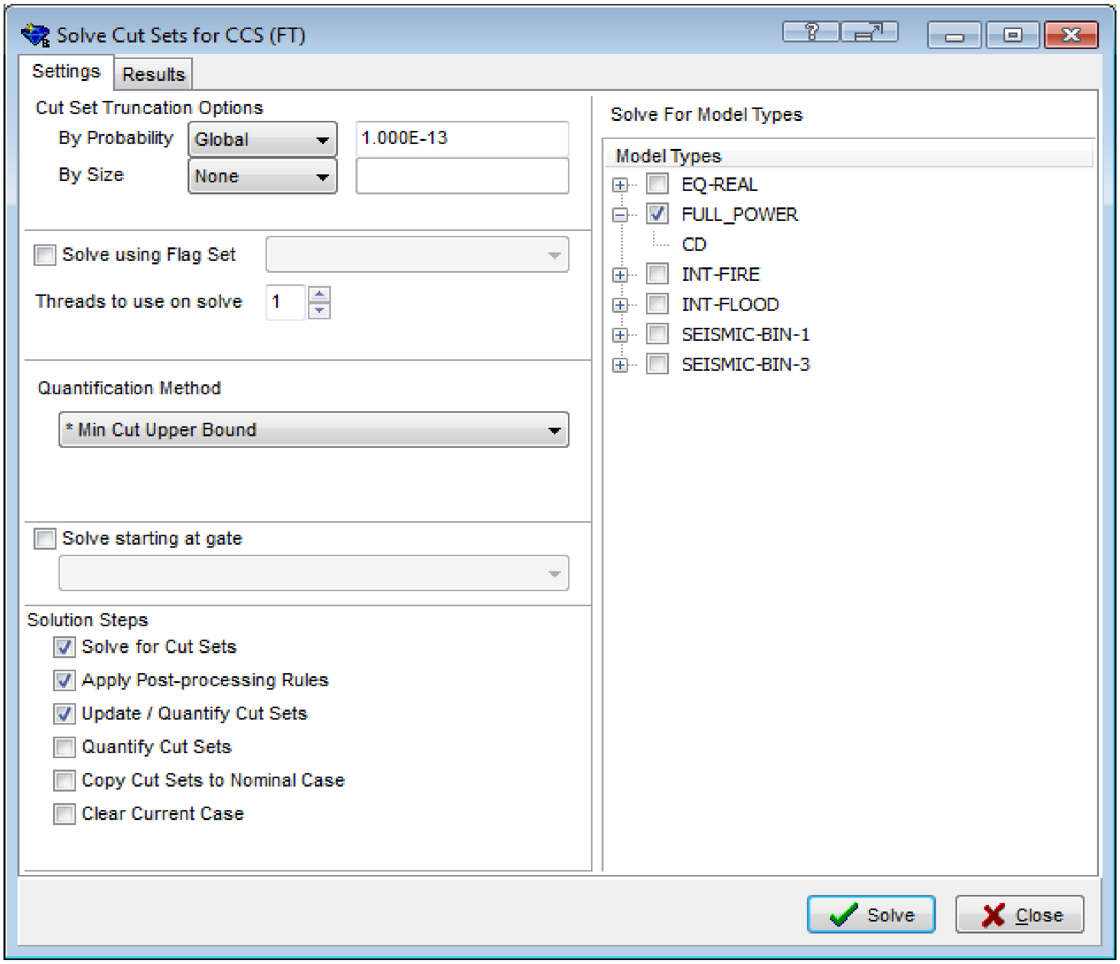
5.6. Fault Tree Post Processing Rules

♦ To create or edit Fault Tree Post Processing Rules on an **individual** fault tree, highlight the fault tree in the fault tree list, right-click and select **Edit Post-processing Rules**.

♦ To create or edit Fault Tree Post Processing Rules that will be used on **all** fault trees in the project, from the main SAPHIRE screen, select **Project ➔ Edit Rules ➔ FT(Post Processing).**



♦ Fault Tree Post Processing Rules are applied to fault tree cut sets by checking the **Apply Post-processing Rules** checkbox. If the project contains post-processing rules, this option will be available to be checked. If there are no post-processing rules, this checkbox option will be grayed out. If there are post-processing rules that need to be applied, this checkbox needs to be checked or no rules will be applied.



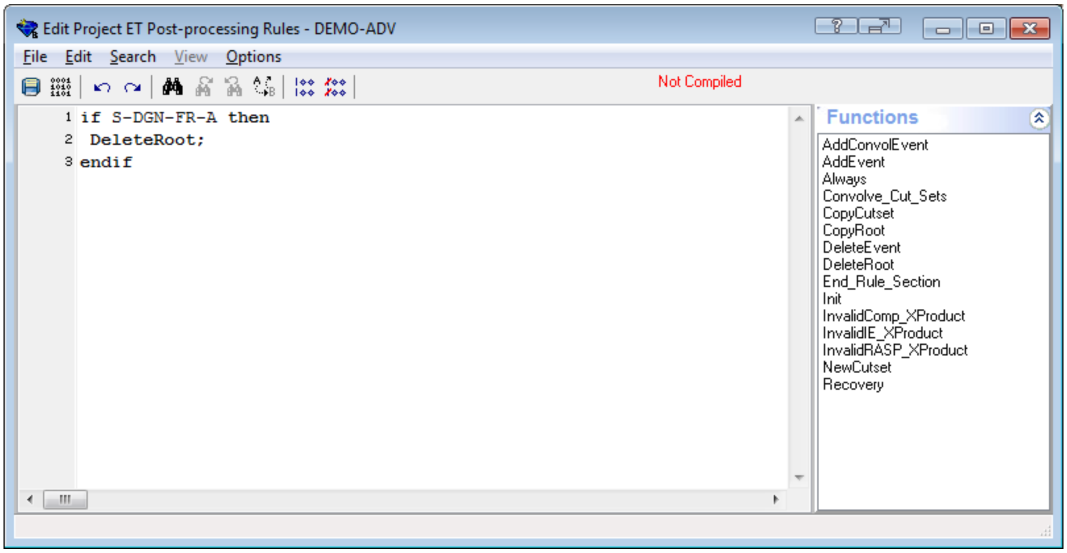
- The post-processing rules are applied in the following order:
  1. Fault tree specific post-processing rule
  2. Project (Fault Tree) post-processing rule

- If there are any over lapping rules, the final cut sets may be incorrect (i.e., fault tree specific rule adds a recovery event and then the project rule adds a different recovery event, the final cut set will have multiple recovery events, which may be incorrect.)

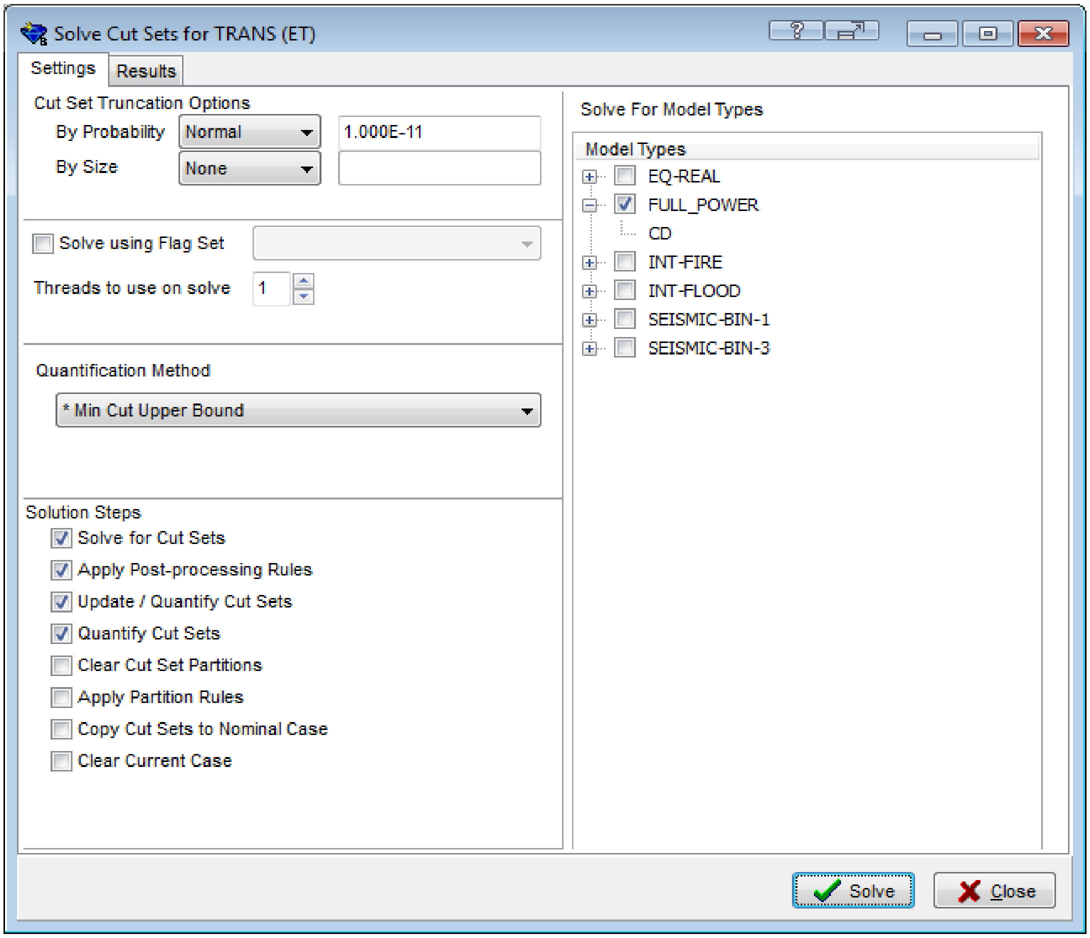
5.7. Event Tree Sequence Post Processing Rules

- To create or edit Event Tree Post Processing Rules on an **individual** event tree, highlight the event tree in the event tree list, right-click and select **Edit Post-processing Rules**.

- To create or edit Event Tree Post Processing Rules on **all** event trees in the project, from the main menu, select **Project ➔ Edit Rules ➔ ET(Post Processing).**



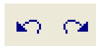
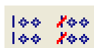
- Event Tree Post Processing Rules are applied to event tree cut sets by checking the **Apply Post-processing Rules** checkbox. If the project contains post-processing rules, this option will be available to be checked. If there are no post-processing rules, this checkbox option will be grayed out. If there are post-processing rules that need to be applied, this checkbox needs to be checked or no rules will be applied.



- The post-processing rules are applied in the following order:
    1. Event tree specific post-processing rule
    2. Project (Event Tree) post-processing rule

- If there are any over lapping rules, the final cut sets may be incorrect (i.e., event tree specific rule adds a recovery event and then the project rule adds a different recovery event, the final cut set will have multiple recovery events, which may be incorrect.)

Rules Editor Toolbar:

	Save
	Compile
	Undo or Redo action
	Find, Find Next, and Find Previous
	Replace
	Comment highlighted block, Un-Comment highlighted block

Rules Editor Menu Bar:

File – The file drop down menu allows the user to compile, import or export text, and save or both save and exit the rule editor. A check of the script status is performed on an exit of the rule editor. If there is a problem found (i.e. script is not compiled or has errors in the compile) the user is prompted to optionally fix the problem without exiting the editor.

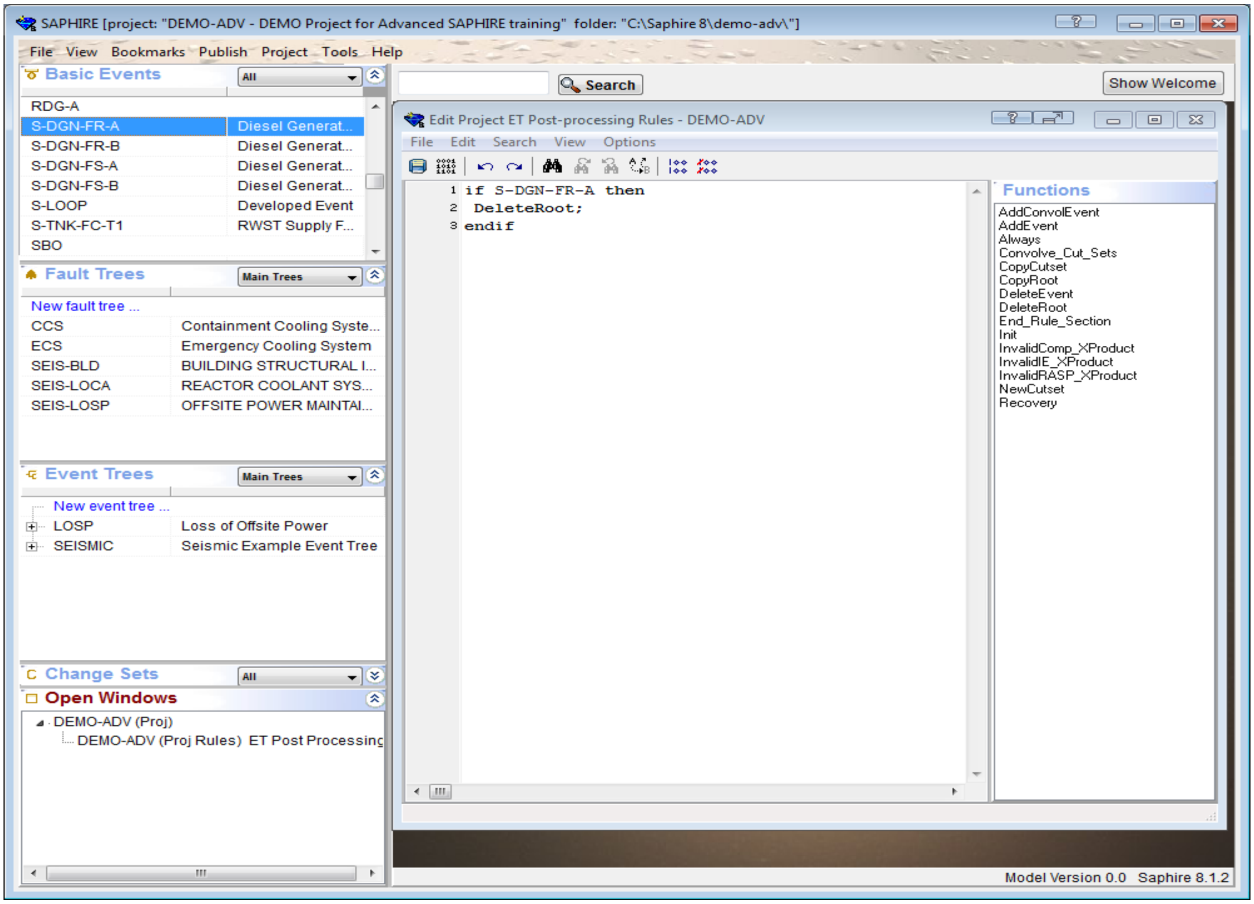
Edit – The edit drop down menu allows the user to undo, redo, cut, copy, paste insert indents or spaces, comment or uncomment any portions of the existing rules,

Search – The search drop down menu allows the user the ability to search the rules, find the next or previous, or replaces character strings.

View – The view drop down menu is used in concurrence with **Options ➔ Show Item Lists** and will show or hide items in the list pane opened to the left of the editing pane.

Options – The options drop down menu controls what panes are visible (Show Function List and Show Item List), what is displayed in the list pane (Header and Description), and can save the current configuration as the default configuration for the rules editors in the project.

The following image shows the rules editor including the list pane.



- Editing Pane
  - ◊ Items can be dragged and dropped to the editing pane from the main list panes outside of the editor, the list pane within the editor, and the function pane within the editor.
  - ◊ Items can be inserted where the cursor is in the editing pane by double-clicking on a list item or function within the editor.
    - Double-clicking on a main list item outside the editor opens that list item for editing.

> **Note:**
>
> Functions are **not** case sensitive, (i.e. AddEvent is the same as addevent) but variables **are** case sensitive (DG-A is different than DG-a).

5.8. A “Complicated” Post Processing Rule Example

Now, an example is presented that utilizes several of the Post Processing 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:

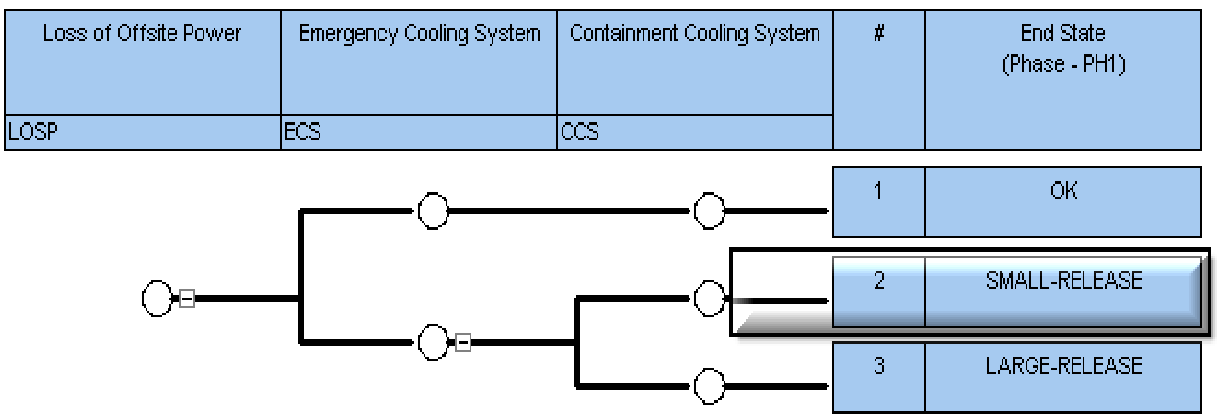
If A **then**
  AddEvent = B;
  NewCutset;
  AddEvent = C;
  DeleteRoot;
  CopyCutset;
  AddEvent = D;
endif

The outcome of applying this Post Processing 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.



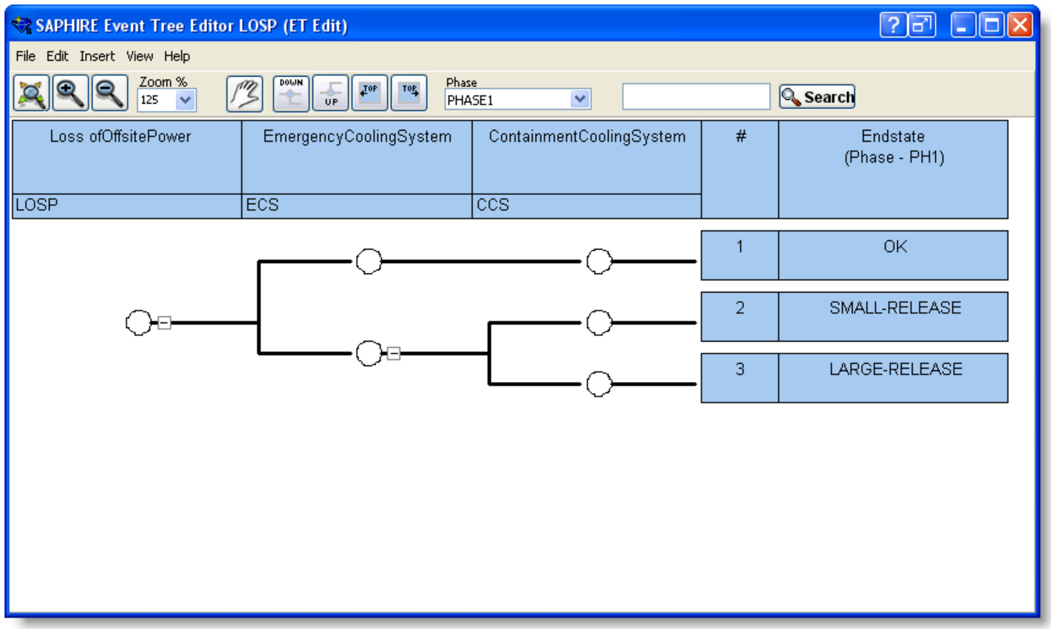
6.1. End State Analysis Approaches

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.
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.



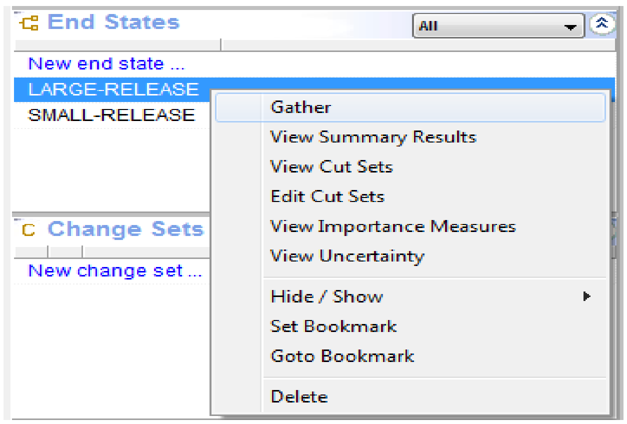
The end states specified in the "End-State" column automatically become end states in the database.

- The graphical event tree editor discussed in the Basics course on how to develop an event tree.
- To edit the description of an end state from the main screen, double click on an end state in the End State list panel. More detailed editing can be accomplished in the Event Tree editor by right clicking on the sequence end state and selecting Edit as described in the basics manual.
- Delete end states that are no longer used by right clicking on the end state and selecting **Delete**.
- Cross reference an end state by highlighting the end state and using the main menu **Tools ➔ Cross References** to list the event tree sequences that are grouped into the highlighted end state.

6.2.1. Gather End State Cut Sets

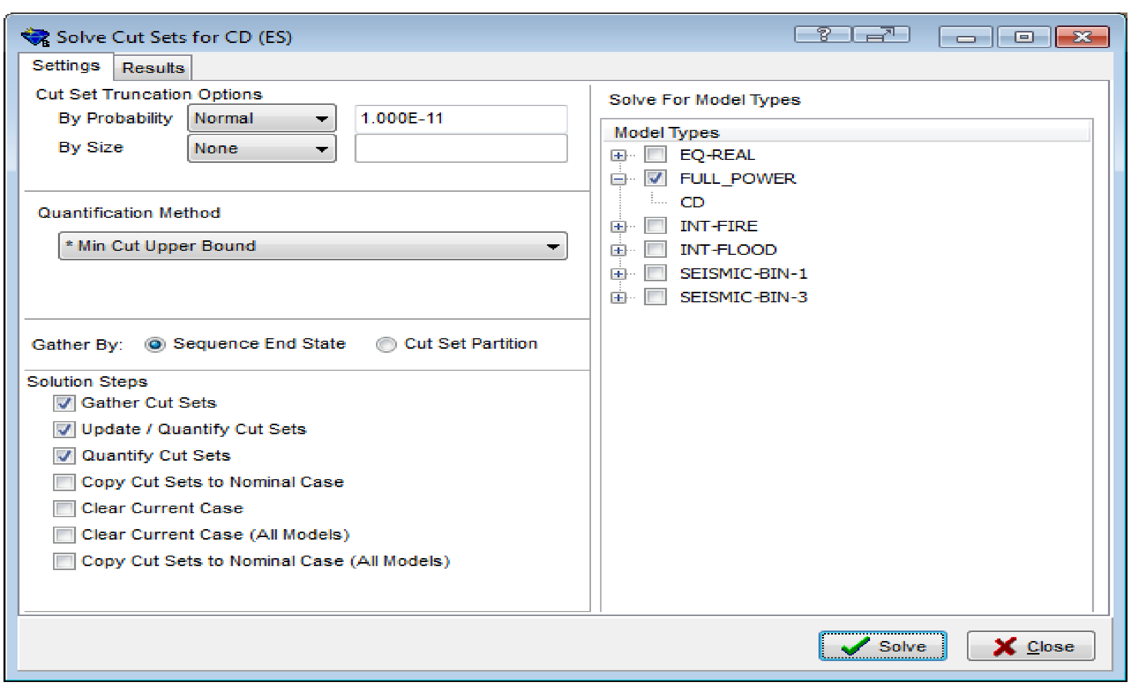
Steps to gather End State Cut Sets:

1. Sequence cut sets must be generated (highlight all Event Trees then right click on the highlighted list and select the Solve option).
2. Gather end state cut sets by highlighting all the End States of interest in the End States list panel, right click on the highlighted list and select Gather (if end states not visible, use the main toolbar **View → End States** to see the End States list panel).



6.2.2. Options to Gather End States Cut Sets

Performing the steps in 6.2.1 opens the Solve Cut Sets form shown below.



Cut Set Truncation Options

Cut Set Truncation (Probability)

- ◊ Choose None (i.e., no truncation) or choose Normal and manually set the truncation value in the textbox (it automatically defaults to the project setting). The cut sets below the selected value will not be retained.

Size Truncation – From the dropdown menu,

- ◊ **Size** eliminates cut sets having more events than specified value textbox to the right of Size Truncation.
- ◊ **None** and the number of events in a cut set will not affect whether the cut set is kept or discarded.
- ◊ **Zone** eliminates cut sets having more events with the Process Flag = Z than specified in the value textbox.

Quantification Method

- ◊ **Minimal Cut Set Upper Bound** – Calculates the end state frequency using the mincut upperbound equation.
- ◊ **Rare Event** – Calculates the end state frequency by summing up the cut sets.
- ◊ **Min/Max** – Calculates the using the "inclusion/exclusion" rule.

Gather By

- ◊ **Sequence End State** – Activate this radio button to gather the end states specified on the event tree sequences (e.g., via the graphics).
- ◊ **By Cut Set Partition** – Activate this radio button to gather the end states created when end state partitioning rules were applied.

Solution Steps

Gather Cut Sets

- ◊ 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.)

Update / Quantify Cut Sets

- ◊ This option uses the existing end state cut sets and performs Boolean algebra step to remove any non-minimal cut sets and then requantifies these cut sets minimal cut set upper bound approximation (or rare event) to obtain the end state frequency.

Quantify Cut Sets

◊ 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.

Copy Cut Sets to Nominal case

◊ Check this box to update the nominal case with end state cut sets and overall frequency.

Clear Current Cut Sets

◊ Check this box to clear out the current case cut sets (i.e., zero out the cut set listing).

Clear Current Cut Sets (All Models)

◊ Check this box to clear out the current case cut sets for all model types within the project (i.e., zero out the cut set listing).

Copy Cut Sets to Nominal case (All Models)

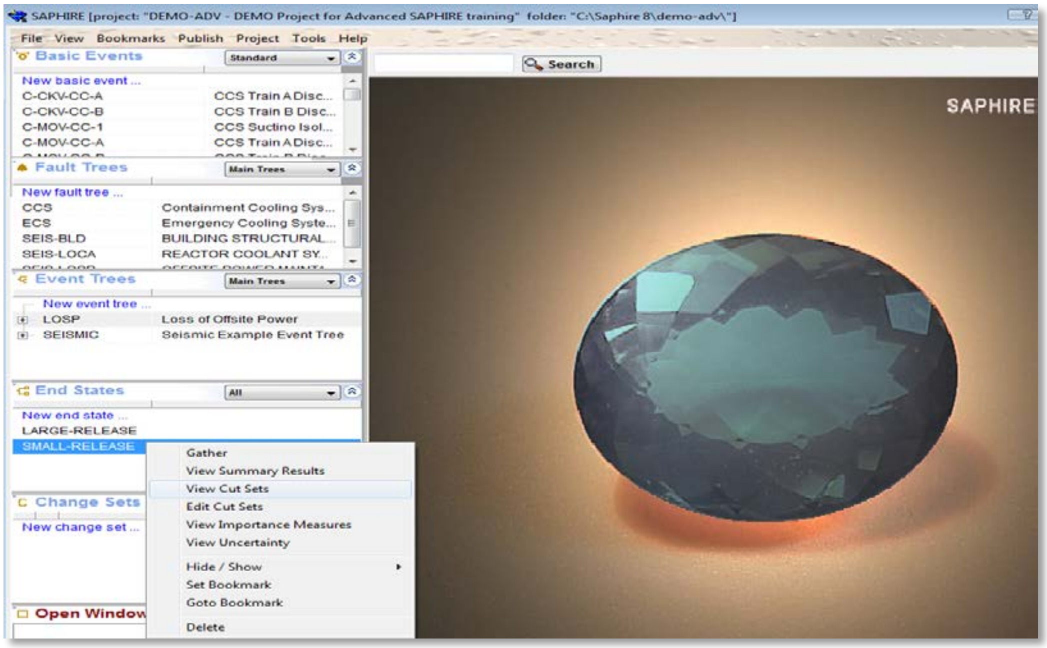
◊ Check this box to update the nominal case with end state cut sets (for all Model types within the project) and overall frequency.

Solve For Model Types

◊ Check marks should be in the model types of interest.

6.2.3. Description of End States list menu options

The list menu for End States is accessed by right clicking on highlighted end state(s) of interest:



Gather

◊ Discussed above; this option gathers existing cut sets (generated from the event tree sequences). The end state frequency is quantified using the specified calculation. (Non-minimal cut sets are eliminated within each end state.)

View Summary Results

◊ This option opens a results form for the chosen end state(s) which includes Point estimates and uncertainty.

View Cut Sets

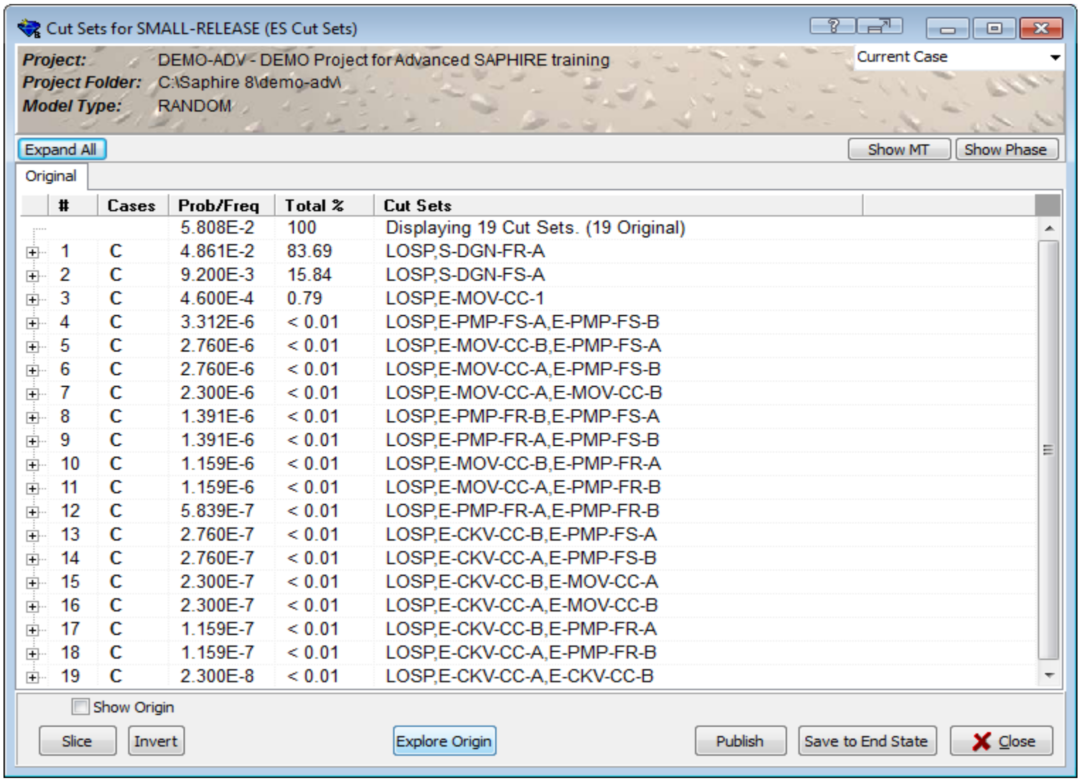
◊ This option displays the chosen end state(s) cut sets as illustrated below:

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

◊ The cut sets can be expanded to show basic information and the path by individually by clicking on the "+" to the left of the numbered cut set or view all cut sets as expanded by clicking on **Expand All**. To show the origin, activate the **Show Origin** checkbox.

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

◊ Click on Slice to slice the end state cut sets by Cutoff, Events, and Attributes using the form and functions described in **Section 15, Viewing Cut Sets (SAPHIRE Basics Manual).**



Edit Cut Sets

◊ This option allows the analyst to edit the cut sets (current and nominal) using the form and functions as described in **Section 14, Editing Cut Sets**.

View Importance Measures

◊ This option calculates and lists the basic events importance measures for a single or group of end states.

View Uncertainty

◊ This option allows Monte Carlo or Latin Hypercube samples calculation and view of uncertainty analysis for a single or group of end states.

6.3. End States via Partition Rules

Partition rules are applied either:

- ♦ Globally to all event trees by using the End State Partition editor under **Project ➔ Edit Rules ➔ End State Partition**, or
- ♦ An individual event tree by right clicking on the event tree in the Event Tree list panel and selecting **Edit Partition Rules**.

6.3.1. End State Partitioning Rules Nomenclature and Structure

The rule structure and nomenclature for the partitioning rules are similar to the "Link Event Tree" rules and “Post Processing 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

6.3.2. End State Partitioning Rules Examples

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 end state 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
```

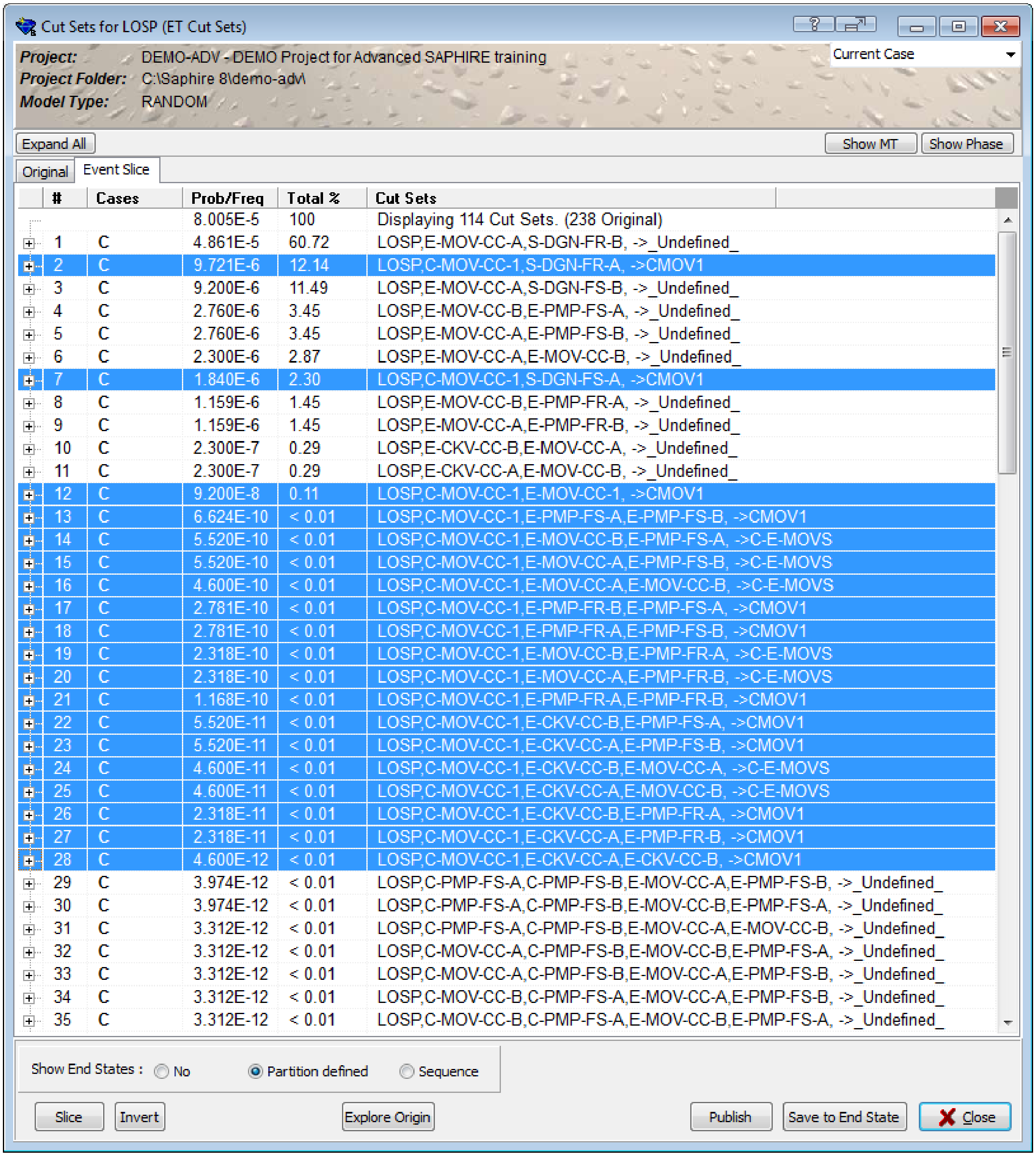
Partition Rule Structure (Example 7 – Current Partition)

```
| 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-CC-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-CC-A or E-MOV-CC-B.
|
if C-MOV-CC-1 then
  partition = "CMOV1";
endif

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



#	Cases	Prob/Freq	Total %	Cut Sets
		8.005E-5	100	Displaying 114 Cut Sets. (238 Original)
1	C	4.861E-5	60.72	LOSP,E-MOV-CC-A,S-DGN-FR-B, ->_Undefined_
2	C	9.721E-6	12.14	LOSP,C-MOV-CC-1,S-DGN-FR-A, ->CMOV1
3	C	9.200E-6	11.49	LOSP,E-MOV-CC-A,S-DGN-FS-B, ->_Undefined_
4	C	2.760E-6	3.45	LOSP,E-MOV-CC-B,E-PMP-FS-A, ->_Undefined_
5	C	2.760E-6	3.45	LOSP,E-MOV-CC-A,E-PMP-FS-B, ->_Undefined_
6	C	2.300E-6	2.87	LOSP,E-MOV-CC-A,E-MOV-CC-B, ->_Undefined_
7	C	1.840E-6	2.30	LOSP,C-MOV-CC-1,S-DGN-FS-A, ->CMOV1
8	C	1.159E-6	1.45	LOSP,E-MOV-CC-B,E-PMP-FR-A, ->_Undefined_
9	C	1.159E-6	1.45	LOSP,E-MOV-CC-A,E-PMP-FR-B, ->_Undefined_
10	C	2.300E-7	0.29	LOSP,E-CKV-CC-B,E-MOV-CC-A, ->_Undefined_
11	C	2.300E-7	0.29	LOSP,E-CKV-CC-A,E-MOV-CC-B, ->_Undefined_
12	C	9.200E-8	0.11	LOSP,C-MOV-CC-1,E-MOV-CC-1, ->CMOV1
13	C	6.624E-10	< 0.01	LOSP,C-MOV-CC-1,E-PMP-FS-A,E-PMP-FS-B, ->CMOV1
14	C	5.520E-10	< 0.01	LOSP,C-MOV-CC-1,E-MOV-CC-B,E-PMP-FS-A, ->C-E-MOVS
15	C	5.520E-10	< 0.01	LOSP,C-MOV-CC-1,E-MOV-CC-A,E-PMP-FS-B, ->C-E-MOVS
16	C	4.600E-10	< 0.01	LOSP,C-MOV-CC-1,E-MOV-CC-A,E-MOV-CC-B, ->C-E-MOVS
17	C	2.781E-10	< 0.01	LOSP,C-MOV-CC-1,E-PMP-FR-B,E-PMP-FS-A, ->CMOV1
18	C	2.781E-10	< 0.01	LOSP,C-MOV-CC-1,E-PMP-FR-A,E-PMP-FS-B, ->CMOV1
19	C	2.318E-10	< 0.01	LOSP,C-MOV-CC-1,E-MOV-CC-B,E-PMP-FR-A, ->C-E-MOVS
20	C	2.318E-10	< 0.01	LOSP,C-MOV-CC-1,E-MOV-CC-A,E-PMP-FR-B, ->C-E-MOVS
21	C	1.168E-10	< 0.01	LOSP,C-MOV-CC-1,E-PMP-FR-A,E-PMP-FR-B, ->CMOV1
22	C	5.520E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-B,E-PMP-FS-A, ->CMOV1
23	C	5.520E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-A,E-PMP-FS-B, ->CMOV1
24	C	4.600E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-B,E-MOV-CC-A, ->C-E-MOVS
25	C	4.600E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-A,E-MOV-CC-B, ->C-E-MOVS
26	C	2.318E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-B,E-PMP-FR-A, ->CMOV1
27	C	2.318E-11	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-A,E-PMP-FR-B, ->CMOV1
28	C	4.600E-12	< 0.01	LOSP,C-MOV-CC-1,E-CKV-CC-A,E-CKV-CC-B, ->CMOV1
29	C	3.974E-12	< 0.01	LOSP,C-PMP-FS-A,C-PMP-FS-B,E-MOV-CC-A,E-PMP-FS-B, ->_Undefined_
30	C	3.974E-12	< 0.01	LOSP,C-PMP-FS-A,C-PMP-FS-B,E-MOV-CC-B,E-PMP-FS-A, ->_Undefined_
31	C	3.312E-12	< 0.01	LOSP,C-PMP-FS-A,C-PMP-FS-B,E-MOV-CC-A,E-MOV-CC-B, ->_Undefined_
32	C	3.312E-12	< 0.01	LOSP,C-MOV-CC-A,C-PMP-FS-B,E-MOV-CC-B,E-PMP-FS-A, ->_Undefined_
33	C	3.312E-12	< 0.01	LOSP,C-MOV-CC-A,C-PMP-FS-B,E-MOV-CC-A,E-PMP-FS-B, ->_Undefined_
34	C	3.312E-12	< 0.01	LOSP,C-MOV-CC-B,C-PMP-FS-A,E-MOV-CC-A,E-PMP-FS-B, ->_Undefined_
35	C	3.312E-12	< 0.01	LOSP,C-MOV-CC-B,C-PMP-FS-A,E-MOV-CC-B,E-PMP-FS-A, ->_Undefined_

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 end state 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 end state 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 to rapidly partition cut sets into end states based on sequence logic. Since individual cut sets are not searched, Global Partitioning rules gather cut sets faster 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.

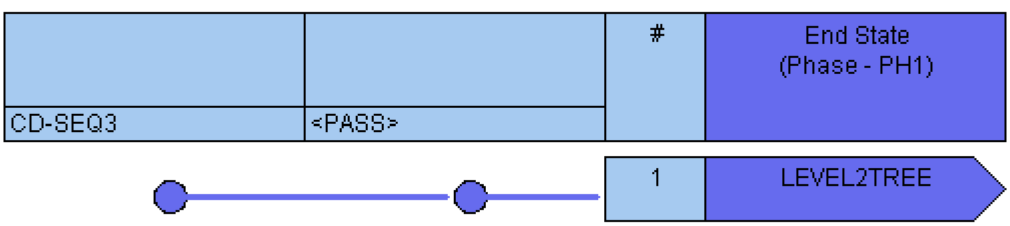
- If a "global partition" is performed on basic events, all cut sets listed after the basic event's cut set will be partitioned into the end state.

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).*
|
| *LEVEL2TREE 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 = LEVEL2TREE;
endif



6.4. Partition Rule Keywords and Nomenclature

Each of the “rules” in SAPHIRE (linking, post processing, 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 post processing 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 post processing 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

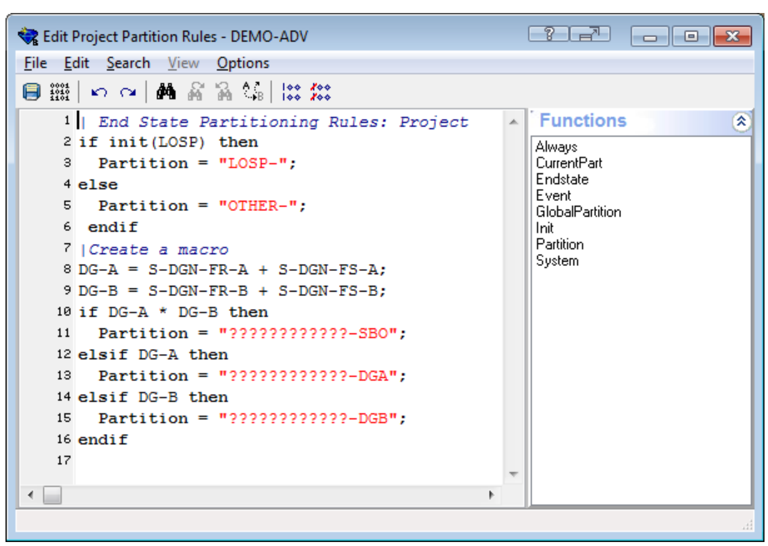
MACRO	A macro is a user-definable keyword that specifies search criteria. The macro name must be all upper-case, must be 16 characters or less, and must not include any of the restricted characters (e.g., a space, *, ?, \, /). The macro line can wrap around to more than one line, but must end with a 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

6.5. Partition Rule Example

A rule for the DEMO project is added by selecting the **Project → Edit Rules → End State Partition**.

- The first project rule shown below add characters 1 to 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).

```
| End State Partitioning Rules: Project
if init(LOSP) then
  Partition = "LOSP-";
else
  Partition = "OTHER-";
 endif
|Create a macro
DG-A = S-DGN-FR-A + S-DGN-FS-A;
DG-B = S-DGN-FR-B + S-DGN-FS-B;
if DG-A * DG-B then
  Partition = "????????????-SBO";
elsif DG-A then
  Partition = "????????????-DGA";
elsif DG-B then
  Partition = "????????????-DGB";
endif
```



Rules for LOSP sequences 2 and 3 are entered by right clicking on **LOSP** in the Event Tree list and selecting **Edit Partition Rules**, then keying on the top event fault tree CCS as shown below:

- Create a rule in sequence 2 to add characters 6 through 12 as ECSONLY and a rule for sequence 3 to add characters 6 through 12 as ECS&CCS by using the following partition rules for the event tree LOSP:

| End State Rule sequence 2

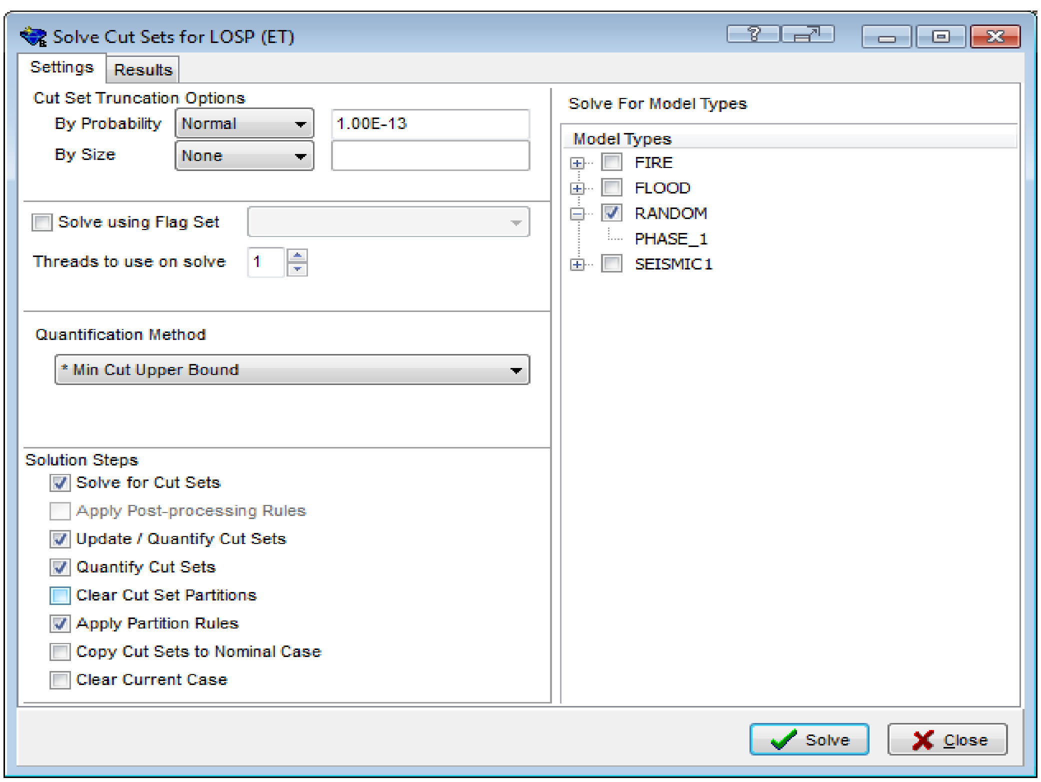
if System(/CCS) **then**
 partition = "?????ECSONLY"**;**
endif

| End State Rule sequence 3
if System(CCS) **then**
 partition = "?????ECS&CCS"**;**
endif

6.5.1. Applying the Partitioning Rules

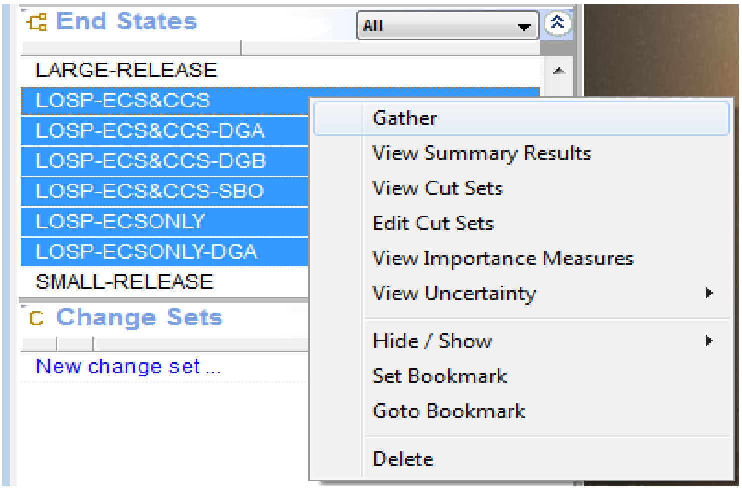
To apply the partitioning rules, highlight all of the event trees in the event tree list pane, right click the mouse, and select **Solve** to open the **Solve Cut Sets (ET)** window.

Ensure that there is a check in the **Apply Partition Rules** checkbox and click **Solve**.

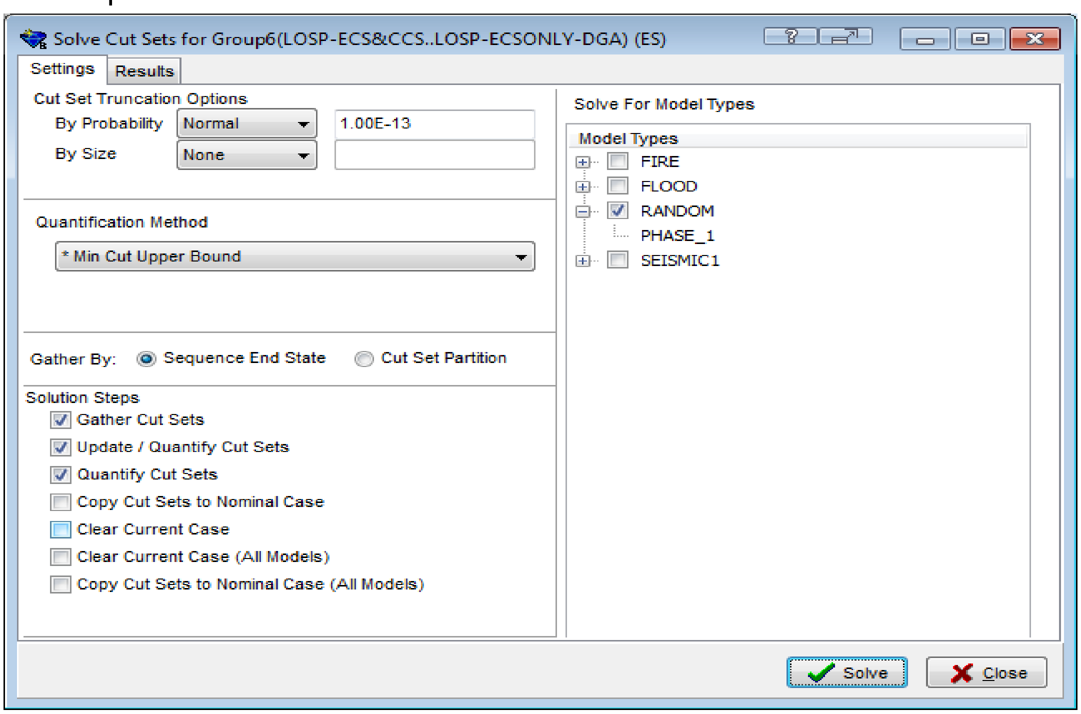


6.5.2. Using End State Analysis to Gather the Partitioned Cut Sets

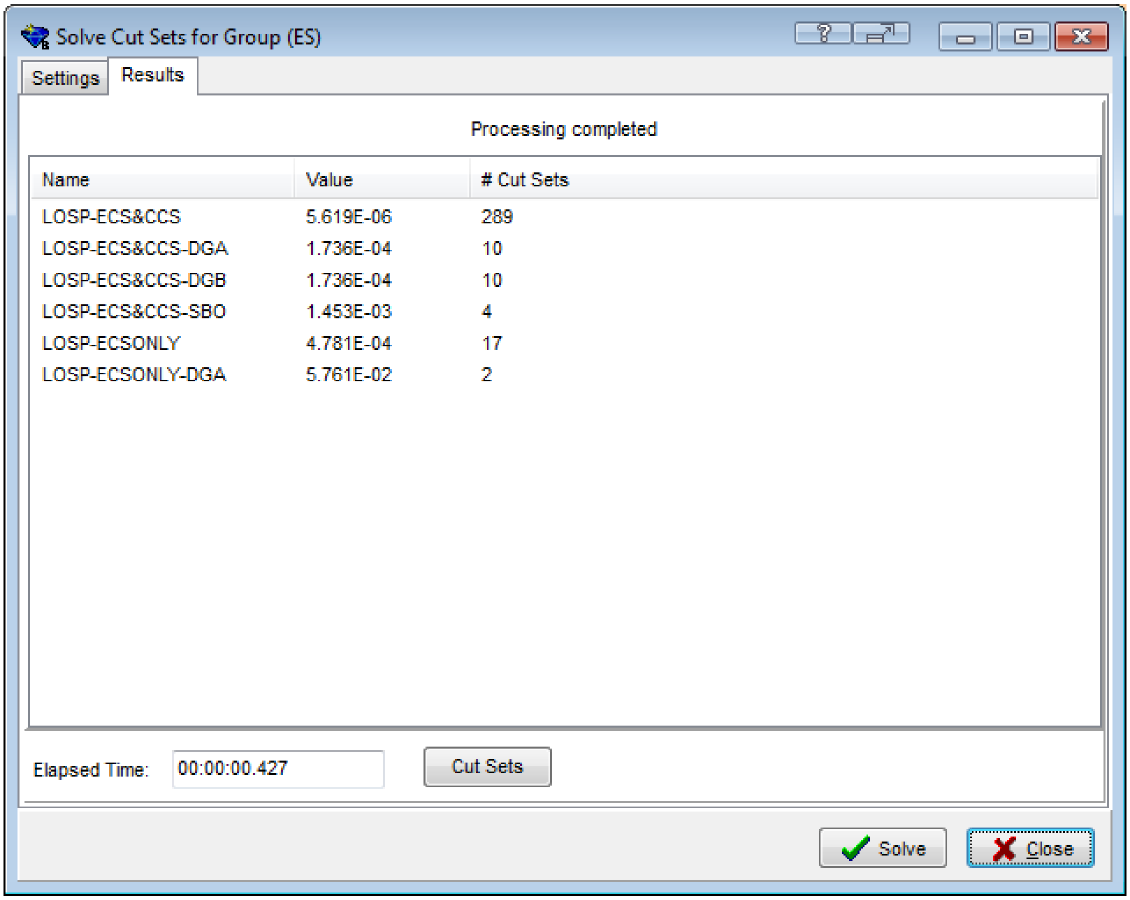
After applying the partition rules, the partitioned end states will be displayed in the **End States** list panel. The end states must be "gathered" to group those cut sets that met the search criteria specified in the rules. To “gather” end states, highlight the end states in **End States** list panel, right click to open the End State menu and select **Gather**.



When you select the Gather option, you will be asked to specify the end state truncation and solution options. Activate the **Cut Set Partition** radio button and click **Solve**.



The Solve Results form lists the frequency and the number of cut sets leading to each end state.



To view the cut sets for an end state, highlight the end state or group of end states and click on **Cut Sets**.

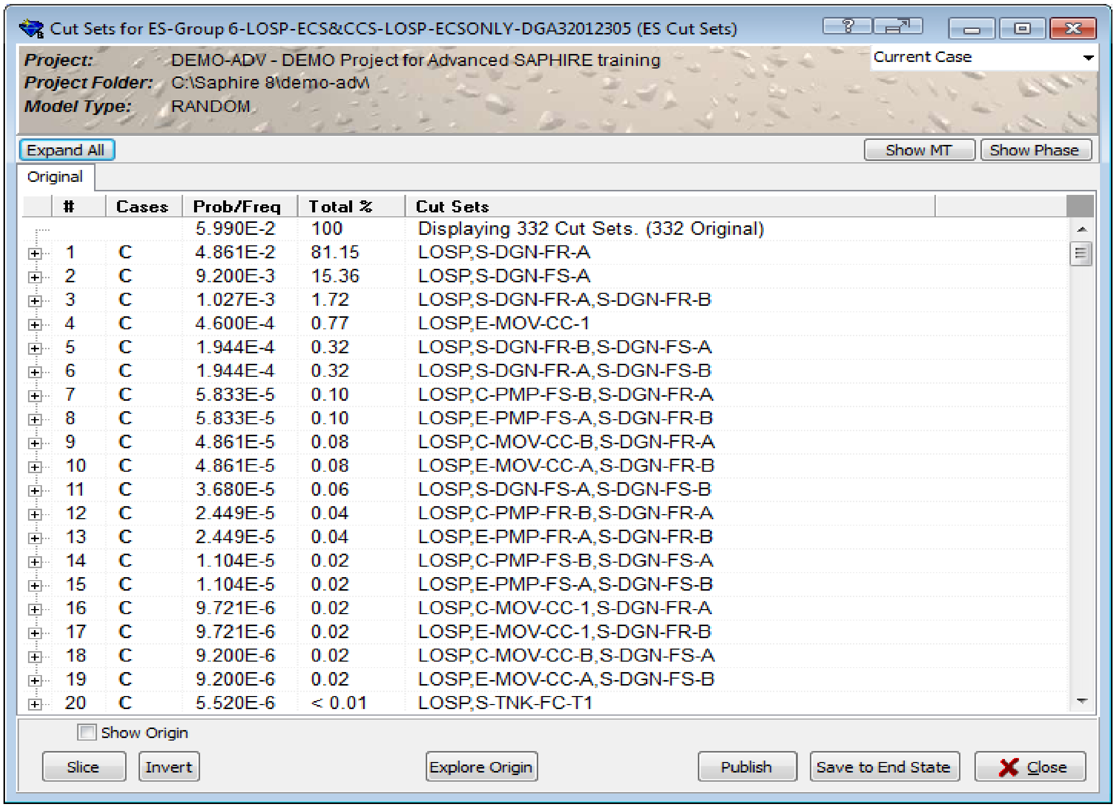
After the end state cut sets have been gathered, uncertainty analysis can be performed by using the End States list menu on a highlighted end state and selecting **View Uncertainty**. Uncertainty analysis is performed in the same manner as covered for event tree sequences.

If the Solve Results form is not active, you can view previously solved end state cut sets by right clicking on the end state or group of end states in the list pane and selecting **View Cut Sets**.

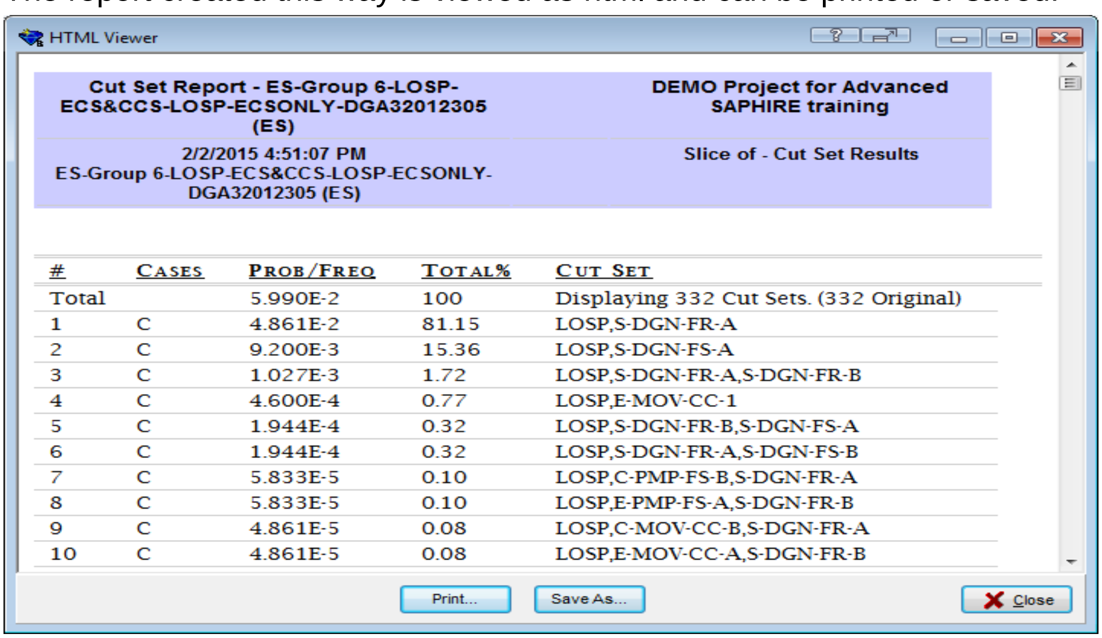
Options for viewing and slicing cut sets are identical to those for the event tree sequence and fault tree cut sets.

6.6. Reporting End State Results

End State Cut Set reports can be generated through the View Cut Sets menu by clicking on **Publish**.

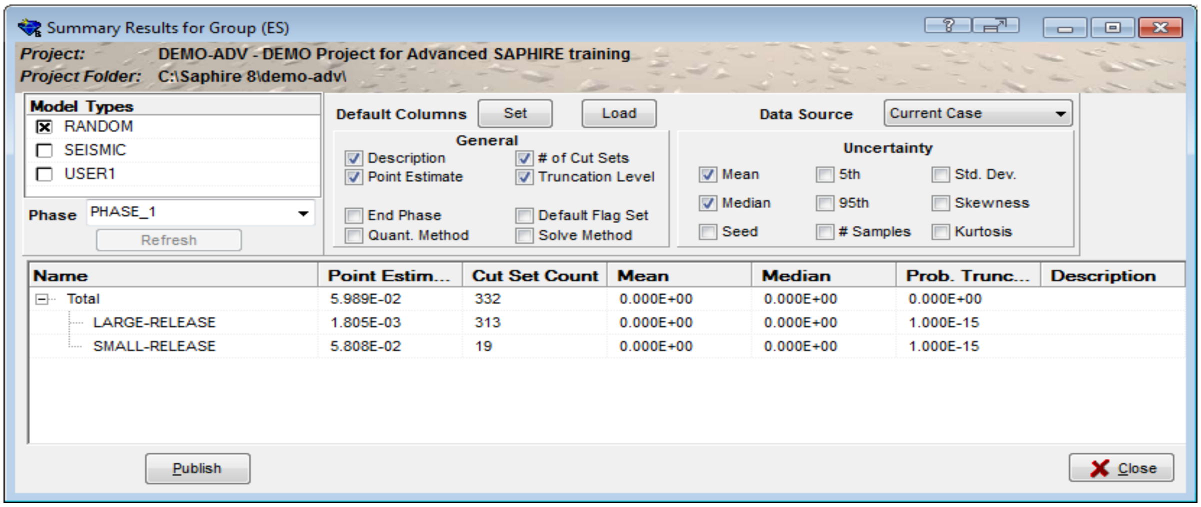


The report created this way is viewed as html and can be printed or saved:

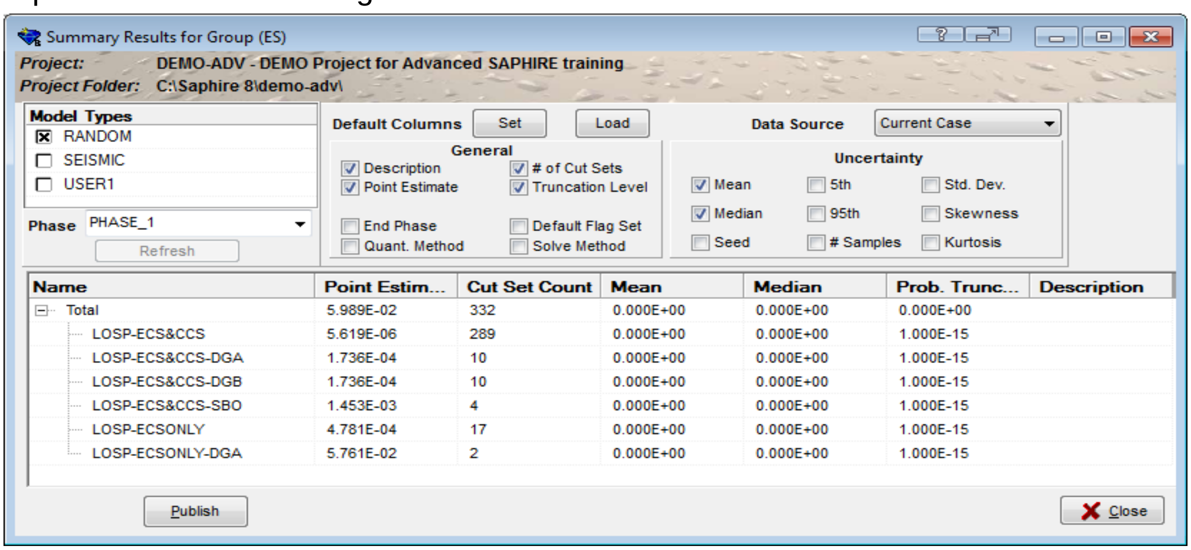


A **Summary** report type is available by highlighting the end states desired in the **End States** list panel, right clicking to open the list menu and selecting **View Summary Results**.

With the end states gathered without the partitioning rules and gathered through Sequence End States, the LARGE-RELEASE and SMALL-RELEASE end states with the uncertainty analysis performed previously generates this report:



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

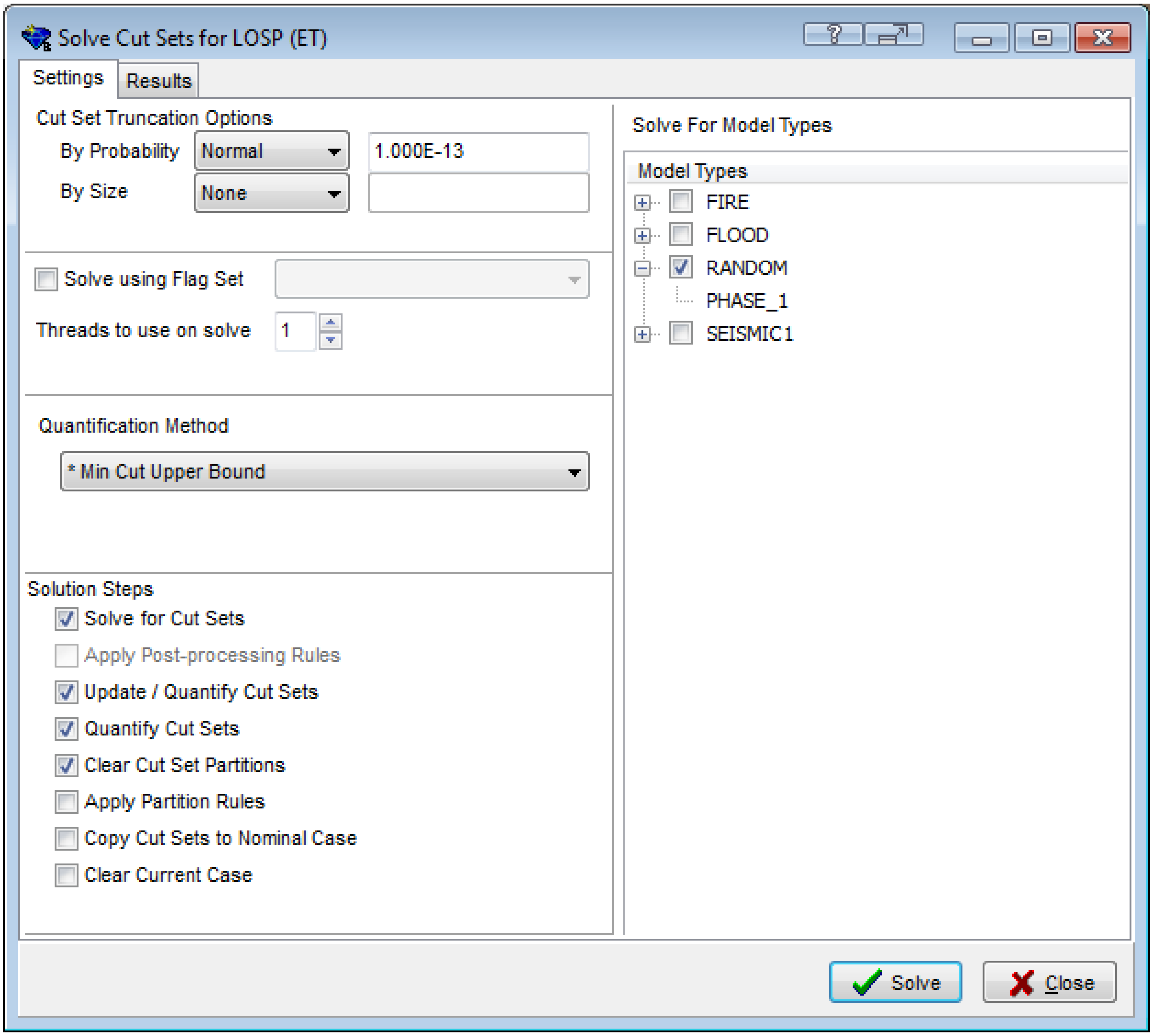


Note:

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.

6.7. Resetting or Deleting Partition Rule End States

To reset the end states created using the partitioning rules, solve the event trees with the **Apply Partition Rules** check box unchecked and the **Clear Cut Set Partitions** box checked. This will not delete the partitioned end states, just clear the cut sets from them.



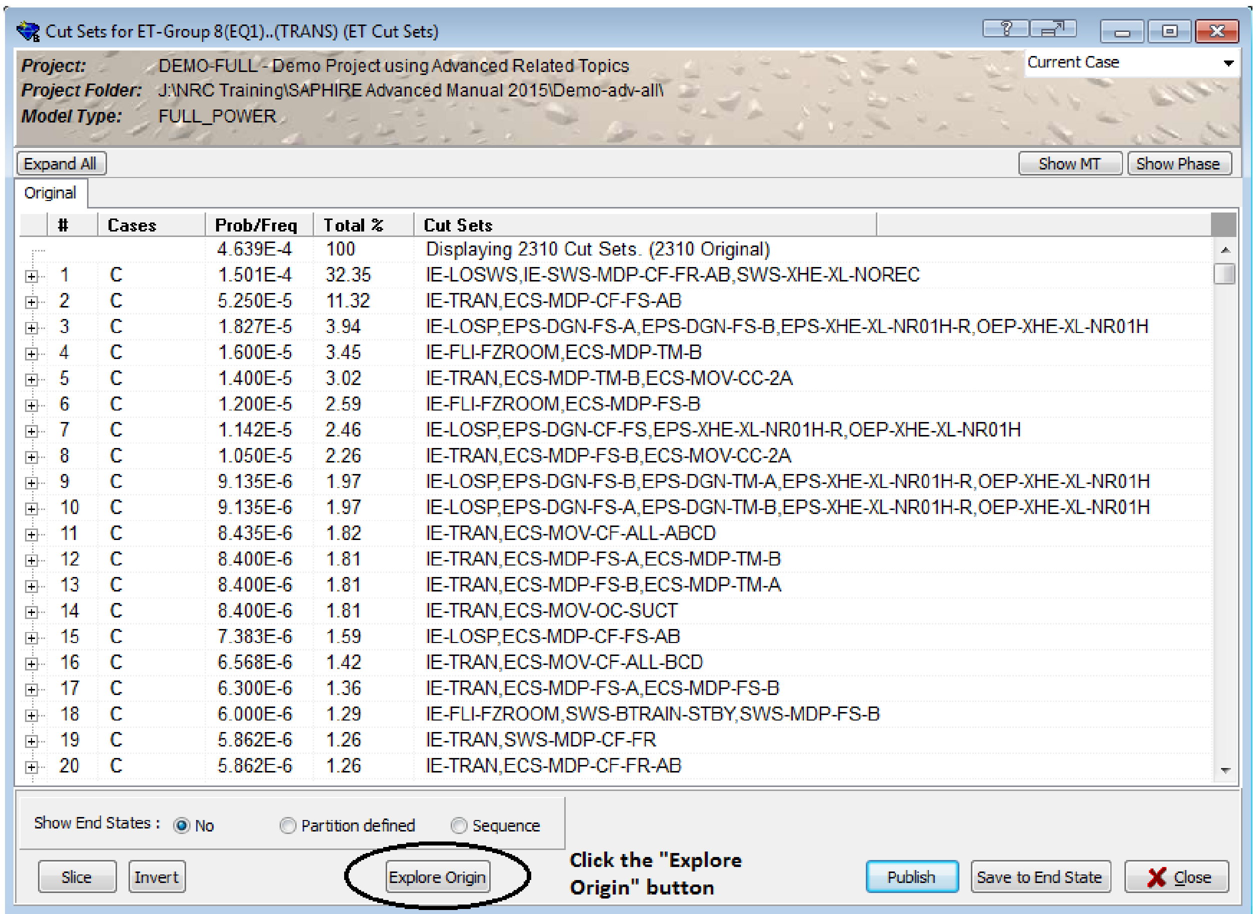
To delete the partitioned end states the event trees need unlinked by highlighting the event tree in the list pane and right clicking to bring up the menu, then selecting **Unlink**. Similarly, perform a **Link** again on the event tree and the partitioned end states can then be deleted.

6.8. Explore Origin from End States

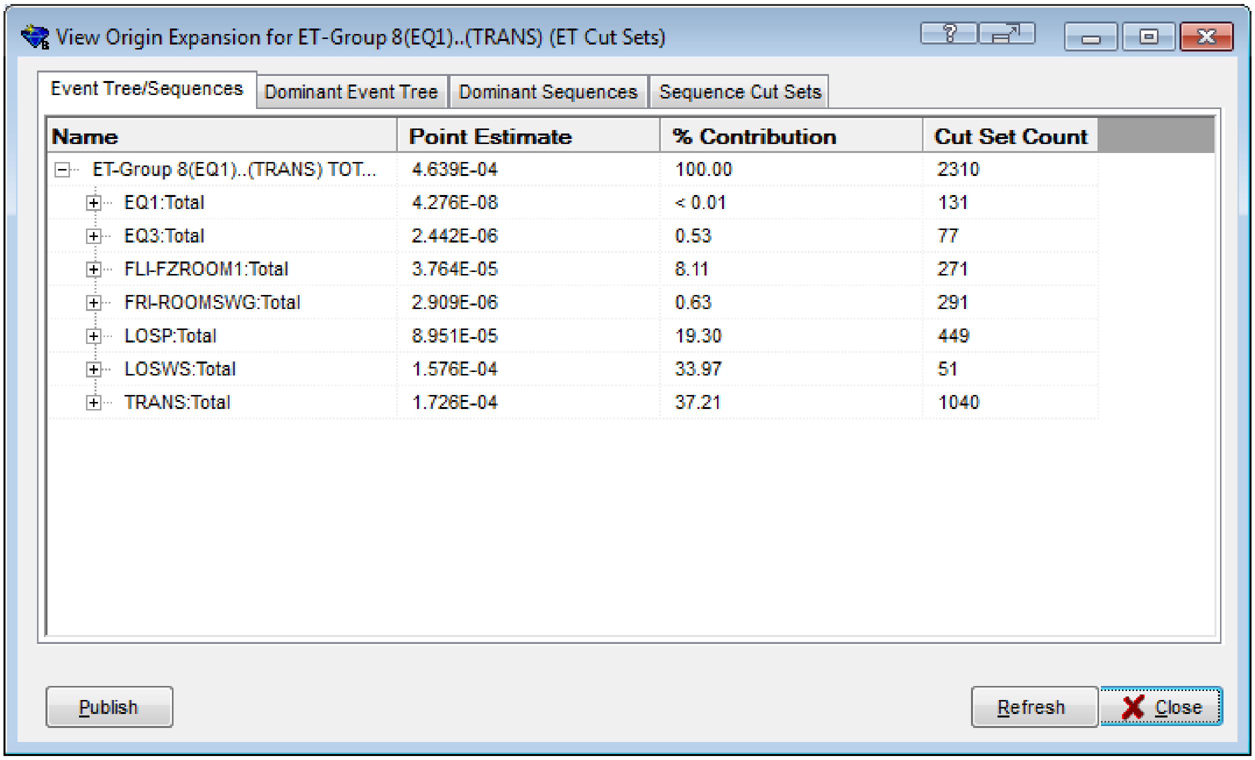
The Explore Origin option from the end state display is designed to provide the analyst with information on where the cut set originated. This origination is tied to the event tree and the event tree accident sequence. From any event tree cut set view option or end state cut set view option, the explore origin option can be executed.

- ♦ To execute the explore origin option, open up any cut set viewer that contains accident sequence cut sets (event tree or end state).

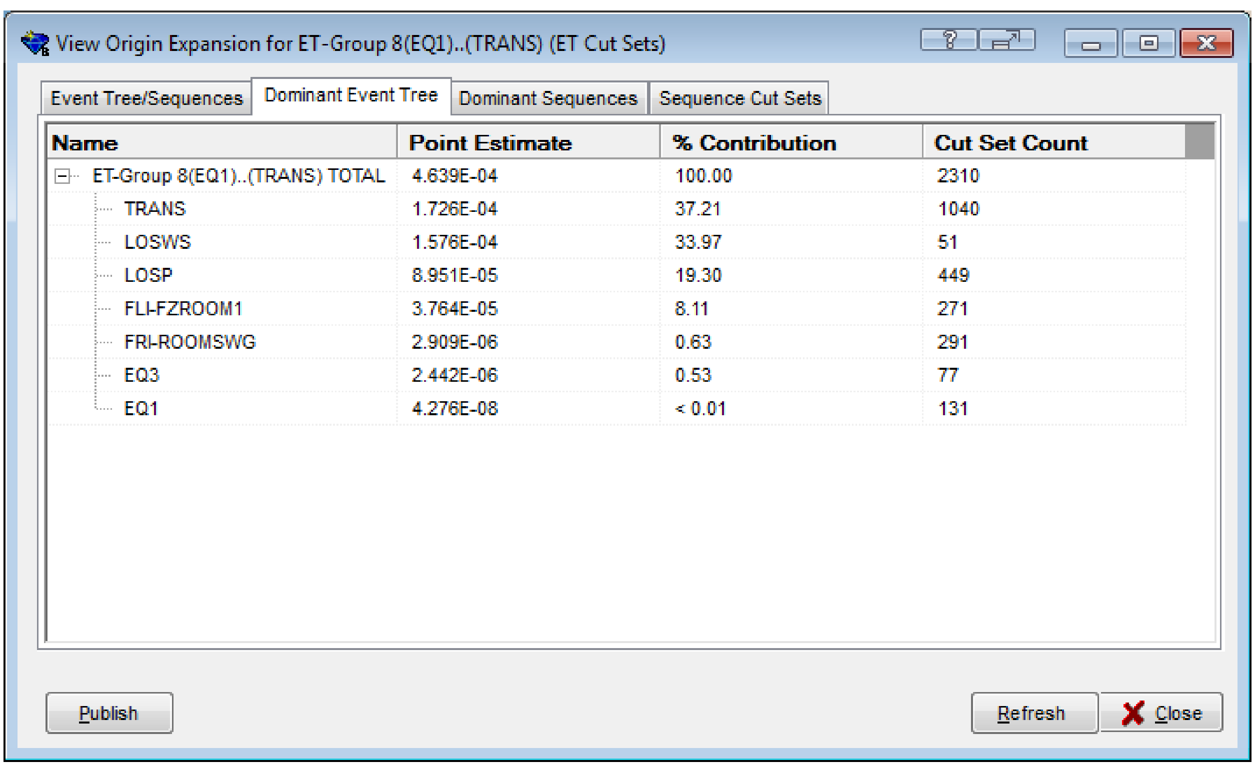
  - ◊ Highlight an event tree, select **View Cut Sets**



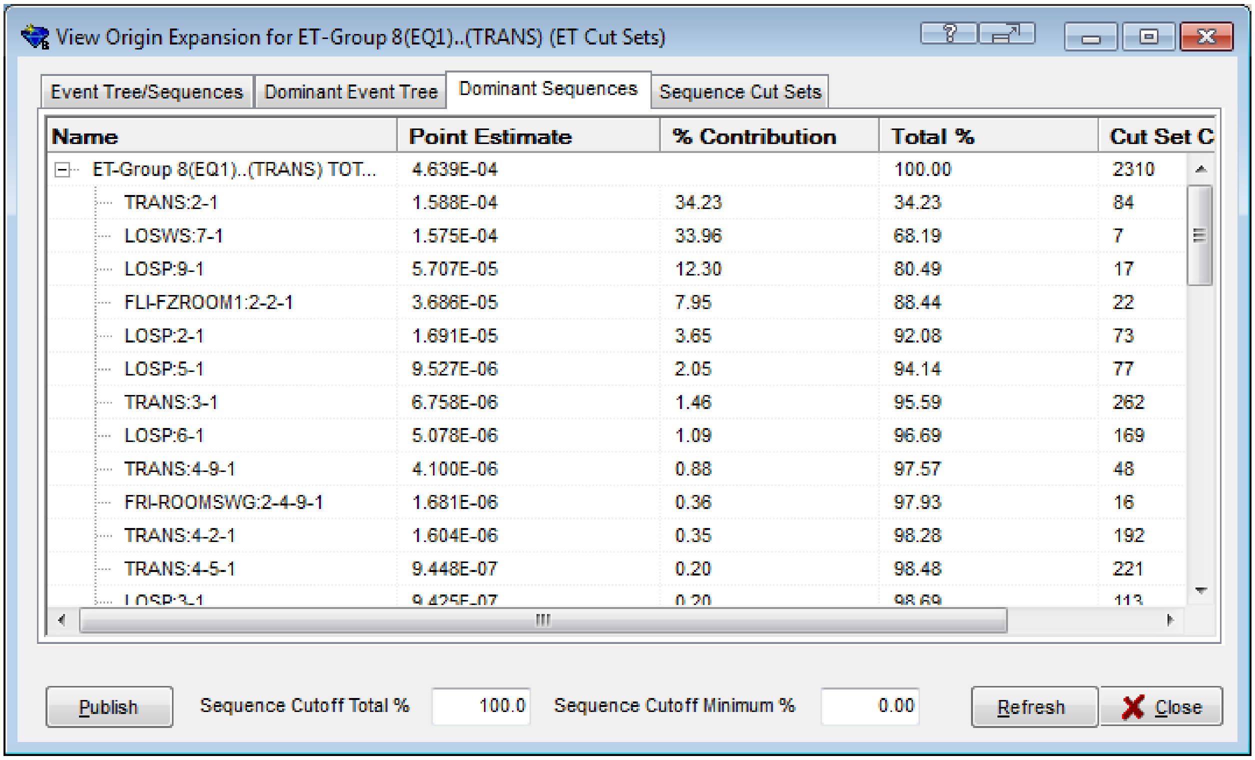
- ♦ The display option lists all of the event trees that the cut sets can be located. The first tab is just an overview of the total results. This tab lists each event tree and their frequency along with the number of cut sets. Each event tree can be expanded out to show their individual sequence by clicking the button.



♦ The next tab “Dominant Event Tree”, sorts the event tree based on its contribution to the overall results.



- The third tab “Dominant Sequences” expands out all of the event tree sequences and sorts them by dominant contribution.



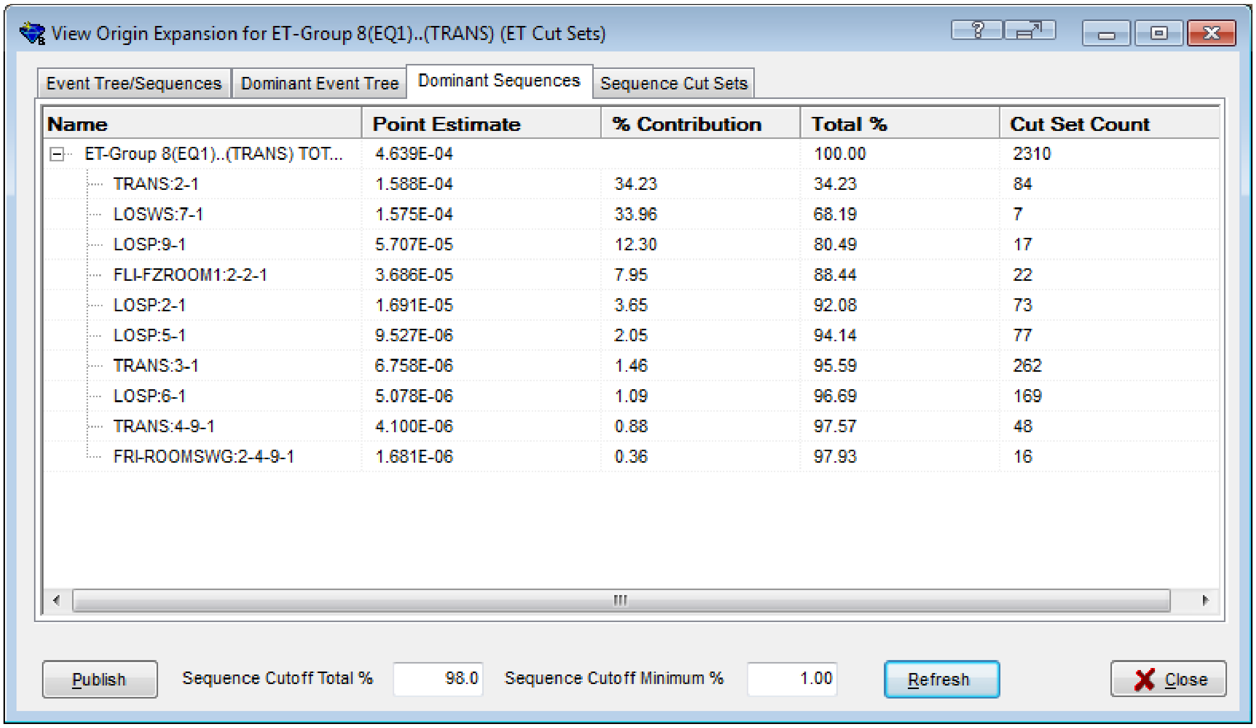
Name	Point Estimate	% Contribution	Total %	Cut Set C
ET-Group 8(EQ1)..(TRANS) TOT...	4.639E-04		100.00	2310
TRANS:2-1	1.588E-04	34.23	34.23	84
LOSWS:7-1	1.575E-04	33.96	68.19	7
LOSP:9-1	5.707E-05	12.30	80.49	17
FLI-FZROOM1:2-2-1	3.686E-05	7.95	88.44	22
LOSP:2-1	1.691E-05	3.65	92.08	73
LOSP:5-1	9.527E-06	2.05	94.14	77
TRANS:3-1	6.758E-06	1.46	95.59	262
LOSP:6-1	5.078E-06	1.09	96.69	169
TRANS:4-9-1	4.100E-06	0.88	97.57	48
FRI-ROOMSWG:2-4-9-1	1.681E-06	0.36	97.93	16
TRANS:4-2-1	1.604E-06	0.35	98.28	192
TRANS:4-5-1	9.448E-07	0.20	98.48	221
LOSP:3-1	9.425E-07	0.20	98.69	113

- This tab has options on the bottom to provide the analyst additional filters (both filters need to be adjusted at the same time).

  ◊ The first filter is Sequence Cutoff Total % - this filter provides a means to display only those sequences that contribute X% to the overall total.

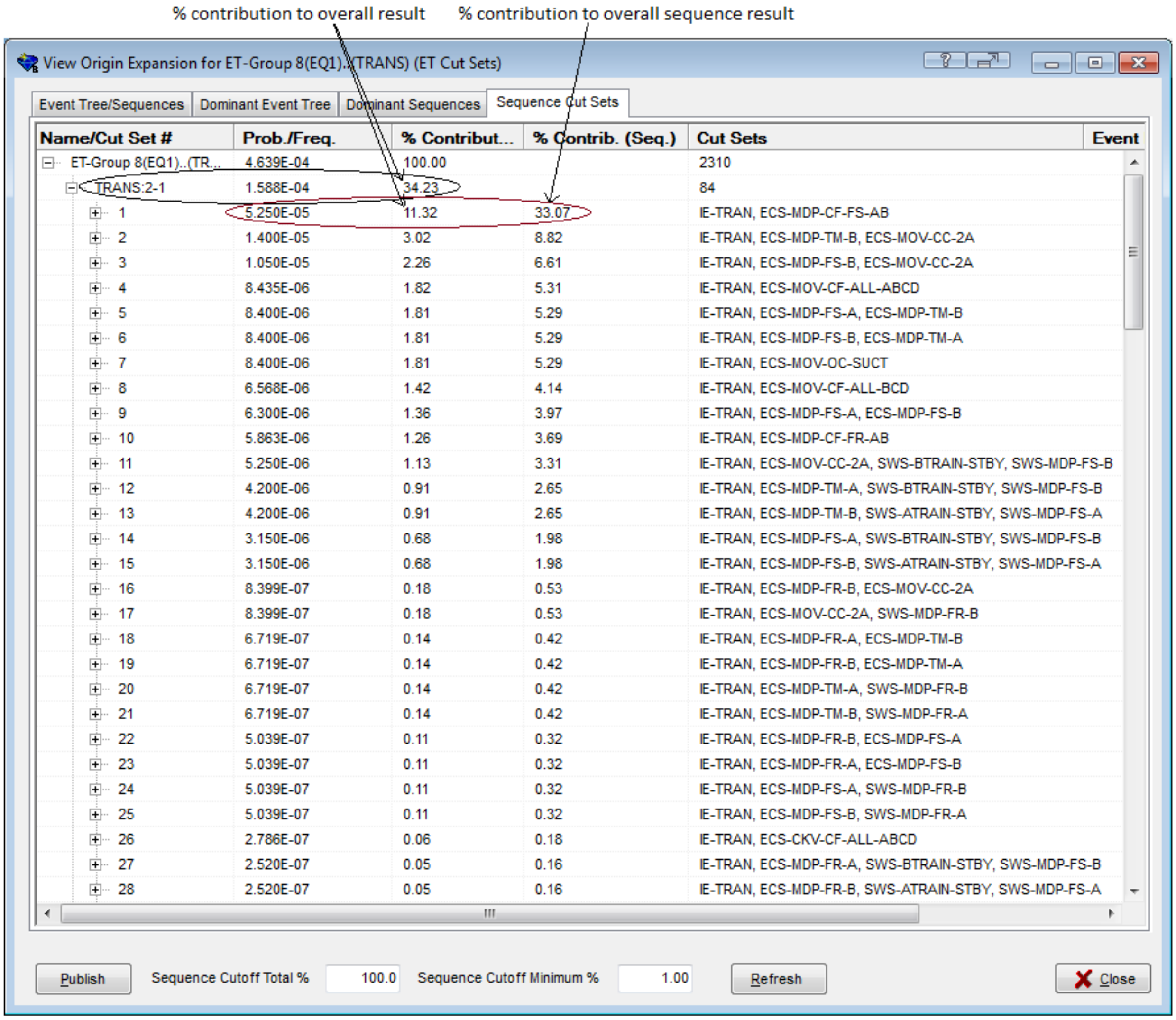
  ◊ The second filter is Sequence Cutoff Minimum % - this filter provides a means to display on those sequences that are above Y% to the overall total.

For example, if the top 98% sequences are required for display, set Sequence Cutoff Total% to 98 and then just set Sequence Cutoff Minimum% to 1.



♦ This last tab “Sequence Cut Sets” provides the cut sets for the dominant sequences. This tab provides the overall contribution of the sequence plus it looks at each individual cut set and provides its contribution to both the overall results and its contribution to the overall sequence result.

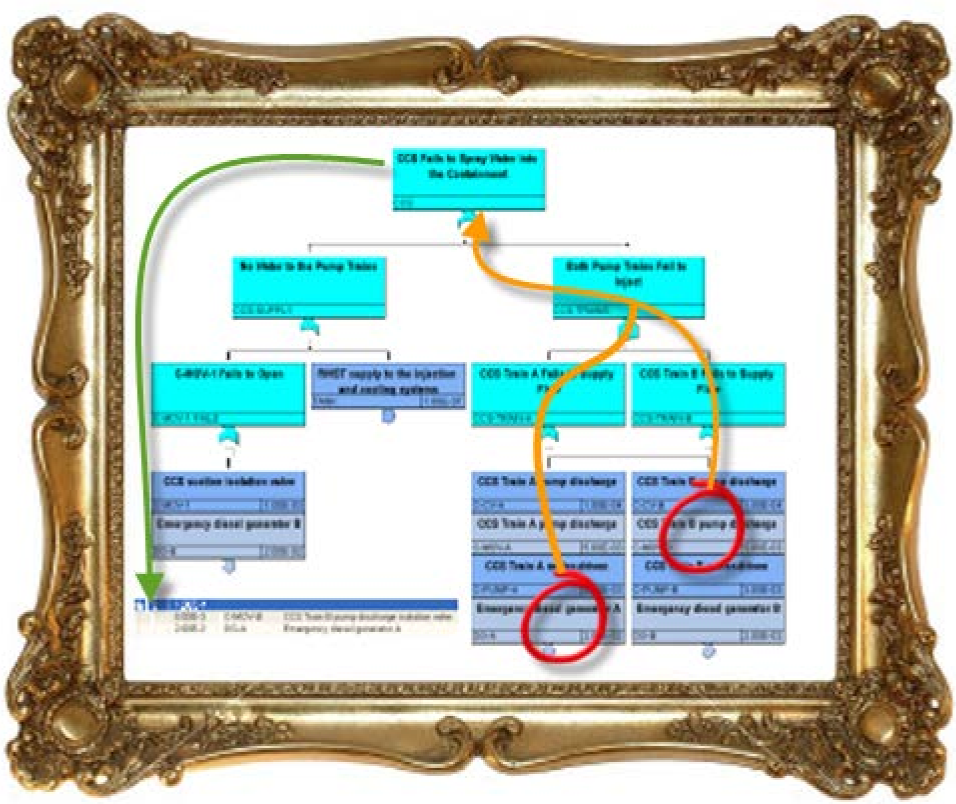
◊ The individual cut sets can be expanded out to display the individual basic event’s probability and description.



Name/Cut Set #	Prob./Freq.	% Contribut...	% Contrib. (Seq.)	Cut Sets	Event
ET-Group 8(EQ1)..(TR...	4.639E-04	100.00		2310	
TRANS:2-1	1.588E-04	34.23		84	
1	5.250E-05	11.32	33.07	IE-TRAN, ECS-MDP-CF-FS-AB	
2	1.400E-05	3.02	8.82	IE-TRAN, ECS-MDP-TM-B, ECS-MOV-CC-2A	
3	1.050E-05	2.26	6.61	IE-TRAN, ECS-MDP-FS-B, ECS-MOV-CC-2A	
4	8.435E-06	1.82	5.31	IE-TRAN, ECS-MOV-CF-ALL-ABCD	
5	8.400E-06	1.81	5.29	IE-TRAN, ECS-MDP-FS-A, ECS-MDP-TM-B	
6	8.400E-06	1.81	5.29	IE-TRAN, ECS-MDP-FS-B, ECS-MDP-TM-A	
7	8.400E-06	1.81	5.29	IE-TRAN, ECS-MOV-OC-SUCT	
8	6.568E-06	1.42	4.14	IE-TRAN, ECS-MOV-CF-ALL-BCD	
9	6.300E-06	1.36	3.97	IE-TRAN, ECS-MDP-FS-A, ECS-MDP-FS-B	
10	5.863E-06	1.26	3.69	IE-TRAN, ECS-MDP-CF-FR-AB	
11	5.250E-06	1.13	3.31	IE-TRAN, ECS-MOV-CC-2A, SWS-BTRAIN-STBY, SWS-MDP-FS-B	
12	4.200E-06	0.91	2.65	IE-TRAN, ECS-MDP-TM-A, SWS-BTRAIN-STBY, SWS-MDP-FS-B	
13	4.200E-06	0.91	2.65	IE-TRAN, ECS-MDP-TM-B, SWS-ATRAIN-STBY, SWS-MDP-FS-A	
14	3.150E-06	0.68	1.98	IE-TRAN, ECS-MDP-FS-A, SWS-BTRAIN-STBY, SWS-MDP-FS-B	
15	3.150E-06	0.68	1.98	IE-TRAN, ECS-MDP-FS-B, SWS-ATRAIN-STBY, SWS-MDP-FS-A	
16	8.399E-07	0.18	0.53	IE-TRAN, ECS-MDP-FR-B, ECS-MOV-CC-2A	
17	8.399E-07	0.18	0.53	IE-TRAN, ECS-MOV-CC-2A, SWS-MDP-FR-B	
18	6.719E-07	0.14	0.42	IE-TRAN, ECS-MDP-FR-A, ECS-MDP-TM-B	
19	6.719E-07	0.14	0.42	IE-TRAN, ECS-MDP-FR-B, ECS-MDP-TM-A	
20	6.719E-07	0.14	0.42	IE-TRAN, ECS-MDP-TM-A, SWS-MDP-FR-B	
21	6.719E-07	0.14	0.42	IE-TRAN, ECS-MDP-TM-B, SWS-MDP-FR-A	
22	5.039E-07	0.11	0.32	IE-TRAN, ECS-MDP-FR-B, ECS-MDP-FS-A	
23	5.039E-07	0.11	0.32	IE-TRAN, ECS-MDP-FR-A, ECS-MDP-FS-B	
24	5.039E-07	0.11	0.32	IE-TRAN, ECS-MDP-FS-A, SWS-MDP-FR-B	
25	5.039E-07	0.11	0.32	IE-TRAN, ECS-MDP-FS-B, SWS-MDP-FR-A	
26	2.786E-07	0.06	0.18	IE-TRAN, ECS-CKV-CF-ALL-ABCD	
27	2.520E-07	0.05	0.16	IE-TRAN, ECS-MDP-FR-A, SWS-BTRAIN-STBY, SWS-MDP-FS-B	
28	2.520E-07	0.05	0.16	IE-TRAN, ECS-MDP-FR-B, SWS-ATRAIN-STBY, SWS-MDP-FS-A	

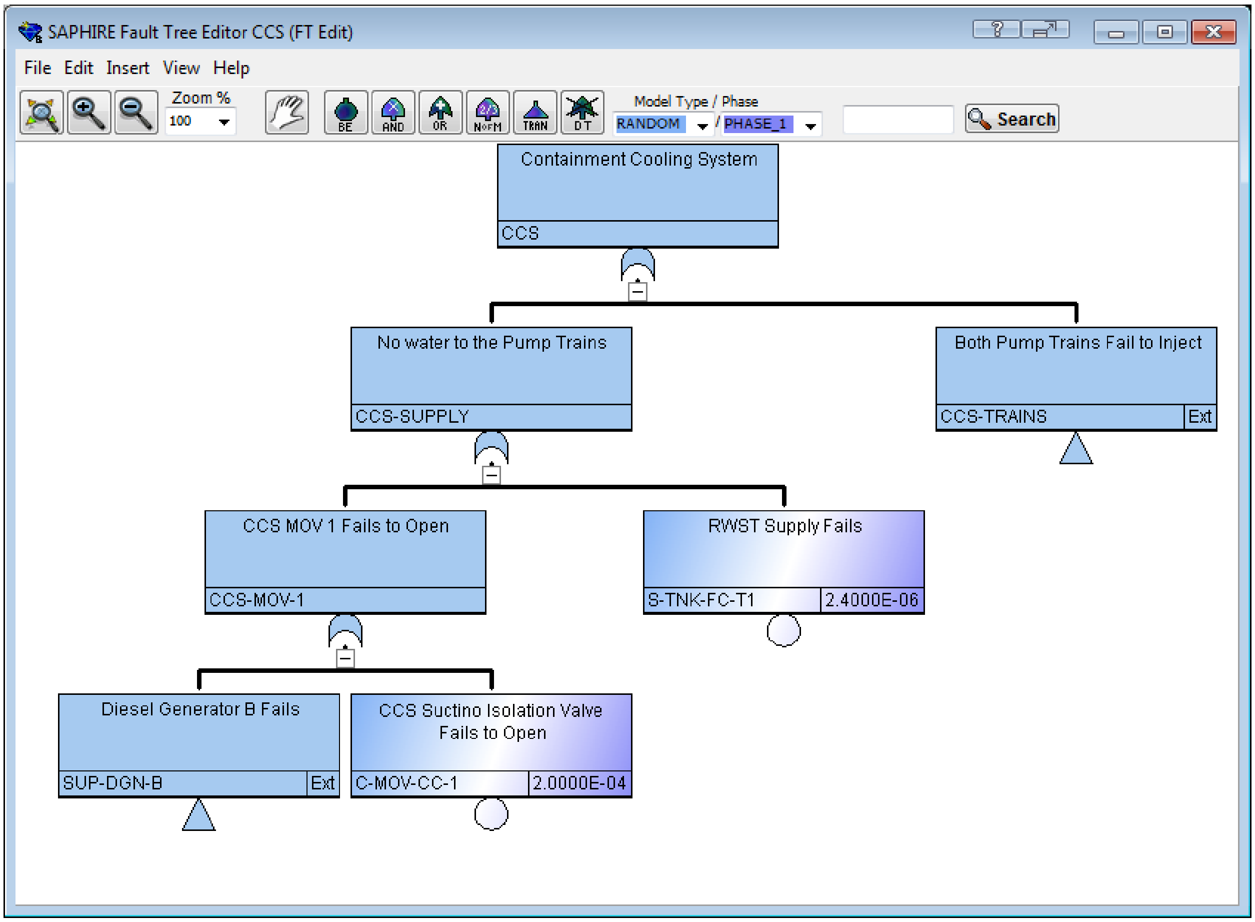
| 7 | SOLVING FAULT TREE CUT SETS

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.



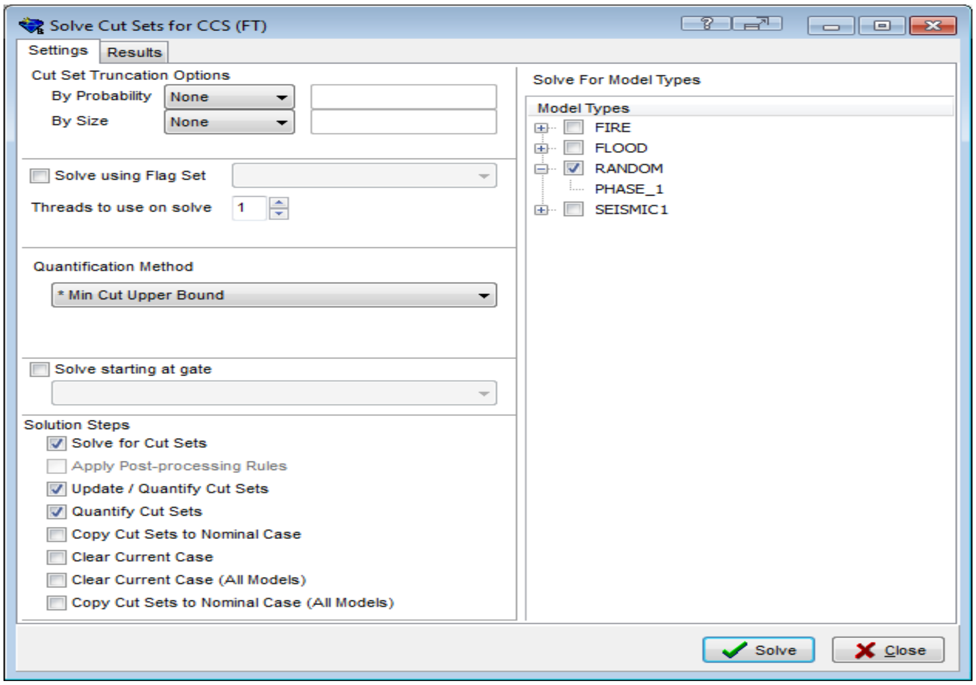
Examples of Fault Tree Solve Options

The CCS Fault Tree shown below will be used to demonstrate the various solve options. The CCS-TRAINS portion of the CCS fault tree logic has been paged out into its own sub-tree. By breaking the CCS fault tree into the two fault trees, the "Developed Event" options will demonstrated.

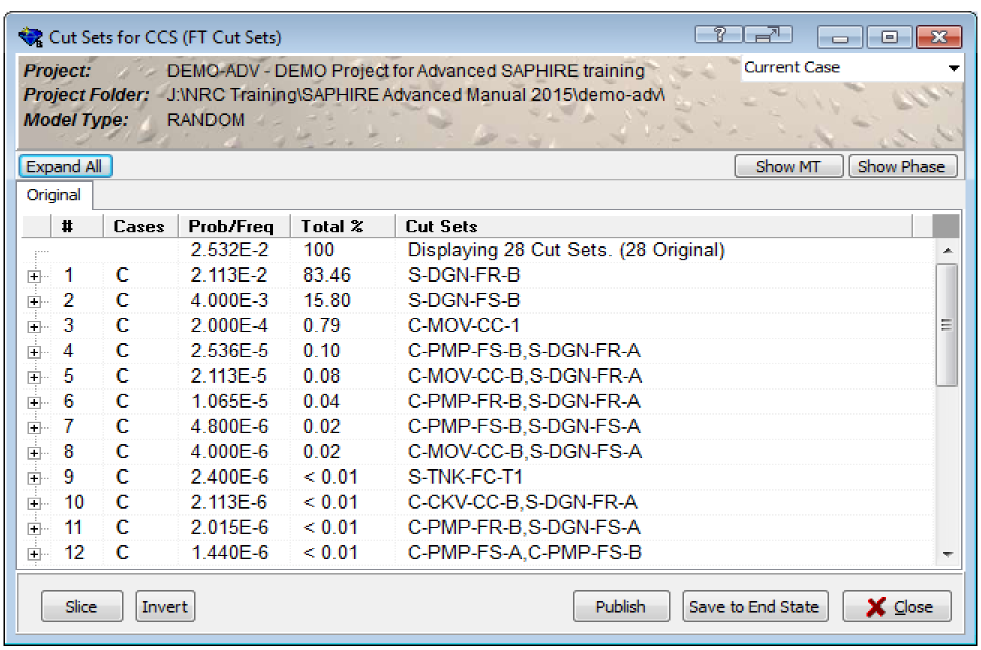


7.1. Fault Tree Cut Sets With No Truncation

To generate cut sets without truncation, highlight the fault tree desired in the Fault Tree list panel, right click and select **Solve**. Then select “**None”** in the drop-down menu for **Cut Set Truncation**.



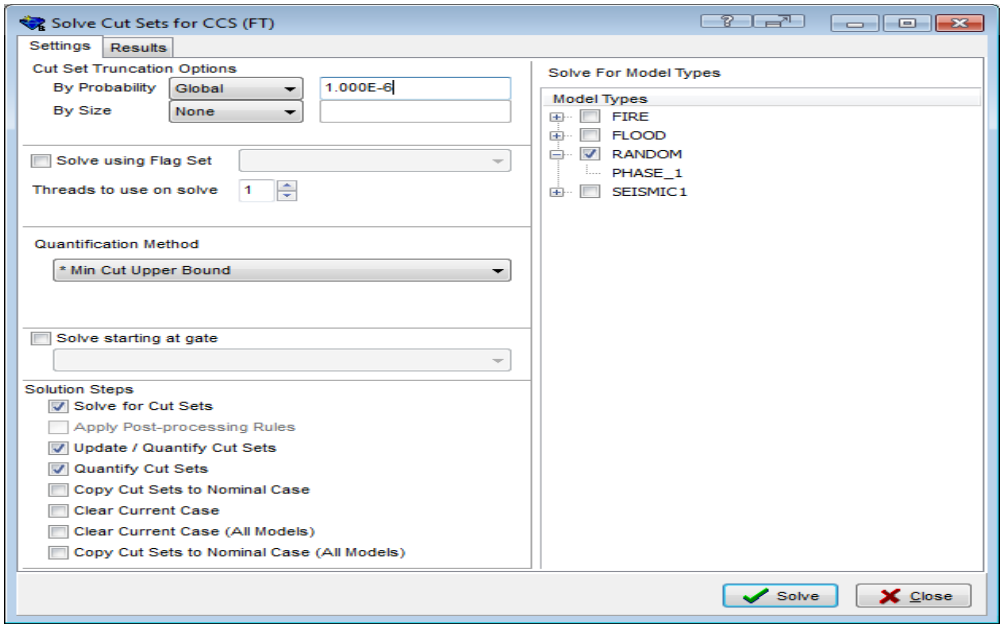
Click the **Solve** button to solve the cut sets, then clicking on **Cut Sets** on the following window:



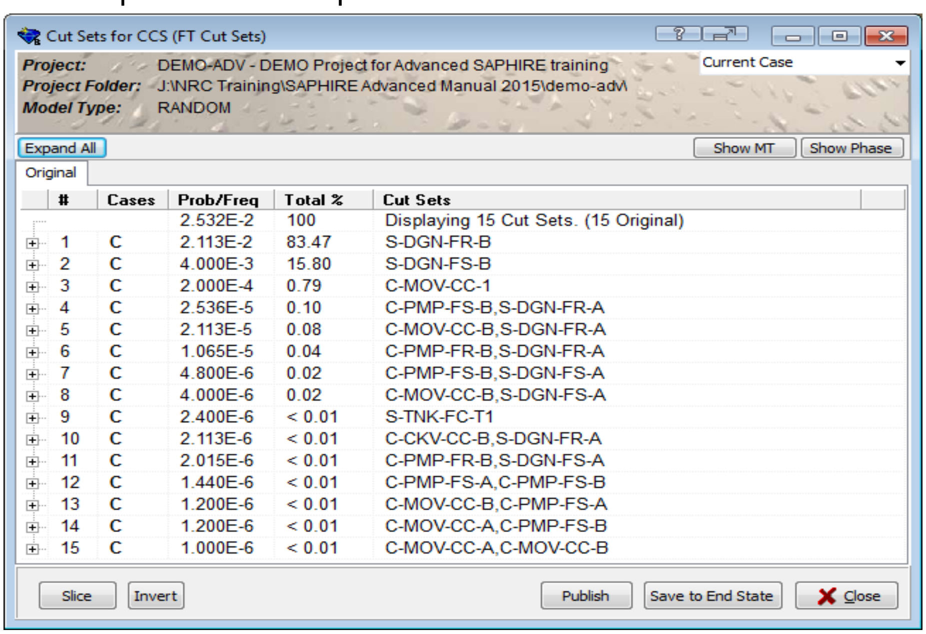
A printable HTML report can be generated by clicking on the **Publish** button.

7.2. Fault Tree Cut Sets With Probability Truncations

The default truncation is Global, which will always be active when the Solve form is displayed. The Global truncation setting is set from the main menu **Project → User Settings → Analysis options**. To use a different truncation enter the truncation value desired for this case, use 1.00E-06.



Click the **Solve** button to solve the cut sets, then clicking on **Cut Sets** on the following window provides this report:

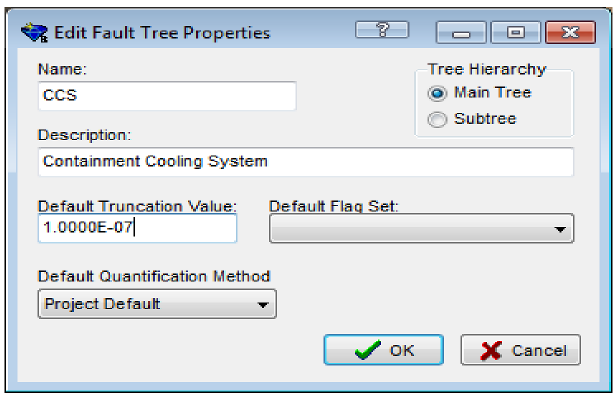


A printable HTML report can be generated by clicking on the **Publish** button.

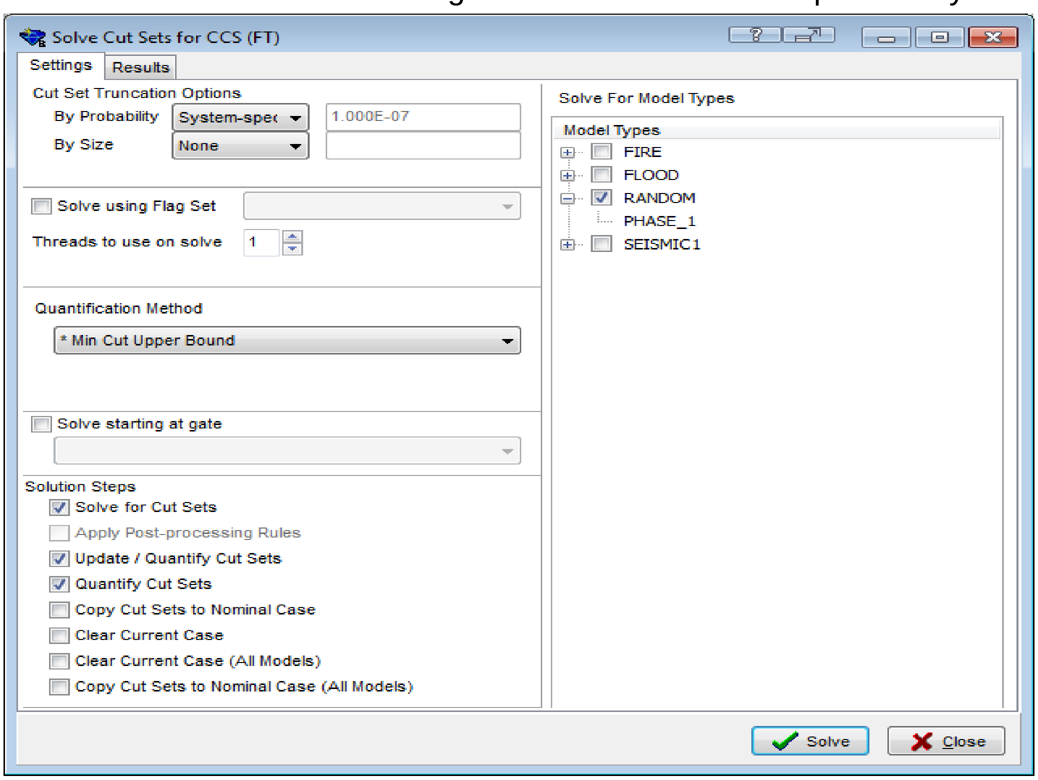
7.3. Fault Tree Specific Probability Truncation

To use a fault tree specific truncation (i.e., one that will be used instead of the Global truncation and be specific to just that fault tree), the truncation probability needs to be applied to the fault tree by highlighting the fault tree and selecting **Edit Properties**.

In the “Default Truncation Value:” field, type in the fault tree specific truncation value. This value will be used when the fault tree cut sets are generated if the Global truncation option is not specified.



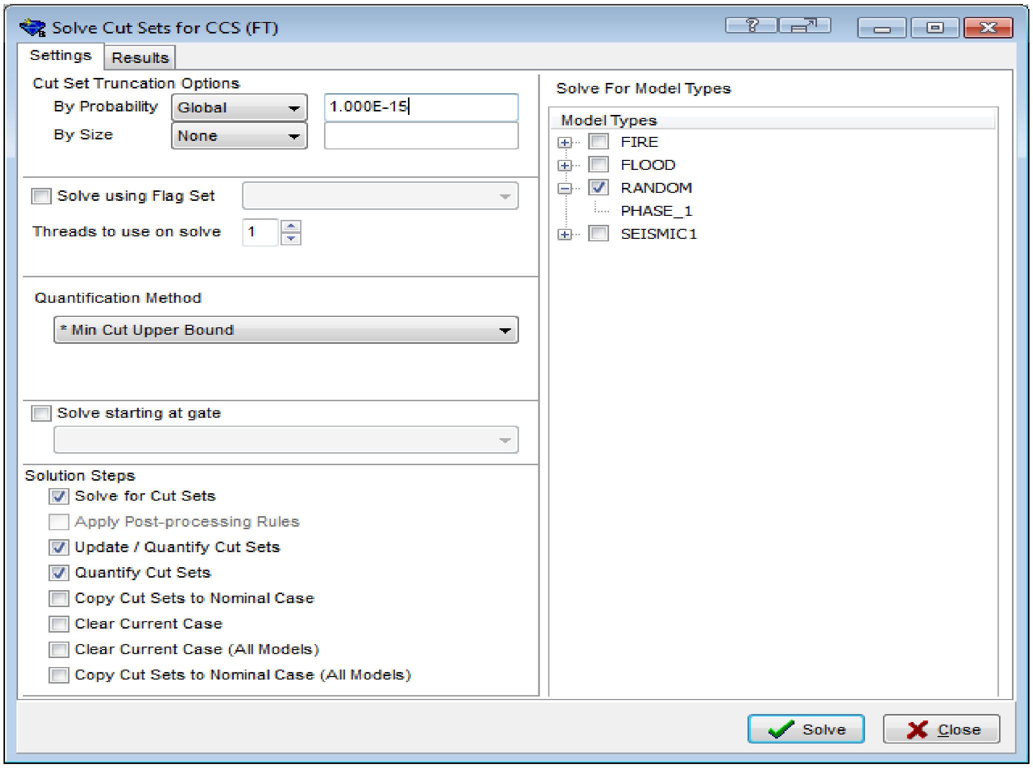
To solve the fault tree using its fault tree specific truncation, highlight the fault tree click the Solve button and then change the Global truncation option to System Specific.



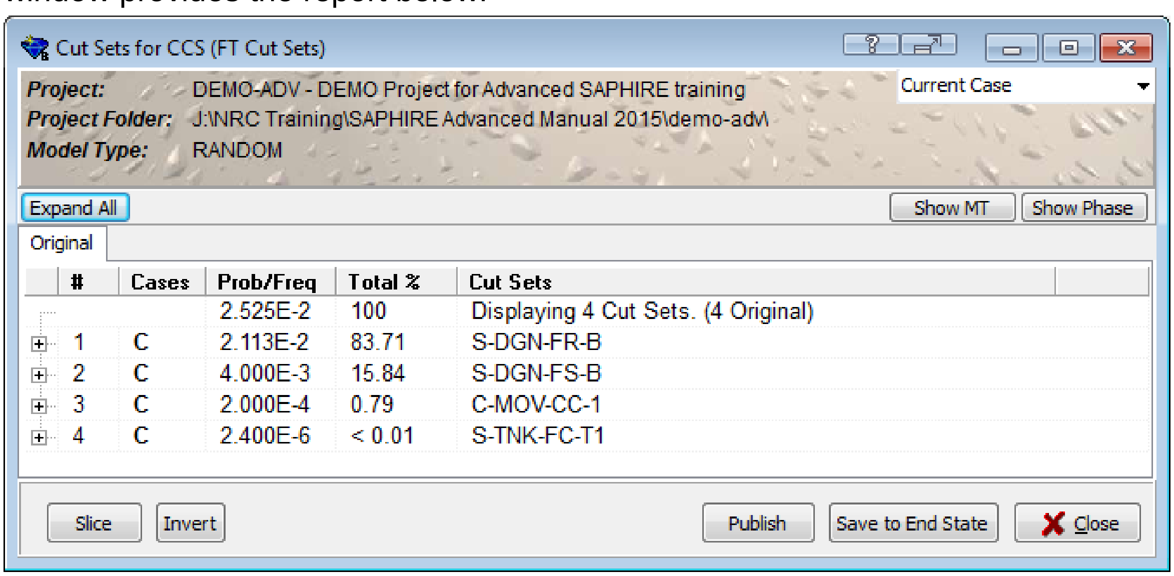
Click the **Solve** button to solve the cut sets.

7.4. Fault Tree Cut Sets with Size Truncation

To generate cut sets with truncation on the number of events in a cut set, select **Size** from the **Size Truncation** pull-down menu of the Fault Tree Solve form and enter the size cutoff value (a 1 in our example). Cut Set Truncation is left at the Global value of 1.00E-15 for this example.

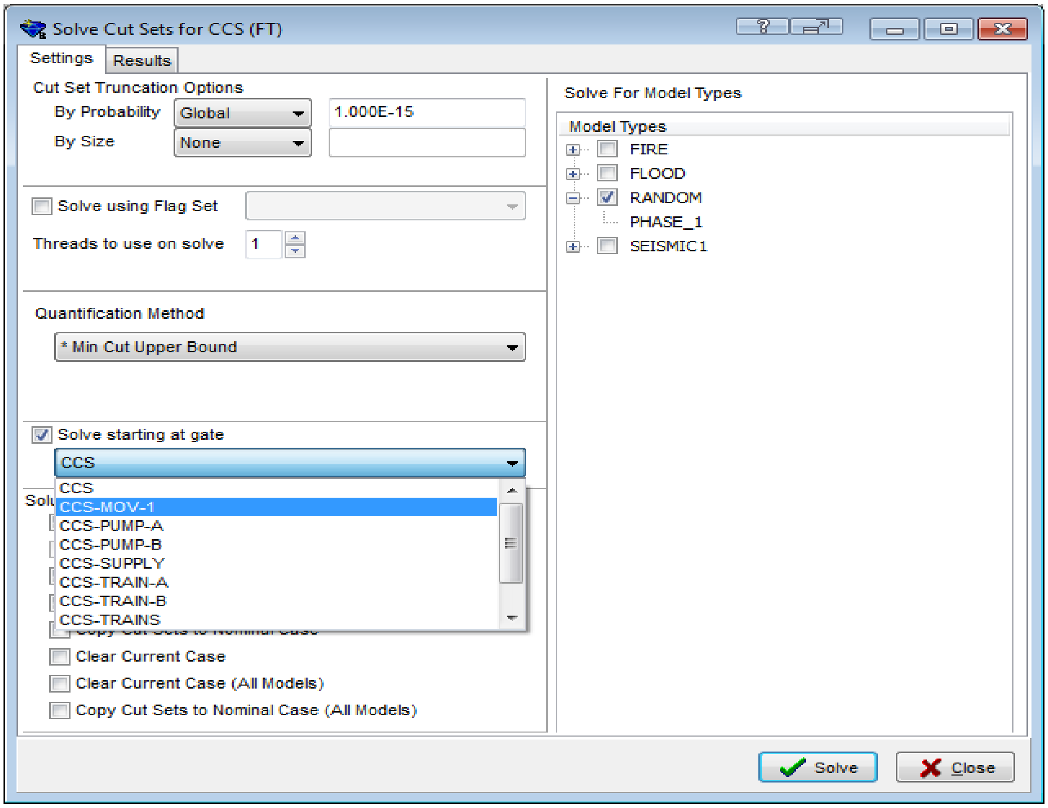


Click the **Solve** button to solve the cut sets, then clicking on **Cut Sets** on the following window provides the report below:

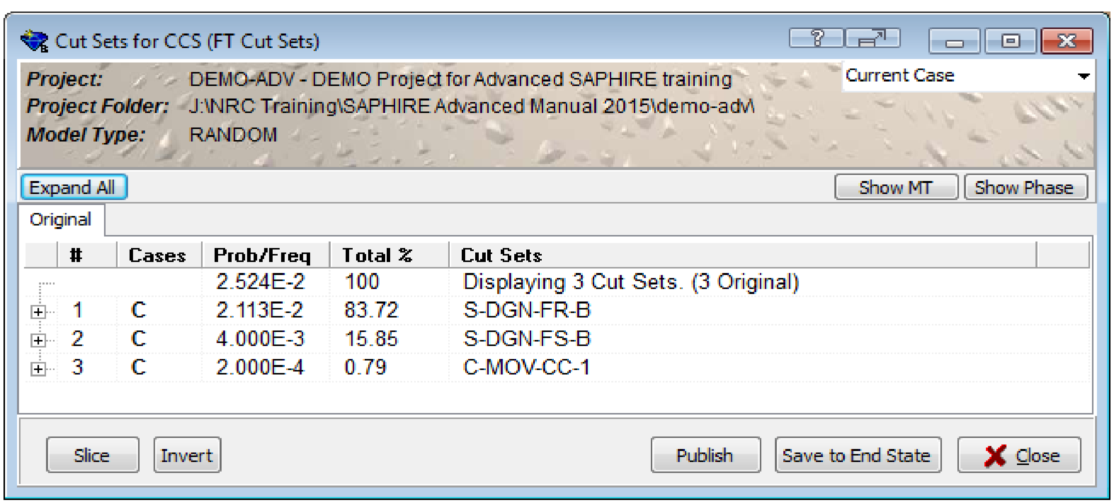


7.5. Analyzing Fault Tree "Gate Level"

To generate cut sets beginning with a gate below the top gate, enter the gate name by selecting the “Solve starting at gate” check box. A drop down box will appear which lists all of the gates in the fault tree. From this list, select the gate that SAPHIRE is going to use as the starting point to generate cut sets.

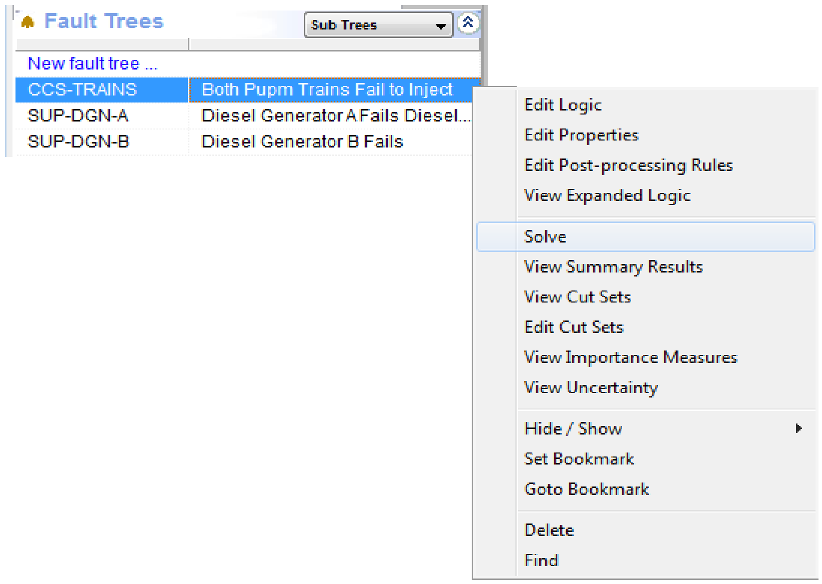


When a gate is selected as the starting gate, SAPHIRE generates cut sets and stores these cut sets for the fault tree selected (e.g., CCS).

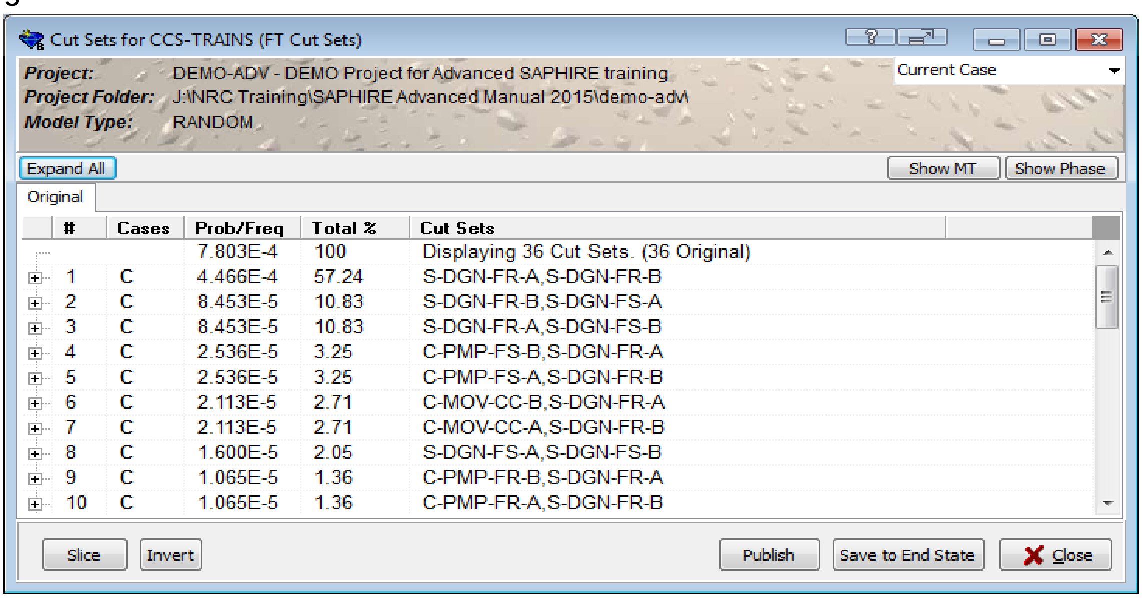


7.6. Analyzing Fault Tree "Sub-trees"

To generate cut sets for a sub-fault tree, the Fault Tree panel drop down box needs to be changed from Main Trees to Sub Trees. Highlight the appropriate sub-tree, right click and solve as normal. The list panel sorts and displays the fault trees based on filter type that is selected from the pull down menu (e.g., Sub-Trees, which in the example includes CCS-TRAINS).



In this example, the cut set probability truncation is the Global value and the cut sets are generated for the sub tree CCS-TRAINS.



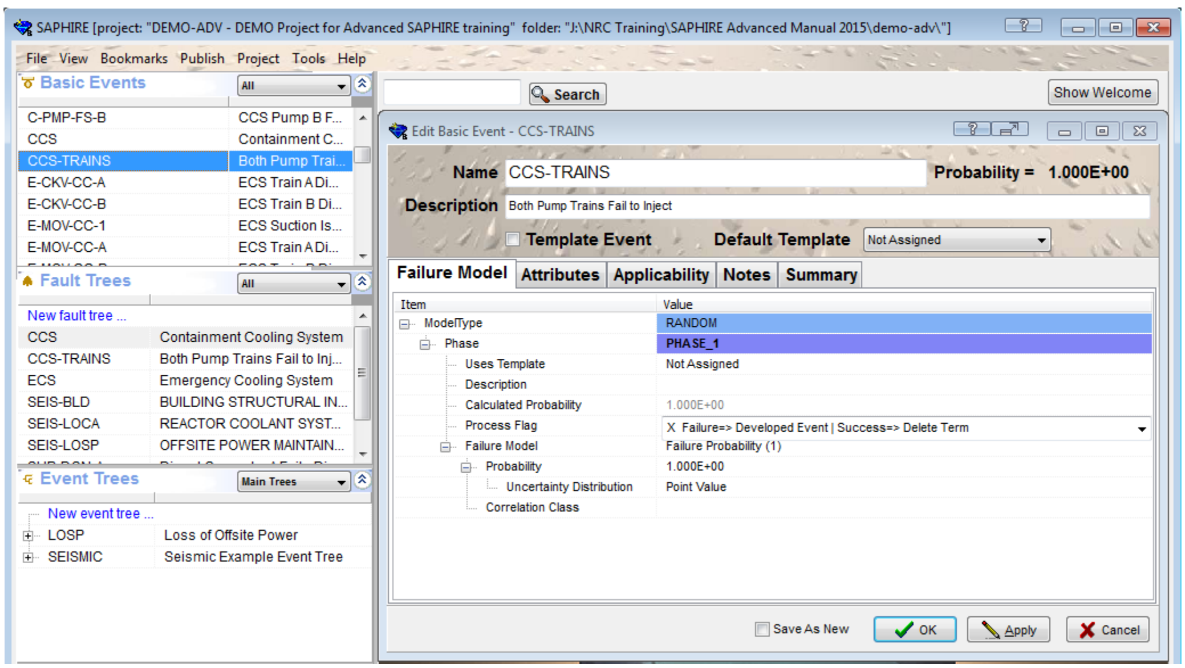
Note that DG-B related failure modes appear in several cut sets when the CCS-TRAINS sub tree is analyzed. Based on the starting gate, CCS-TRAINS, the cut sets are minimal and therefore, DG-B shows up in multiple cut sets. However, when the top gate, CCS, is the starting point, DG-B is a single event and all combinations of DG-B become non-minimal cut sets. After Boolean reduction, all cut sets shown above that contain DG-B will be subsumed and removed from the list.

7.7. Treating a Fault Tree Gate as a Basic Event

Now, we will solve the CCS tree while treating the CCS-TRAINS sub-tree as a basic event (rather than using its logic structure).

There are a couple of ways to set the CCS-TRAINS sub-tree as though it were a basic event.

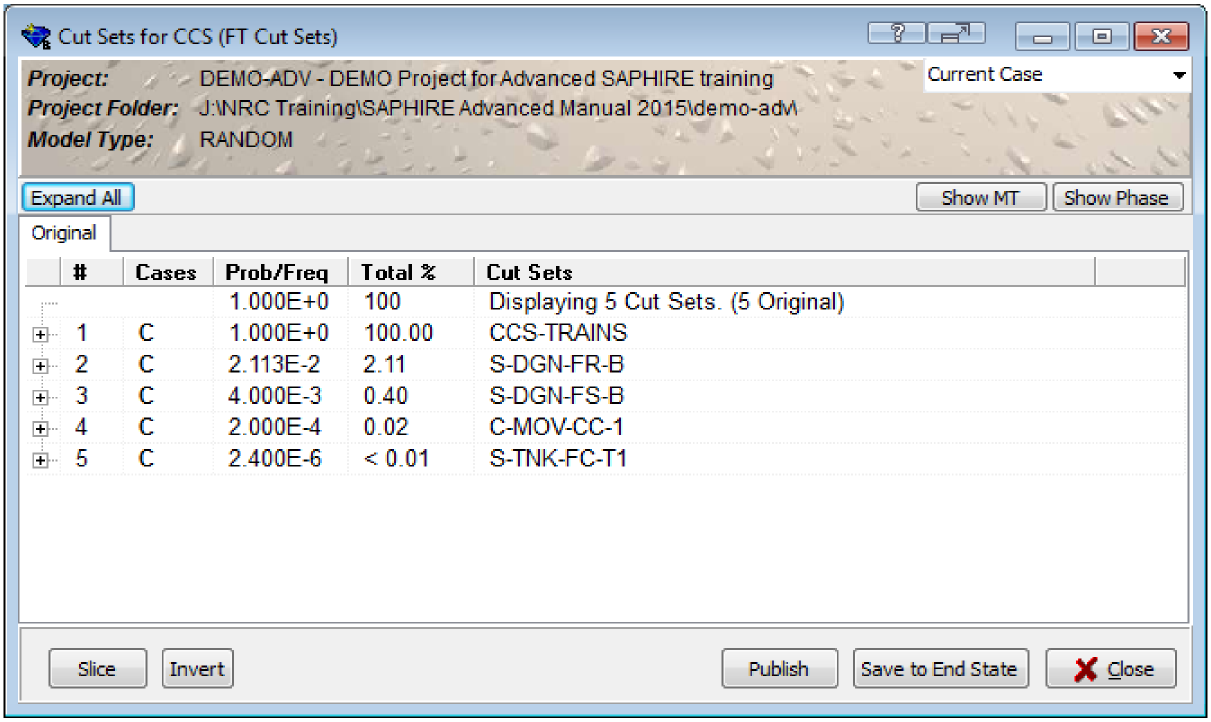
1. Set its *Process Flag* to the “X” type in the **Edit Basic Event** form (this is a permanent change):



2. Make a Change Set in the Change Set list panel or from the main menu **Project → Change Sets** to set the CCS-TRAINS event’s process flag is set to “X” (temporary change).

   - Both methods allow any probability to 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 (termed Develop Event). As such, they can be edited with the methods noted.

Solving the CCS Fault Tree in the normal manner with the Global truncation produces the following report:



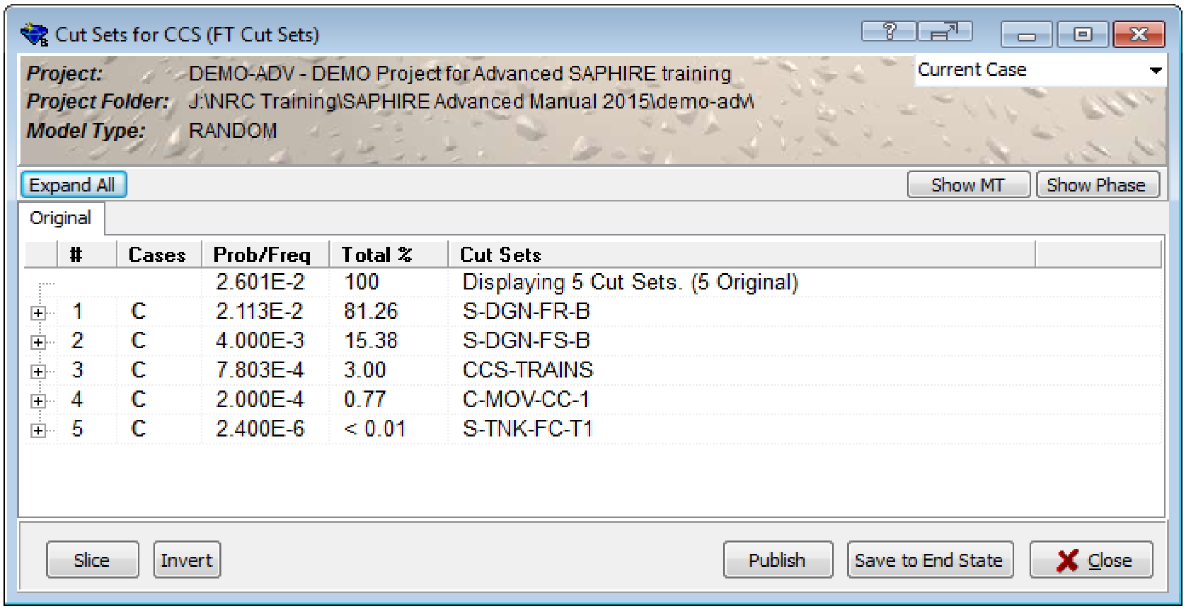
7.8. Treating a Fault Tree Gate as a Basic Event with an Appropriate Probability

In this example, we will treat the CCS-TRAINS sub-tree 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. Make sure CCS-TRAINS basic event Calculation type is “Failure Probability (1)” and Process Flag to “Failure=>System Logic | Success=> Delete Term”
2. Generate cut sets for the CCS-TRAINS fault trees
3. Modify the CCS-TRAINS basic event’s Calculation Type to “Use mincut upper bound …(S)” and the Process Flag to “X” using the Edit Basic Event form for CCS-TRAINS in the Basic Event list pane.

Now, solve the CCS fault tree without truncation. Reporting the solve results shows:

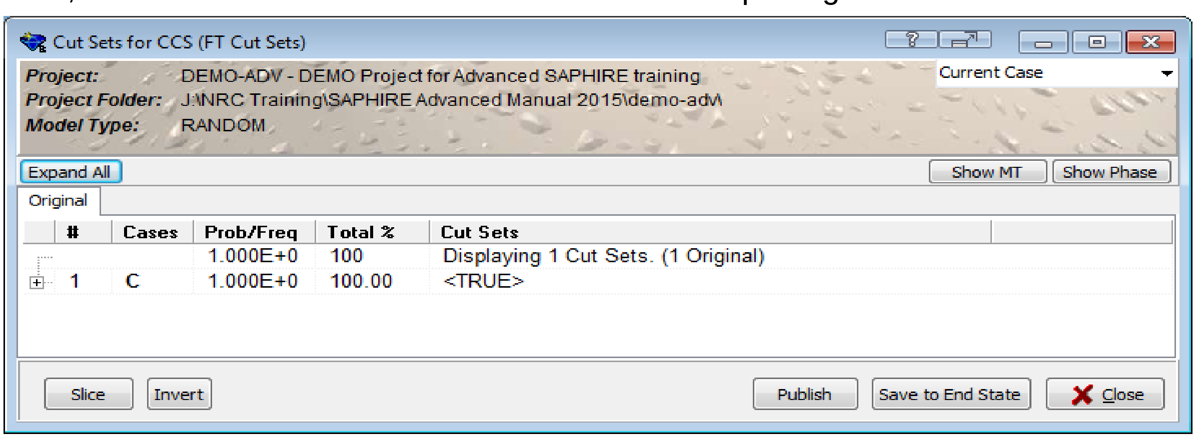


7.9. Treating a Fault Tree Gate as Failed

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). This again can be set either as a Change Set or by modifying CCS-TRAINS in edit Basic Event.

- Set the CCS-TRAINS Process Flag equal to "X" and set the Calculation Type equal to "T"

Now, solve the CCS fault tree without truncation. Reporting the solve results shows:

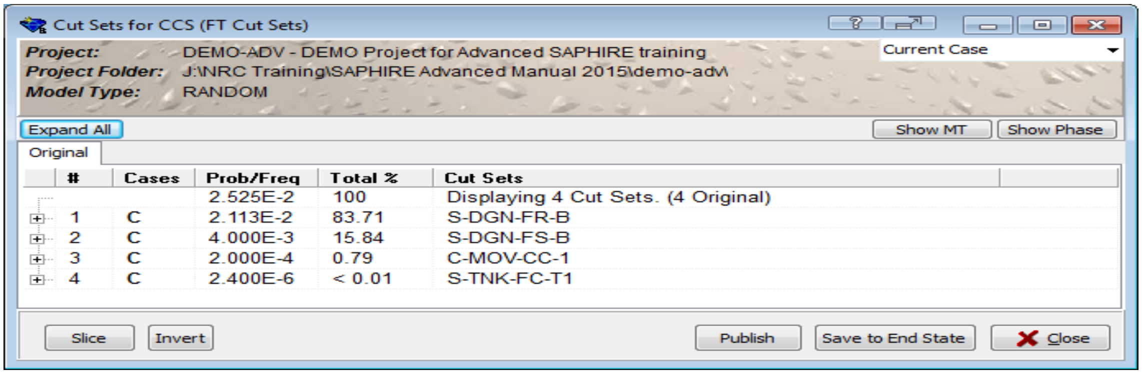


7.10. Treating a Fault Tree Gate as Working

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” and set the Calculation Type equal to “F”

Now, solve the CCS fault tree without truncation. Reporting the solve results shows:



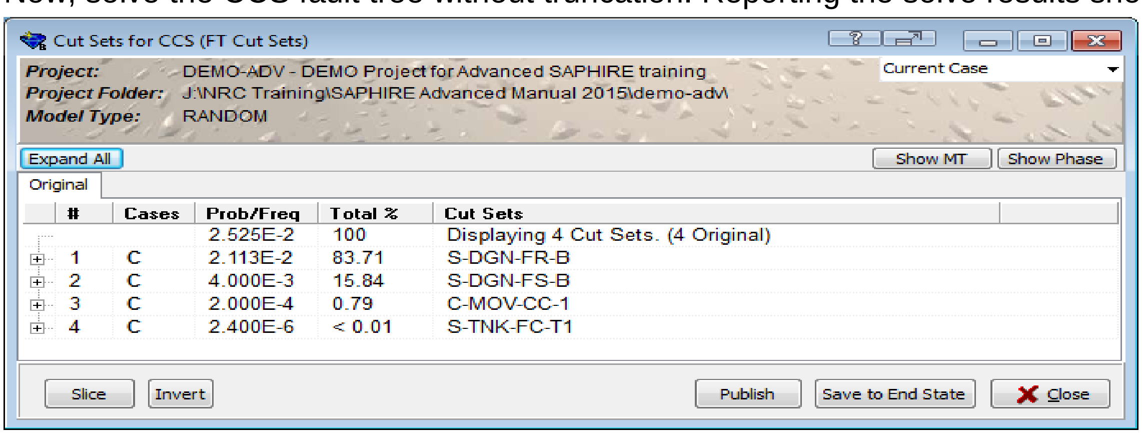
7.11. Ignoring a Fault Tree Gate

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” and set the Calculation Type equal to “I”

Now, solve the CCS fault tree without truncation. Reporting the solve results shows:

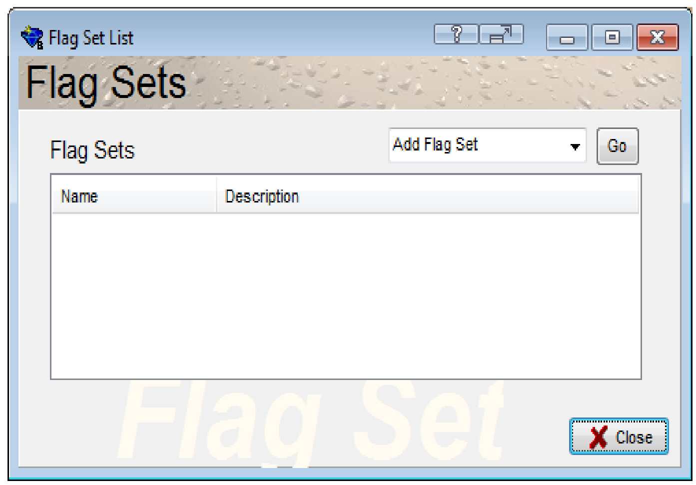


7.12. Using Flag Sets during Fault Tree Cut Set Solving

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 **Project → Flag Sets** menu. This will open the **Flag Set List** form:

From the **Flag Set List** a flag set can be added, modified or deleted.



- 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.

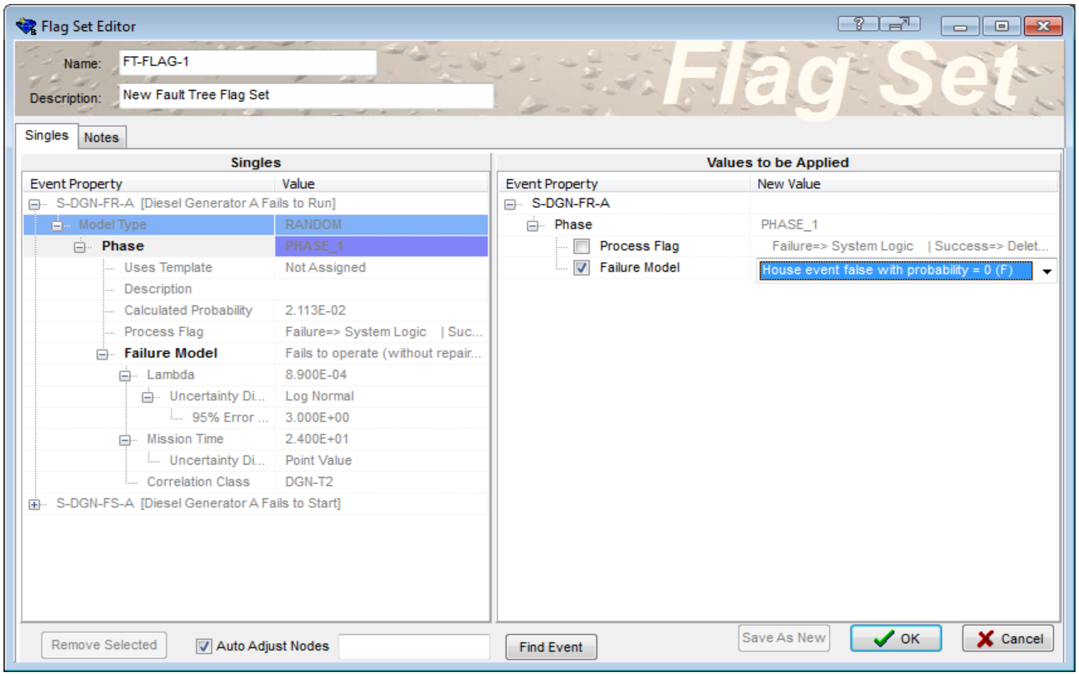
When generating fault tree cut sets, Flag Sets are used for setting house events or basic events to either TRUE, FALSE, or IGNORE.

7.12.1. To make a Flag Set

As an example we'll make a flag set FT-FLAG-1 which will change the basic event S-DGN-FR-A and S-DGN-FS-A from their nominal probability value to a FALSE value.

- Enter the Project → Flag Sets menu
- Select Add Flag Set from the dropdown menu and Click **Go**, this will open the **Flag Set Editor** window.
- Enter the Flag Set name (maximum of 24 characters) and description.
- Click and drag desired Basic Events from the basic event list panel and drop them into the **Singles** area.
- Select the event to modify by clicking on the basic event in the **Singles** window. This opens the event in the **Values to be Applied** area.

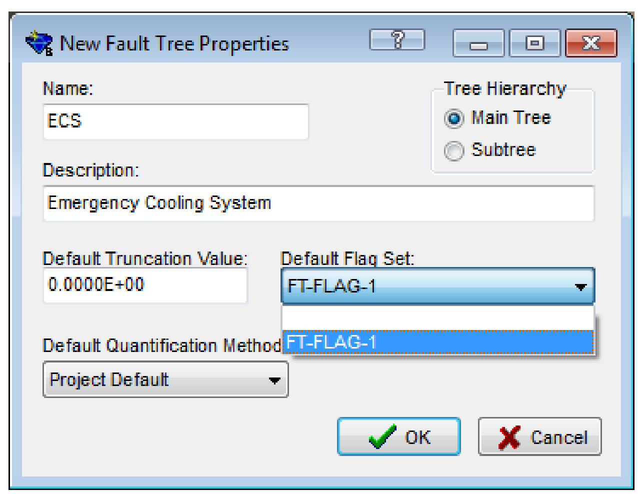
- Check the Model Type to modify by checking the box and then selecting the New Value from the dropdown list.



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

7.12.2. To use a Flag Set

- The Flag set can be assigned to a fault tree permanently using the following steps or applied temporarily during the solve option, each will be discussed.
- After the Flag Set has been created, the Flag Set name needs to be assigned to one or more fault trees.
- Open the desired fault tree (ECS in our example) by double clicking on the fault tree name in the Fault Tree list panel and select **Edit → Properties**.
- With the ECS fault tree open, select from the Fault Tree main menu **Edit → Properties** to open the **Edit Fault Tree Properties** form.
- Alternately, the flag set can be assigned by highlighting the fault tree, right-clicking to invoke the menu and selecting **Edit Properties**.



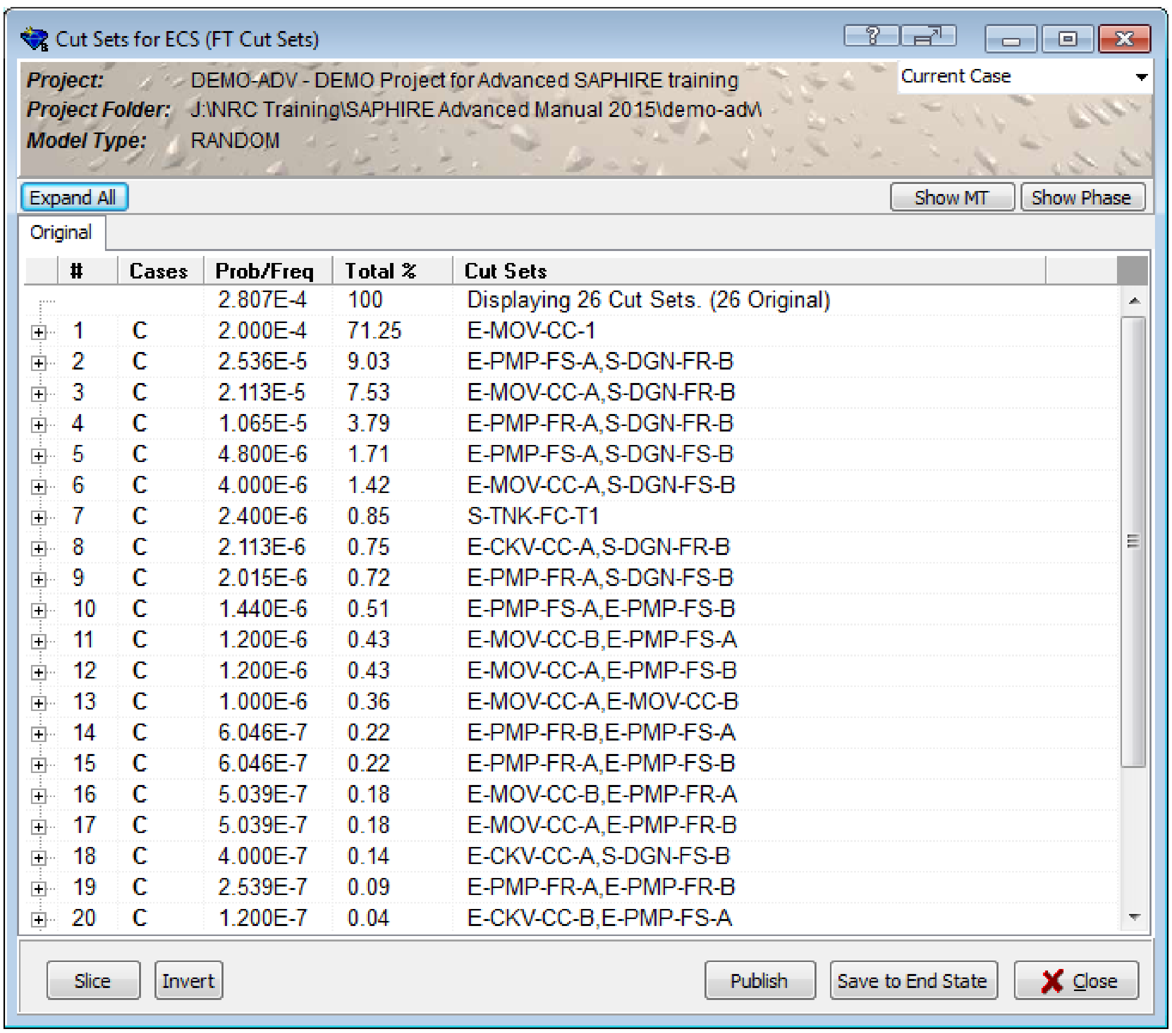
- The FT-FLAG-1 flag set is assigned to the ECS fault tree by using the pull down menu on the Edit Fault Tree Properties window as shown. Click **OK** and to save the flag set to the fault tree (permanently assigned to this fault tree and every time it is solved the flag set will be applied).

- In summary, thus far:

  - Flag Set FT-FLAG-1 was created which set the DG-A basic event failure models to FALSE.

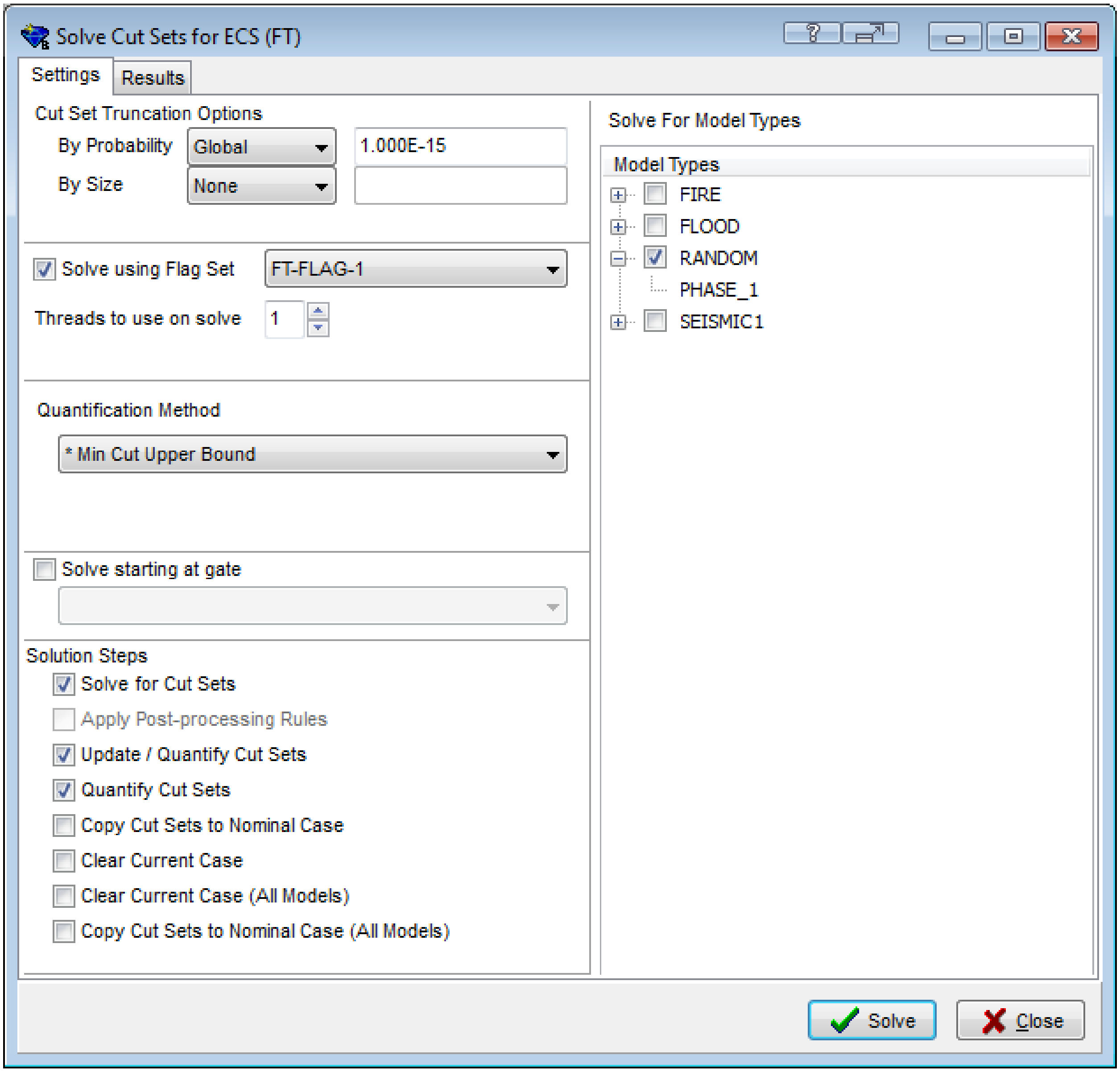
  - The Flag Set FT-FLAG-1 was assigned to fault tree ECS.

- Now, highlight the ECS fault tree, right mouse click and select the **Solve** option and solve without truncation. Reporting the solve results shows the following (note that the cut sets do not include basic events S-DGN-FR-A and S-DGN-FS-A).

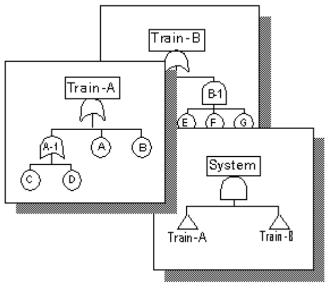


#	Cases	Prob/Freq	Total %	Cut Sets
		2.807E-4	100	Displaying 26 Cut Sets. (26 Original)
1	C	2.000E-4	71.25	E-MOV-CC-1
2	C	2.536E-5	9.03	E-PMP-FS-A,S-DGN-FR-B
3	C	2.113E-5	7.53	E-MOV-CC-A,S-DGN-FR-B
4	C	1.065E-5	3.79	E-PMP-FR-A,S-DGN-FR-B
5	C	4.800E-6	1.71	E-PMP-FS-A,S-DGN-FS-B
6	C	4.000E-6	1.42	E-MOV-CC-A,S-DGN-FS-B
7	C	2.400E-6	0.85	S-TNK-FC-T1
8	C	2.113E-6	0.75	E-CKV-CC-A,S-DGN-FR-B
9	C	2.015E-6	0.72	E-PMP-FR-A,S-DGN-FS-B
10	C	1.440E-6	0.51	E-PMP-FS-A,E-PMP-FS-B
11	C	1.200E-6	0.43	E-MOV-CC-B,E-PMP-FS-A
12	C	1.200E-6	0.43	E-MOV-CC-A,E-PMP-FS-B
13	C	1.000E-6	0.36	E-MOV-CC-A,E-MOV-CC-B
14	C	6.046E-7	0.22	E-PMP-FR-B,E-PMP-FS-A
15	C	6.046E-7	0.22	E-PMP-FR-A,E-PMP-FS-B
16	C	5.039E-7	0.18	E-MOV-CC-B,E-PMP-FR-A
17	C	5.039E-7	0.18	E-MOV-CC-A,E-PMP-FR-B
18	C	4.000E-7	0.14	E-CKV-CC-A,S-DGN-FS-B
19	C	2.539E-7	0.09	E-PMP-FR-A,E-PMP-FR-B
20	C	1.200E-7	0.04	E-CKV-CC-B,E-PMP-FS-A

- Another way to solve a fault tree using a flag set is to select the Flag Set in the Solve option. This will apply the flag set during just this cut set generation. It can be viewed as a temporary application, since the next time the fault tree is solved this flag set will not be applied.



7.13. 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.

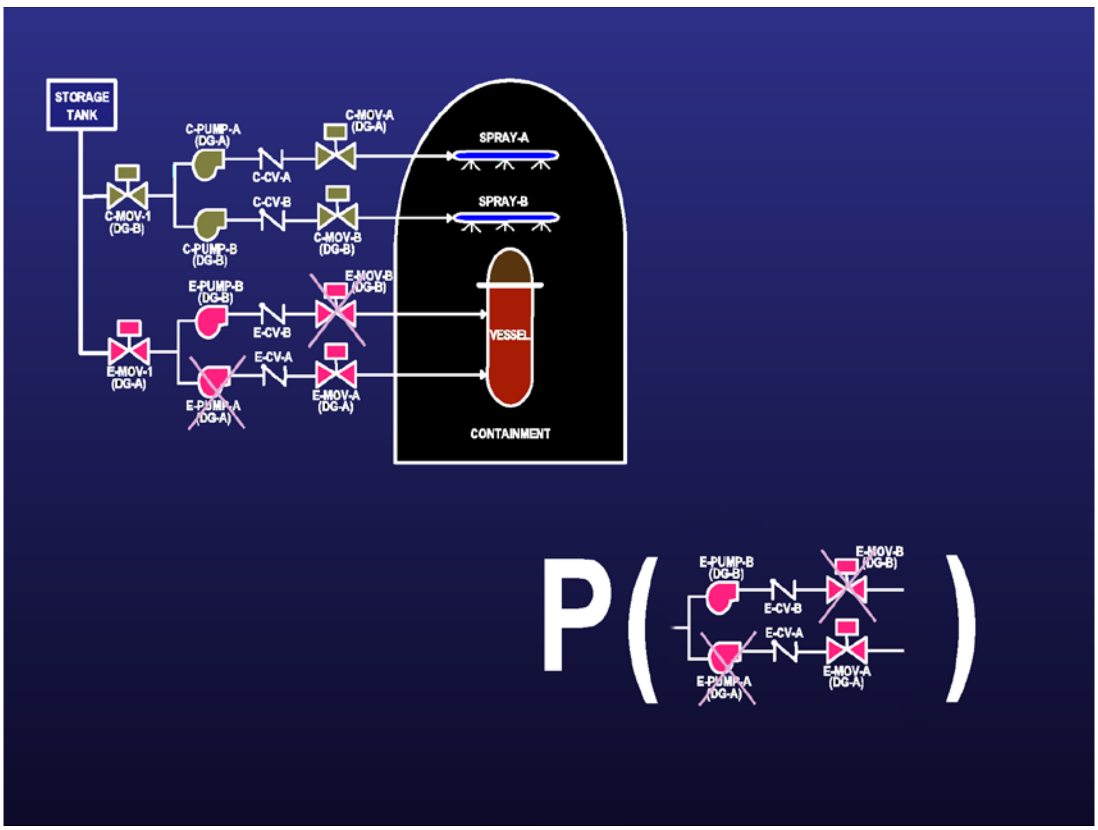
System	AND	Train-A	Train-B	
Train-A	OR	A-1	A	B
A-1	OR	C	D	
Train-B	OR	H		

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 different ways to quantify minimal cut sets. But, it is standard to use one of three methods, which include

- **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.

- **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.

- **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 [BE-A*BE-B], [BE-B*BE-C], and [BE-D]; which can be expressed as [BE-A∩BE-B] ∪ [BE-B∩BE-C] ∪ [BE-D].

For 3 passes, the exact solution is:

[BE-A*BE-B]+[BE-B*BE-C]+[BE-D] - {[BE-A*BE-B]*[BE-B*BE-C]+[BE-A*BE-B]*[BE-D]+ [BE-B*BE-C]*[BE-D]} + {[BE-A*BE-B]*[BE-B*BE-C]*[BE-D]}, which reduces down to:

[BE-A*BE-B]+[BE-B*BE-C]+[BE-D] - {[BE-A*BE-B*BE-C]+[BE-A*BE-B*BE-D]+ [BE-B*BE-C*BE-D]} + {[BE-A*BE-B*BE-C*BE-D]}

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.

8.2.1. Example Quantification Options

For our example above, let’s assume the probability of the basic events are P(BE-A) = 0.7, P(BE-B) = 0.7, P(BE-C) = 0.7 and P(BE-D) = 0.5.

- the minimal cut set upper bound approximation for fault tree X is

  Pr(X) = 1 – [(1-Pr(BE-A*BE-B))(1-Pr(BE-B*BE-C))(1-Pr(BE-D))]

  Pr(X) = 1-[(1-0.49)*(1-0.49)*(1-0.5)] = 0.870

- the rare event approximation for X is simply

  Pr(X) = Pr(BE-A*BE-B) + Pr(BE-B*BE-C) + Pr(BE-D)

  Pr(X) = 0.49 + 0.49 + 0.5 = 1.0

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.

The Min/Max calculation is shown below, where the results are displayed for each pass. (Note: the Min/Max and the minimal cut set upper bound will be equal when the cut sets do not contain common events.) However, in this case they do contain common events.

# of Passes	Min/Max Equation	Min/Max Probability
1	BE-A*BE-B + BE-B*BE-C + BE-D	1.48
2	BE-A*BE-B + BE-B*BE-C + BE-D – {[BE-A*BE-B*BE-C]+[BE-A*BE-B*BE-D]+ [BE-B*BE-C*BE-D]}	1.5 - 0.833 = 0.647
3	BE-A*BE-B + BE-B*BE-C + BE-D – {[BE-A*BE-B*BE-C]+[BE-A*BE-B*BE-D]+ [BE-B*BE-C*BE-D]} + {[BE-A*BE-B*BE-C*BE-D]}	1.48 - 0.833 + 0.1715 = 0.8185
4	Same as above	Same as above

8.2.2. Using the Min/Max Quantification Option

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

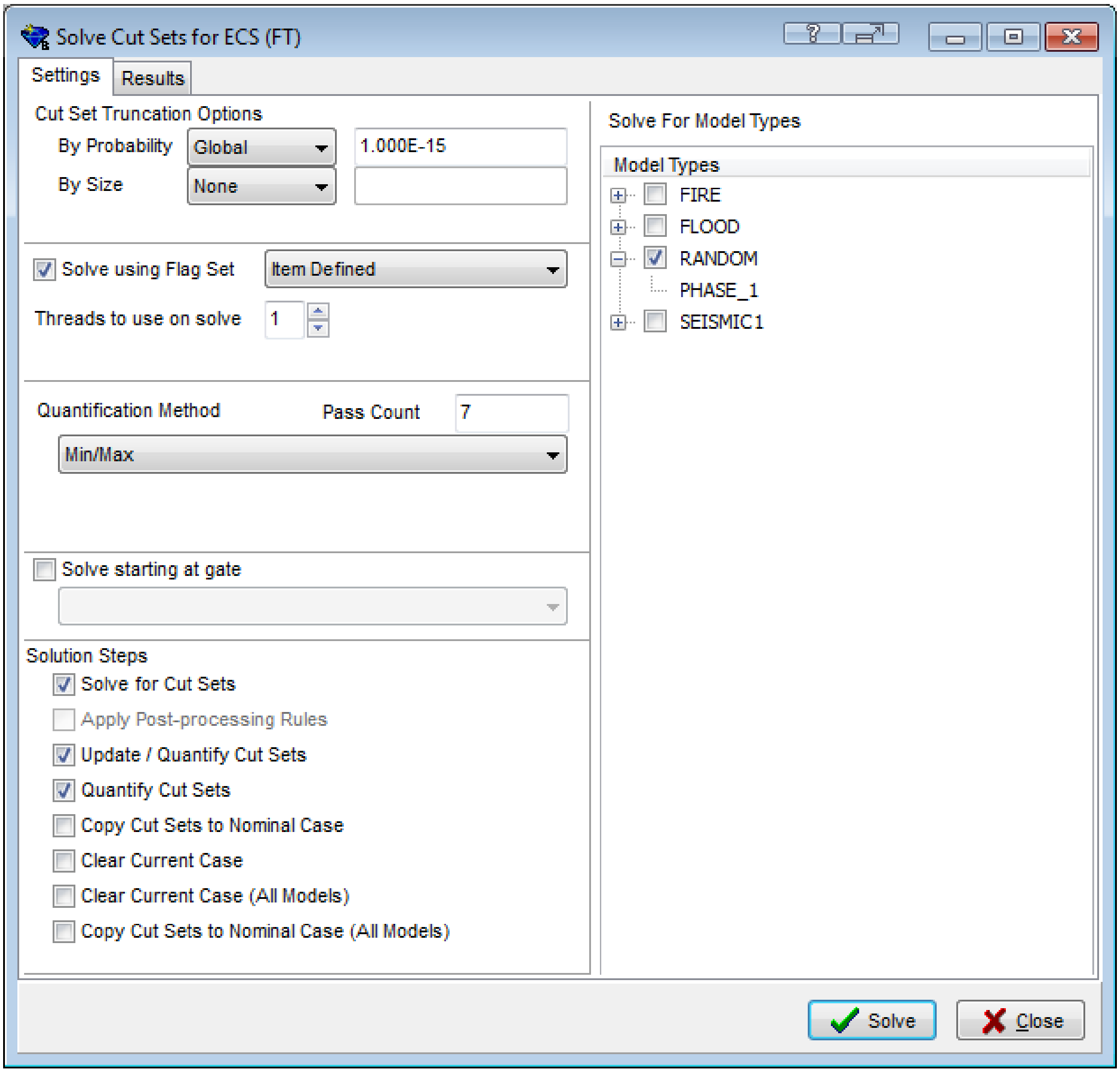
The number of passes must be selected by the user.

- 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.

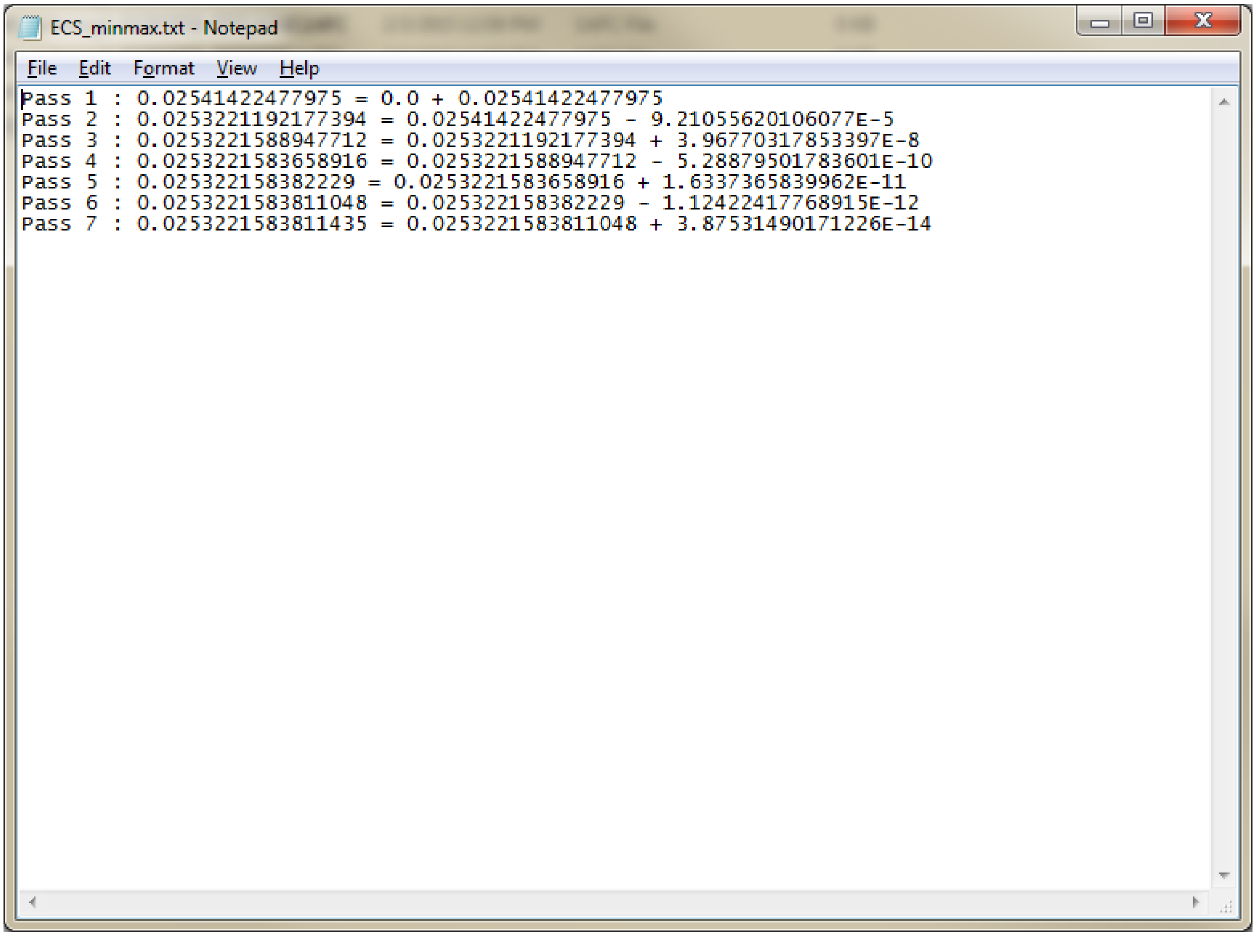
The computer run-time needs to be compatible with the user's needs. The Min/Max run-time is a function of the number of cut sets and the number of passes.

To analyze fault tree cut sets, select the **Fault Tree** menu by right clicking on the highlighted fault tree of interest and select **Solve** (Similarly, to analyze sequence cut sets, select the **Event Tree** of interest). Then choose Quantification Method of Min/Max in the drop-down menu and enter a Pass Count to use in the text field. The graphic below shows the CCS tree ready to solve for Min/Max quantification with 15 passes.



Click the Solve button and the Min/Max Quantification results will display on the screen for review (very quickly).

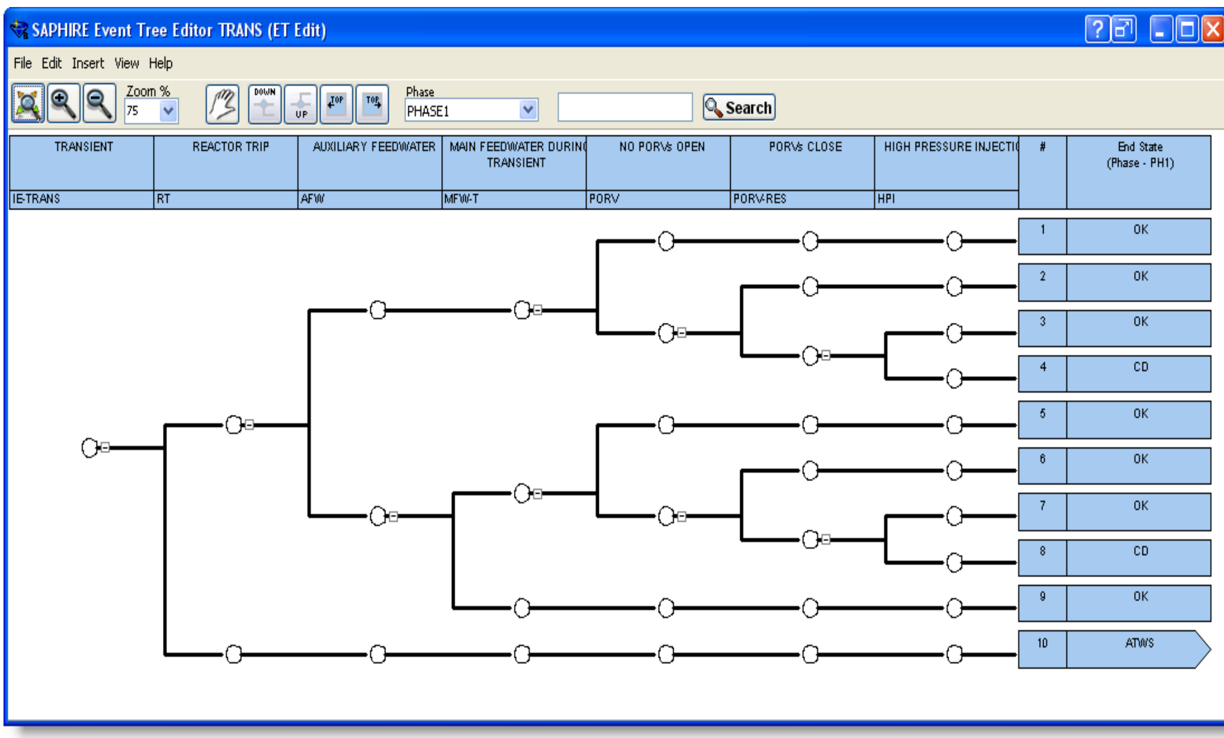
The following was captured to show the pass calculation for the ECS fault tree at 7 passes.




```
Pass 1 : 0.02541422477975 = 0.0 + 0.02541422477975
Pass 2 : 0.0253221192177394 = 0.02541422477975 - 9.21055620106077E-5
Pass 3 : 0.0253221588947712 = 0.0253221192177394 + 3.96770317853397E-8
Pass 4 : 0.0253221583658916 = 0.0253221588947712 - 5.28879501783601E-10
Pass 5 : 0.025322158382229 = 0.0253221583658916 + 1.6337365839962E-11
Pass 6 : 0.0253221583811048 = 0.025322158382229 - 1.12422417768915E-12
Pass 7 : 0.0253221583811435 = 0.0253221583811048 + 3.87531490171226E-14
```

| 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.

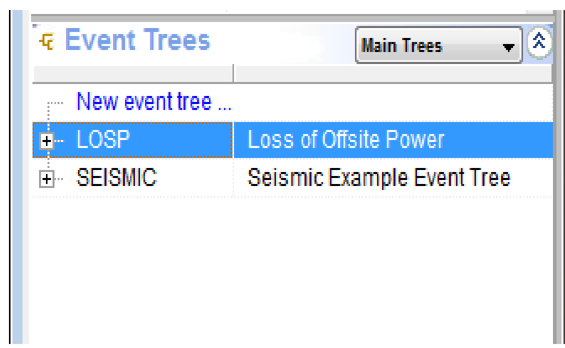
Prerequisites that are required prior to solving for sequence minimal cut sets are:

- ♦ Fault tree and event tree logic was created by using the graphics editors (or loaded into the database via the MAR-D interface).
- ♦ Basic event data were added through the Basic Event edit menu.
- ♦ Sequence logic was generated via the **Event Tree list menu** → **Link** option (SAPHIRE automatically performs this step when the event tree is saved).

Menus and options for sequence cut set solving

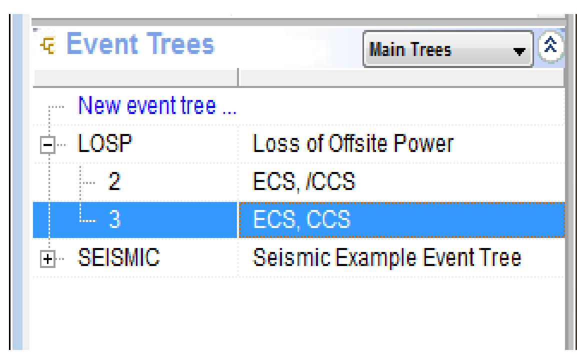
Selecting Event Trees to Solve

- ♦ Highlight the Event Tree(s) of interest in the Event Tree list panel to solve.
- ♦ Right-click to invoke the pop-up menu.
- ♦ Select the **Solve** option.



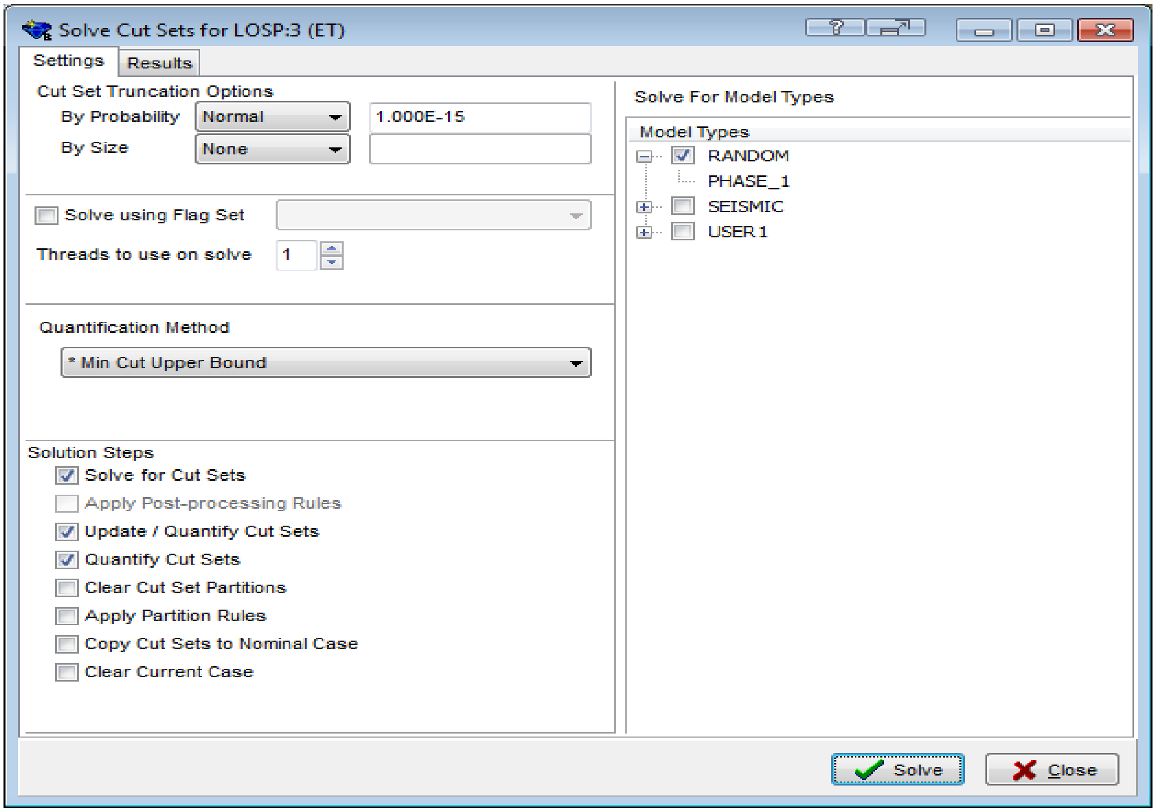
Selecting Individual Sequences to Solve

- ♦ Expand the Event Tree(s) of interest in the Event Tree list panel and highlight the sequence(s) to solve.
- ♦ Right-click to invoke the pop-up menu.
- ♦ Select the **Solve** option.



Solve Sequence Cut Sets Dialogue

The Solve Cut Sets window for sequences is similar to the one for fault trees.



Cut Set Truncation Options – Use the pull down menus to select the options.

By Probability:

None – solves without probability value truncation.

Normal – uses the cutoff value in the text field divided by the initiating event frequency.

Conditional – uses the cutoff value in the text 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.

By Size:

None – solves without size truncation

Size – cut sets having more events than specified in the text field will not be retained.

Zone – cut sets having more Zone Flagged Events than specified in the text field will not be retained. This option is generally not used.

Solve Using Flag Set – Check this box to enable the drop down menu that lists available Flag Sets to use when solving the sequences selected.

Threads to use on solve – This value can be set from 1 through 32. Leaving this as a value of 1 will solve at the same speed as previous versions of SAPHIRE. To take advantage of multiple processor computers, increasing the number of threads can save significant solving time for lower level truncations or large models.

Quantification Method – This drop down menu that lists available quantification methods available to calculate the resultant sequence cut sets.

Solution Steps – This section of the form provides check boxes to choose the steps that are taken to solve the sequence cut sets.

Solve for Cut Sets – If you check box, sequence logic will be solved for minimal cut sets using the truncation options specified. A situation where one might want to uncheck this box would be to re-quantify the results of the cut sets after changing some basic event values, but the structure of the fault trees and event trees stayed the same. This would save some time in solving large models.

Apply Post Processing Rules – If you check box, any post-processing rules associated with the sequence(s) will automatically be applied after the sequence(s) cut sets have been generated. Generally, this box will be checked in parallel with Solve for Cut Sets.

Update / Quantify Cut Sets – If this box is checked, the cut sets will be updated by removing any non-minimal cut sets that may have been introduced when post-processing rules were applied and then quantified. Again, if one might want to uncheck this box to just solve the cut sets during a model review and save some time by omitting the quantification.

Quantify Cut Sets – If this box is checked, cut sets will be re-quantified using the truncation options specified.

Clear Cut Set Partitions – If this box is checked, the end states created with the Apply Partition option will be delinked (i.e., no longer associated to cut sets and they can be deleted).

Apply Partition Rules – If this box is checked, any partition rules associated with this sequence will be applied to group sequence cut sets into end states based on user specified partition rules.

Copy Cut Sets to Nominal case – If this box is checked, the solved sequence cut sets will be stored into the Nominal case (permanent record of this information).

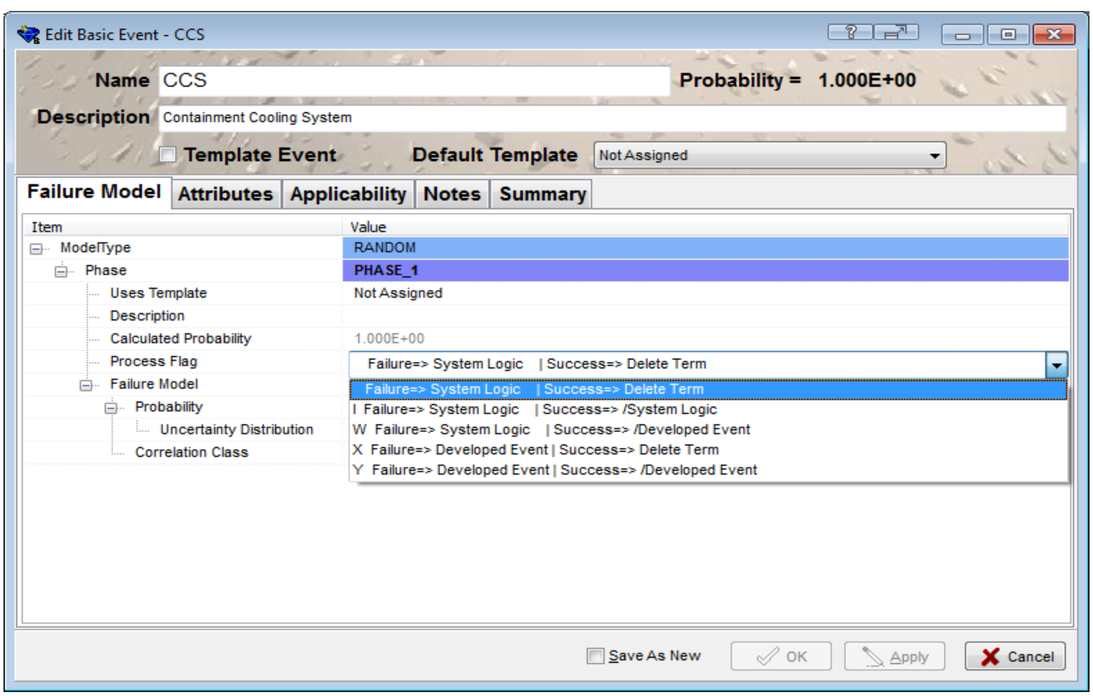
Clear Current Case – If this box is checked, all sequence(s) cut sets is cleared from the current case.

Solve for Model Types – Select as many model types as desired to include in the cut sets. Typically, only one would be selected at a time.

9.2. Process Flags and Sequence Cut Set Generation

Process Flags are special identifiers that tell SAPHIRE how to the treat top events in an event tree when the sequences are solved. For example, SAPHIRE has a Process Flag that identifies the top events as Basic Events with a split-fraction probability rather than its Fault Tree logic.

- ♦ The process flag is entered in the basic event editing. From the **Basic Event** list drop down menu, select **All** (or **Developed Event**) to be able to view the top events of the fault trees. Highlight the fault tree top event to be modified, click the right mouse button, and select **Edit Basic Event.**



- 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 (they are Developed Events).

- 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.

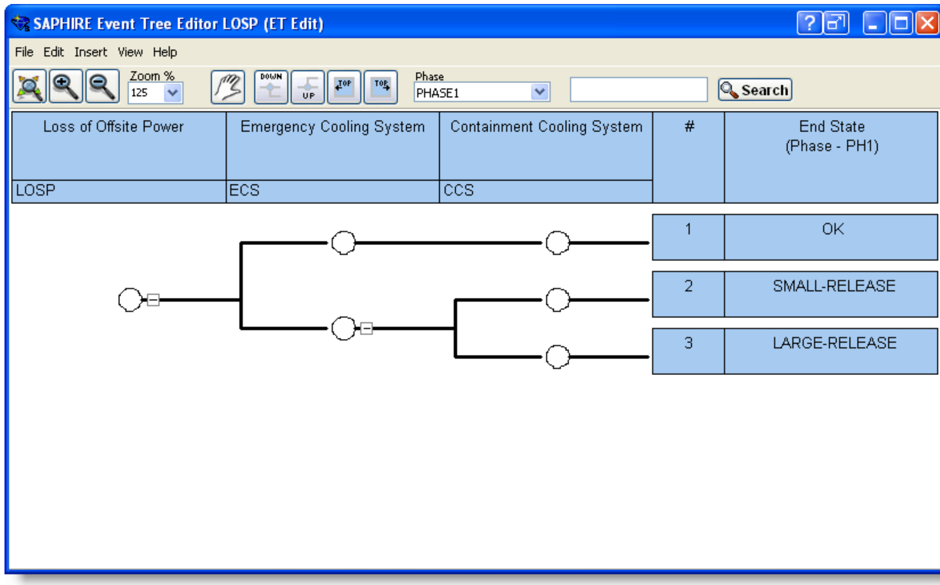
> **Example:**
> If an event tree top event is treated as a basic event (via the **Y** process flag) but is not independent of other top events, it is possible to obtain non-conservative results due to double counting of basic events.

- ♦ The " " (space) process flag gets the most use since this is the default flag.
- ♦ 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.

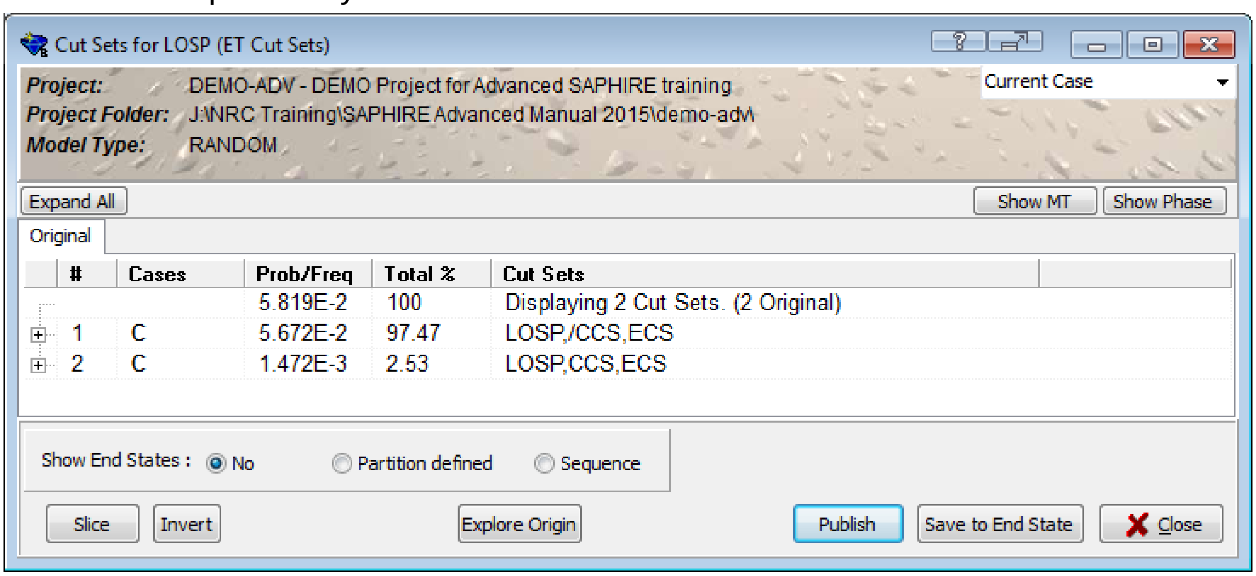
9.3. Process Flag Example

Once the process flags have been defined for the top events, sequence cut sets are then solved by highlighting the event tree(s) of interest in the **Event Tree** panel, right clicking and selecting **Solve**.

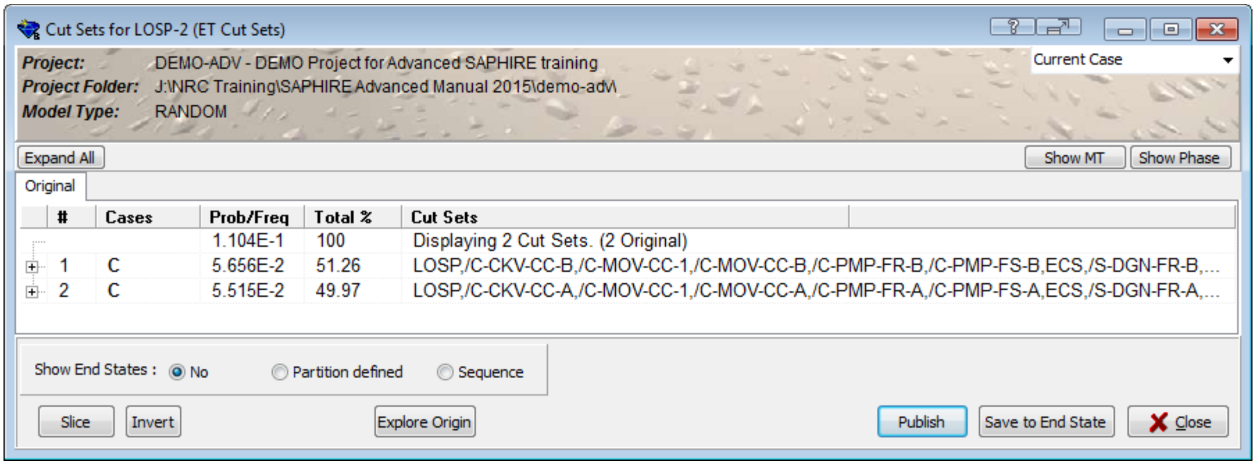
- ♦ The LOSP event tree will be used to demonstrate how process flags operate.
  - ◊ Modify the process flag via a change set (section 2.4) for both CCS and ECS to **Y**. Also, set the CCS and ECS developed events to a probability of 2.53E-2 (representing the individual system failure probabilities).



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



- 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 2.53E-2.
- Generate the sequence cut sets. Sequence 2 cut sets are shown below.



Notes:

- ♦ The original frequency (i.e., calculated without process flags) for sequence 2 was found to be 5.81E-2 (truncation at 1.0E-8)
- ♦ 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.

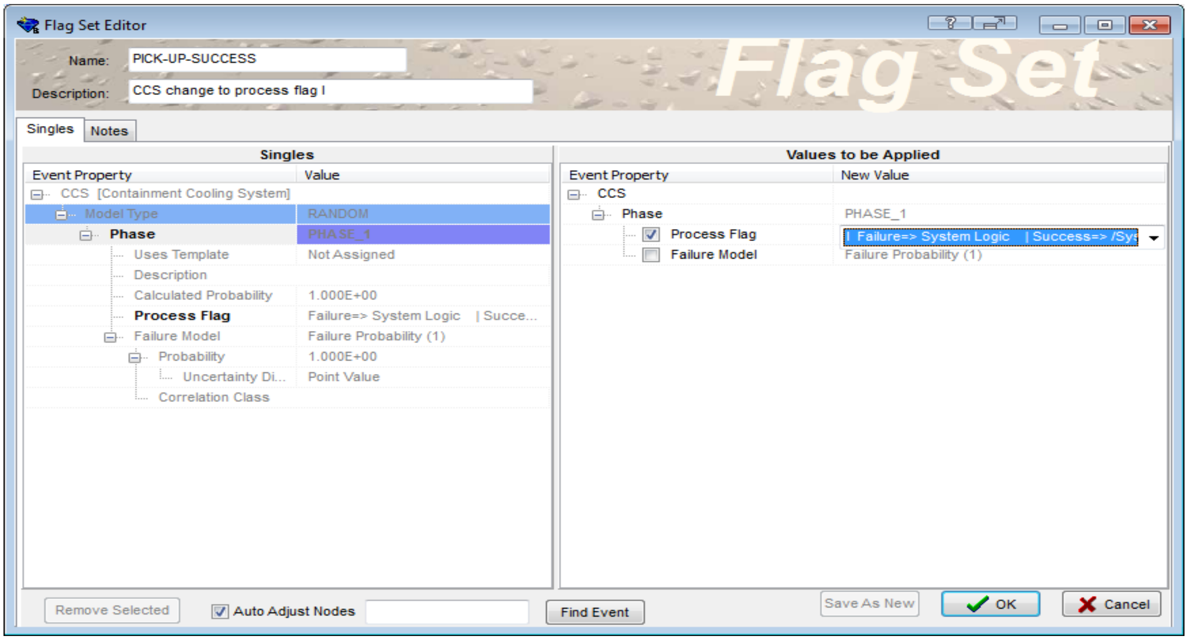
- Flag Sets are a special type of change set. SAPHIRE will keep flag sets separate from change sets by specifying it as a flag set. Flag sets are created under **Project → Flag Sets** menu which opens the **Flag Set List**, then select **Add Flag Set** from the drop down menu and click **Go**. The **Flag Set Editor** looks similar to the change set editor.
- 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**).

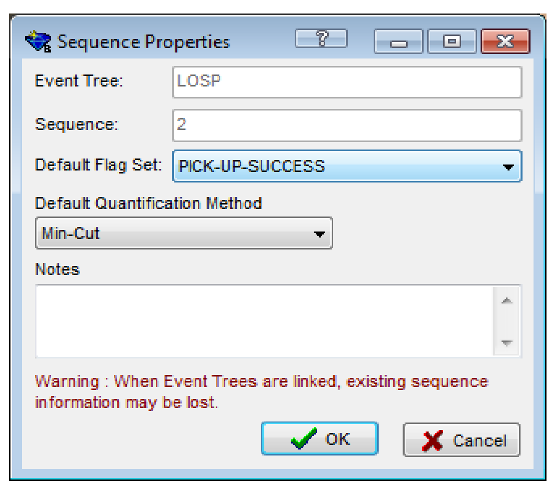
- **To make a Flag Set:**
    - ◊ Open the Flag Set Editor and name the Flag Set using 24 or less characters in length.
    - ◊ Click the event to be modified, drag and drop it into the **Singles** area of the **Flag Set Editor**.



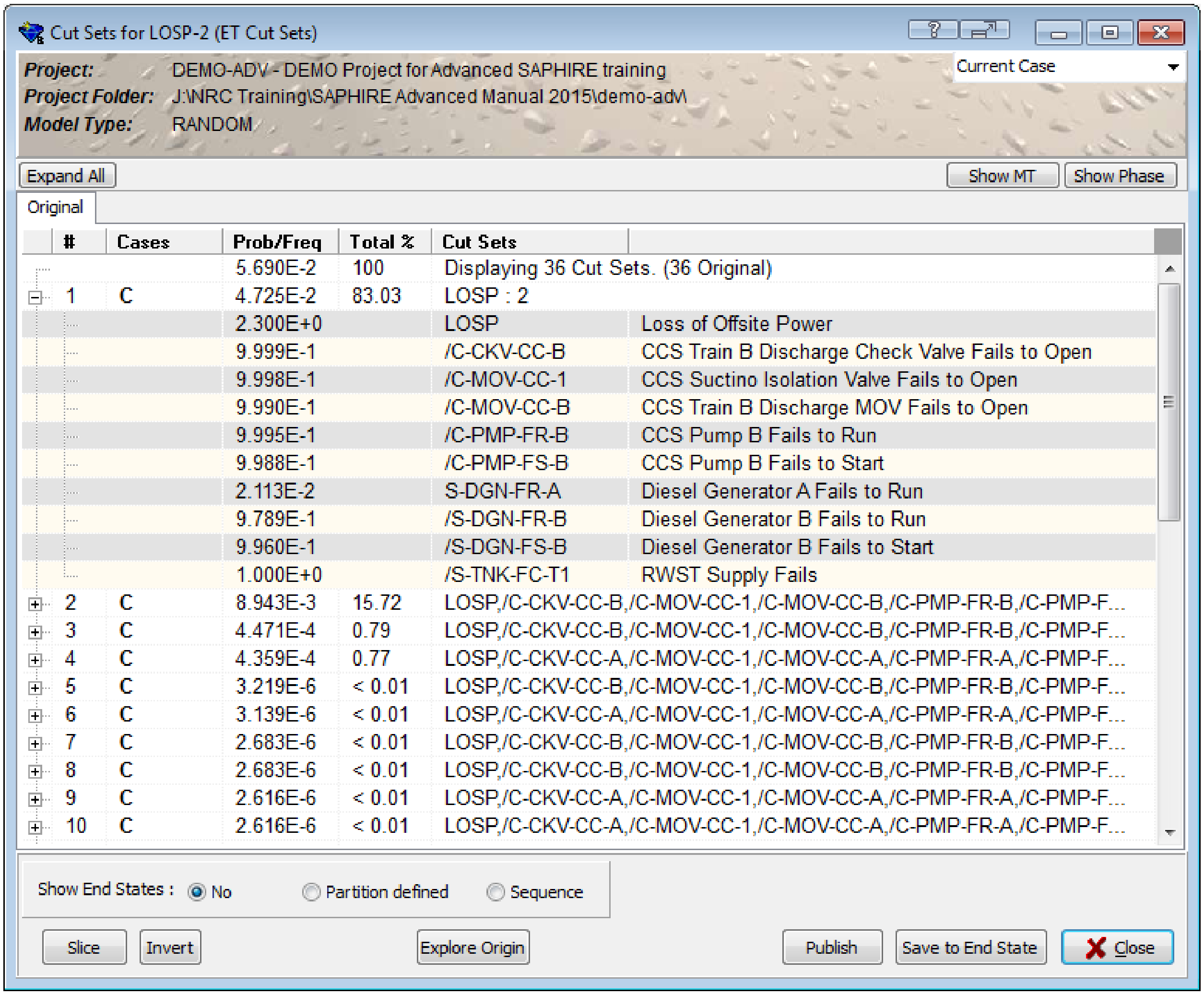
- Modify either the calculation type or the process flag.
- Continue to add events and modify in the same manner until finished.
- To use a Flag Set after it has been created the Flag Set name must be assigned to a sequence or sequences.
- To assign the Flag Set to a sequence
  - Highlight the accident sequence after the event tree has been expanded, right click and select **Edit Properties**.
  - Click the drop down box next to the “Default Flag Set:” option and select the Flag Set from the list of created flag sets.

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

- Create a Flag Set named "PICK-UP-SUCCESS"
  - In this Flag Set, set the process flag for top event CCS was changed to a value of **I**.
- Assign the Flag Set "PICK-UP-SUCCESS" to LOSP sequence 2.



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



♦ Assigning Flag Sets to individual sequences this way is not recommended, since the event tree logic could change or just opening the event tree and saving it will cause the event tree to generate new sequence logic and the Flag Set would no longer be utilized by that specific sequence.

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. Also, existing Flag Sets can be assigned using event tree linking rules.

- 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.

9.5.1. "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-CC-A and E-PUMP-FR-A to TRUE only if top event ECS
| fails in the LOSP event tree sequence.

if  ECS  then
  Eventree(LOSP) = True(E-MOV-CC-A, E-PMP-FR-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.
```

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, post processing, 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 post processing 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 post processing 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.

<table>
<tr><th>Keyword or symbol</th><th>Definition</th><th>Usage</th></tr>
<tr><td>/</td><td>Symbol used to represent a complemented event (i.e., the success of a system).</td><td>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").</td></tr>
<tr><td>|</td><td>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.</td><td>| Place your comments here!

| Note that blank lines are also permissible!</td></tr>
<tr><td>;</td><td>Symbol to indicate the end of a macro line or a line that modifies the cut set being evaluated.</td><td>| usage for a macro command
MACRO-NAME = "search criteria" ;</td></tr>
<tr><td>*</td><td>Symbol to indicate the logical AND command.</td><td>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.</td></tr>
<tr><td>+</td><td>Symbol to indicate the logical OR command.</td><td>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.</td></tr>
<tr><td>( )</td><td>Symbols to indicate a specific grouping of items.</td><td>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].</td></tr>
<tr><td>True( )</td><td>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.</td><td>if “search criteria” then
eventree(ET-NAME) = True (EVENT1, EVENT2, EVENT3, …);
endif</td></tr>
</table>

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

9.6.1. To make a Dynamic Flag Set

- Enter the **Event Tree** list panel. Highlight the event tree, right click the mouse, and select **Edit Linkage Rules**.
- Using the rule structures discussed above, construct a rule that will modify a basic event.
- Generate event tree sequences by: saving the event tree – SAPHIRE automatically links the event tree or 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.

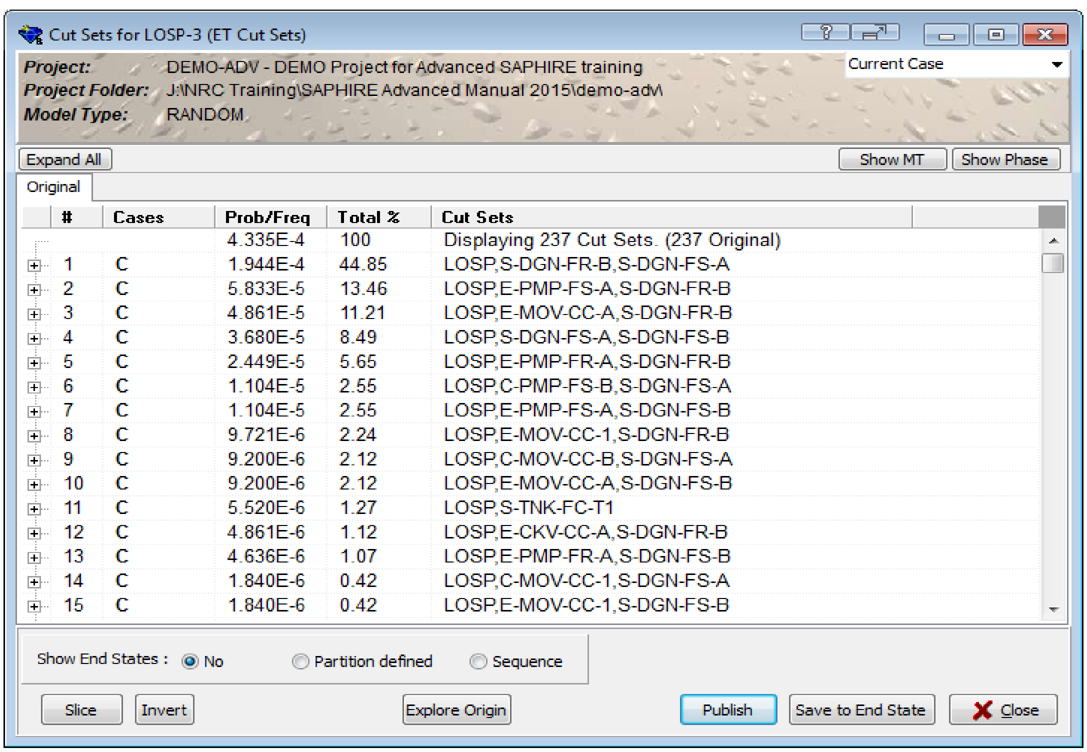
- ♦ The event tree sequences are now ready to be analyzed in the Sequence menu option.

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

- ♦ A rule was entered to set S-DGN-FR-A and C-PMP-FR-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**(S-DGN-FR-A, C-PMP-FR-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 S-DGN-FR-A and C-PMP-FR-B *do not* show up in the cut sets.

9.7. Hierarchical Flag Set Applications

Hierarchical flag sets are created when event tree sequences are generated when the event trees transfer from one event tree to another and both event trees contain flag sets. This operation is required when external events are added to models and flag sets are created for sub-event trees, which are necessary for the accident sequences to be analyzed correctly.

Example of this type of flag set:

- A new event tree is added to the project (assume Fire event tree), which requires certain basic event settings. This event tree then transfers to an existing event tree (assume LOOP), which has a pre-existing flag set

- The new event tree requires a flag set to set basic events; HE-FIRE and OEP-XHE-XL-NREC to TRUE, so a flag set is created and applied to this new event tree. Create the linkage rule for the fire event tree

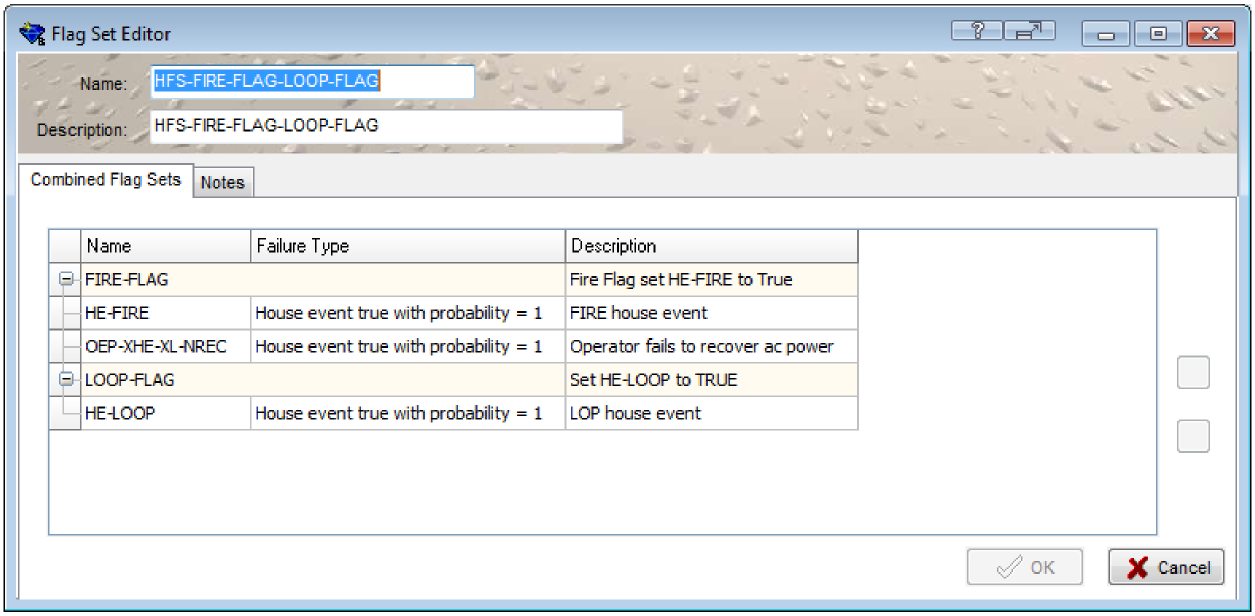
```
if always then
  eventree(FIRE) = Flag(FIRE-FLAG);
endif
```

- The LOOP event tree already contains a flag set to analyze that event tree. Instead, of creating a super flag set manually, SAPHIRE will now create the hierarchical flag set automatically and append the two flag sets together.

```
if always then
  eventree(LOSP) = Flag(LOSP-FLAG);
endif
```

- Now when the FIRE event tree is linked the two different flag sets will be combined to create the hierarchical flag set with a unique name based on the names of the two individual flag sets (HFS-FIRE-FLAG-LOOP-FLAG).

  This flag set looks like:



9.8. Other Rule Based Flag Set Applications

Previously created Flag Sets can be assigned to specific sequence via use Link Rules. This performs the same option as "Dynamic" Flag Sets except the user has provided a specific name and specific combinations of basic events.

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 is the predominate way Flag Sets are used.)
|
| 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
```

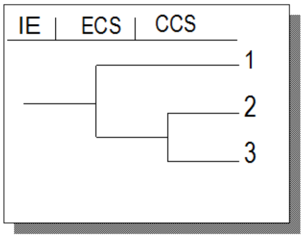
9.9. Steps Used by SAPHIRE to Solve Sequences

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")



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

FAILED LOGIC

- SAPHIRE creates a fault tree which represents the failed system(s).

"Failed" tree is shown below.

Seq-2-failed	AND	ECS
ECS	TRAN	

FAILURE CUT SETS

- Generate the failure cut sets for the failed tree.
  - See "System Cut Set Generation" section for details.

Cut sets for "Failure" fault tree

1	DG-A
2	E-MOV-1
3	TANK
4	DG-B, E-CV-A
5	DG-B, E-MOV-A
6	DG-B, E-PUMP-A
7	E-CV-A, E-CV-B
8	E-CV-A, E-MOV-B
9	E-CV-A, E-PUMP-B
10	E-CV-B, E-MOV-A
11	E-CV-B, E-PUMP-A
12	E-MOV-A, E-MOV-B
13	E-MOV-A, E-PUMP-B
14	E-MOV-B, E-PUMP-A
15	E-PUMP-A, E-PUMP-B

SUCCESS LOGIC

- SAPHIRE sets any basic event not appearing in the failed cut sets to a FALSE event.
- SAPHIRE creates a fault tree which represents the success system(s).

"Success" tree is shown below for sequence 2.

Seq-2-success	OR	CCS
CCS	TRAN	

SUCCESS CUT SETS

- Generate the success cut sets for the success fault tree.
  - See "System Cut Set Generation" section for details.

Cut sets for "Success" tree (with some events FALSE)

1	DG-B
2	TANK

CUT SET MATCHING

- SAPHIRE performs "cut set matching" on the failure cut sets.
  - If success cut set matches a failed cut set (or is subset of), the failed cut set is deleted.

Cut sets to be deleted are highlighted.

1	DG-A
2	E-MOV-1
3	***TANK***
4	***DG-B, E-CV-A***
5	***DG-B, E-MOV-A***
6	***DG-B, E-PUMP-A***
7	E-CV-A, E-CV-B
8	E-CV-A, E-MOV-B
9	E-CV-A, E-PUMP-B
10	E-CV-B, E-MOV-A
11	E-CV-B, E-PUMP-A
12	E-MOV-A, E-MOV-B
13	E-MOV-A, E-PUMP-B
14	E-MOV-B, E-PUMP-A
15	E-PUMP-A, E-PUMP-B

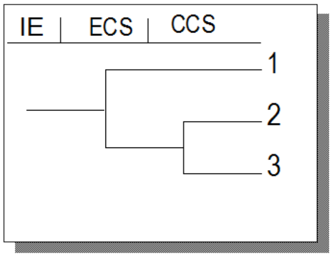
SEQUENCE CUT SETS

- SAPHIRE saves the remaining cut sets as the sequence results.

Sequence 2 cut set results are shown below.

1	DG-A
2	E-MOV-1
3	E-CV-A, E-CV-B
4	E-CV-A, E-MOV-B
5	E-CV-A, E-PUMP-B
6	E-CV-B, E-MOV-A
7	E-CV-B, E-PUMP-A
8	E-MOV-A, E-MOV-B
9	E-MOV-A, E-PUMP-B
10	E-MOV-B, E-PUMP-A
11	E-PUMP-A, E-PUMP-B

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).

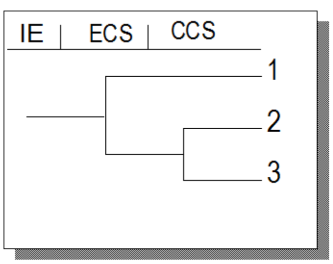
Sequence 2 fault tree is shown below.

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.

9.10. 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:

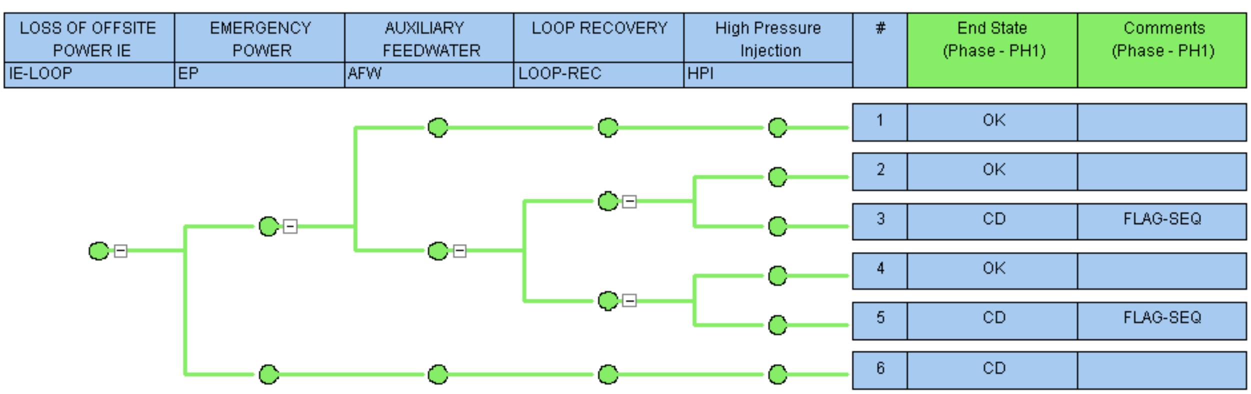
1. Use multiple fault trees
2. Use fault tree and sequence flag sets

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**:
>
> Both fault tree and sequence flag sets can be used simultaneously when analyzing event tree accident sequences.

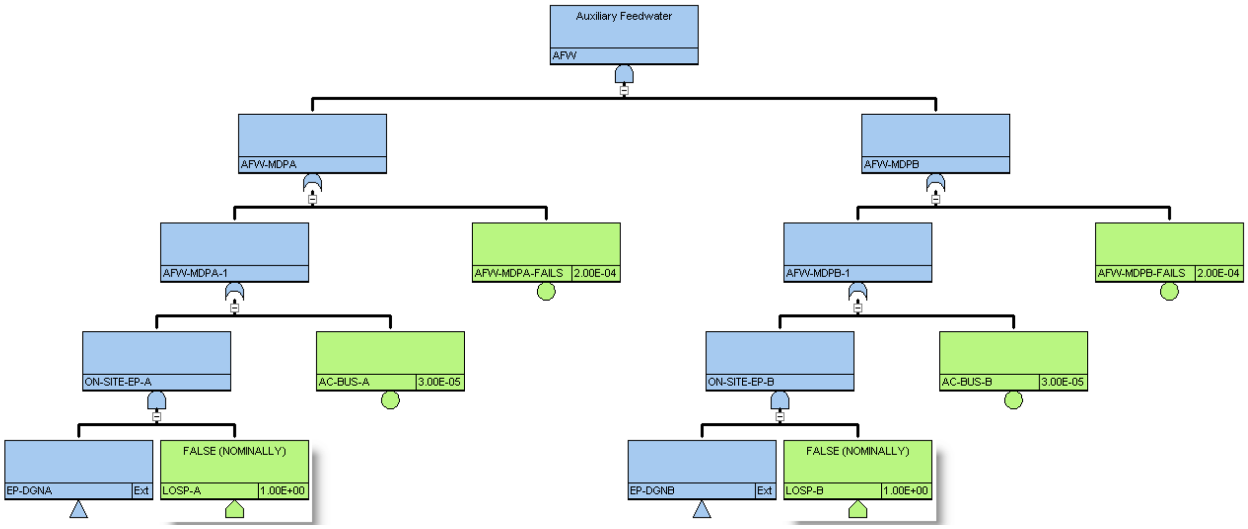
If a given sequence contains a flag set and a fault tree in that sequence has a flag set, the *fault tree flag set* takes precedence over the sequence flag set.

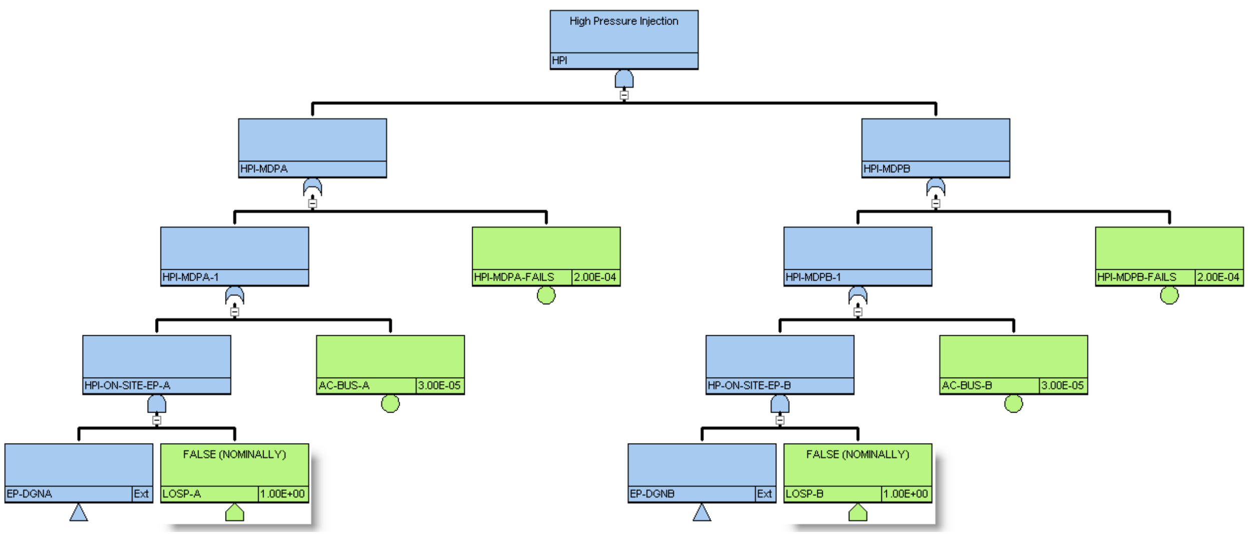
The following event tree will be used to illustrate this example.



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.

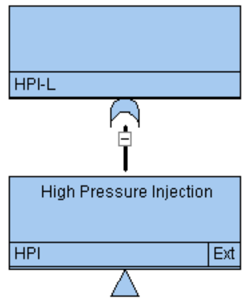
    Sequence 3 requires the house events LOSP-A and LOSP-B be set to TRUE in order for the EDGs to supply ac power to the AFW pumps. But, this sequence also requires the house events LOSP-A and LOSP-B be set to FALSE since the HPI pumps are no longer dependent upon the EDGs. So, how do we model something 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. or;
  2. We could use both sequence and fault tree flag sets when solving accident sequences. To use this method, a flag set needs 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:



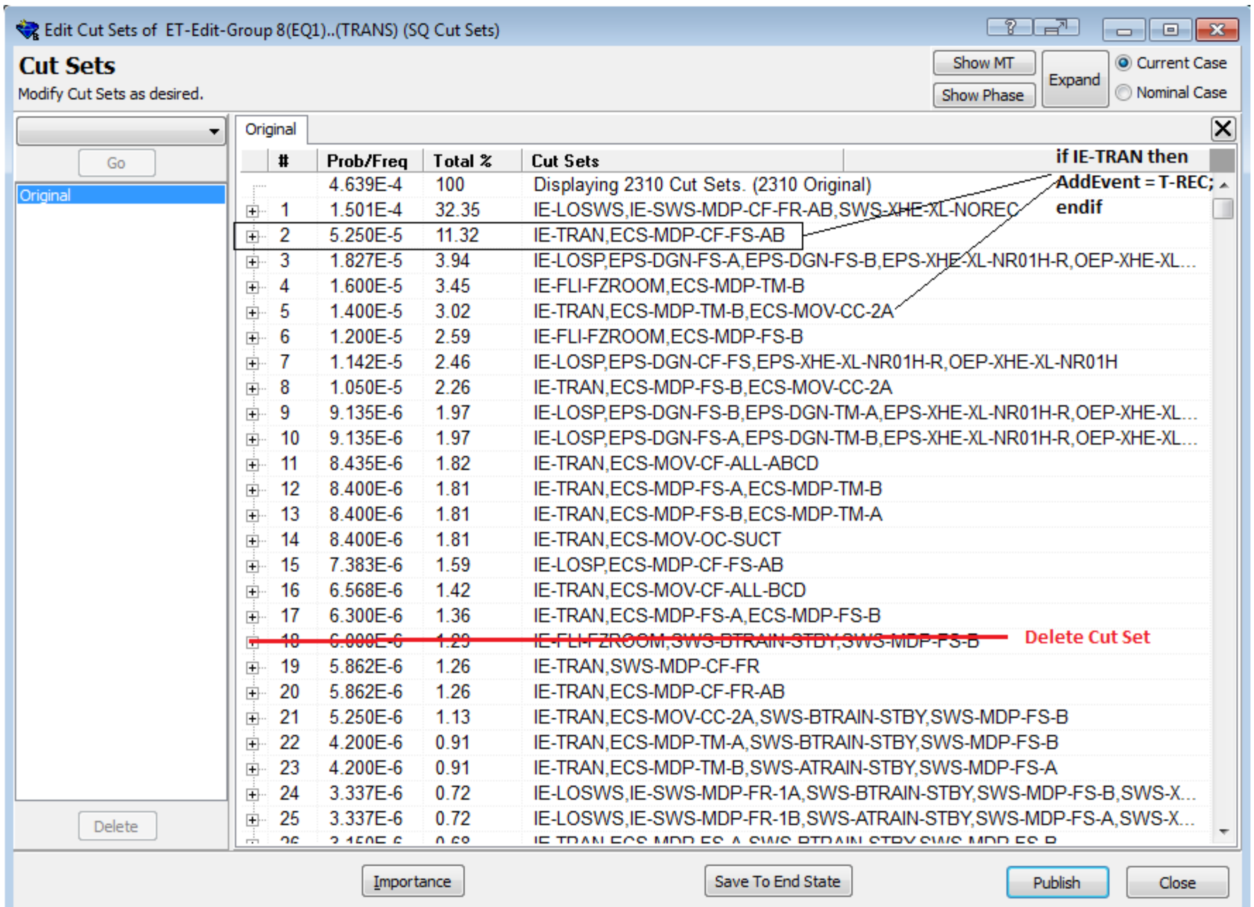
No flag set is assigned to HPI-L. HPI is assigned a flag set where LOSP-A and LOSP-B are FALSE.

- 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 | EDITING CUT SETS

Section 10 describes the edit cut set option that allows you to manipulate cut sets based on user-defined options. The edited cut sets may then be viewed or reported. The edit cut set feature is available for fault tree, sequence, or end state cut sets.

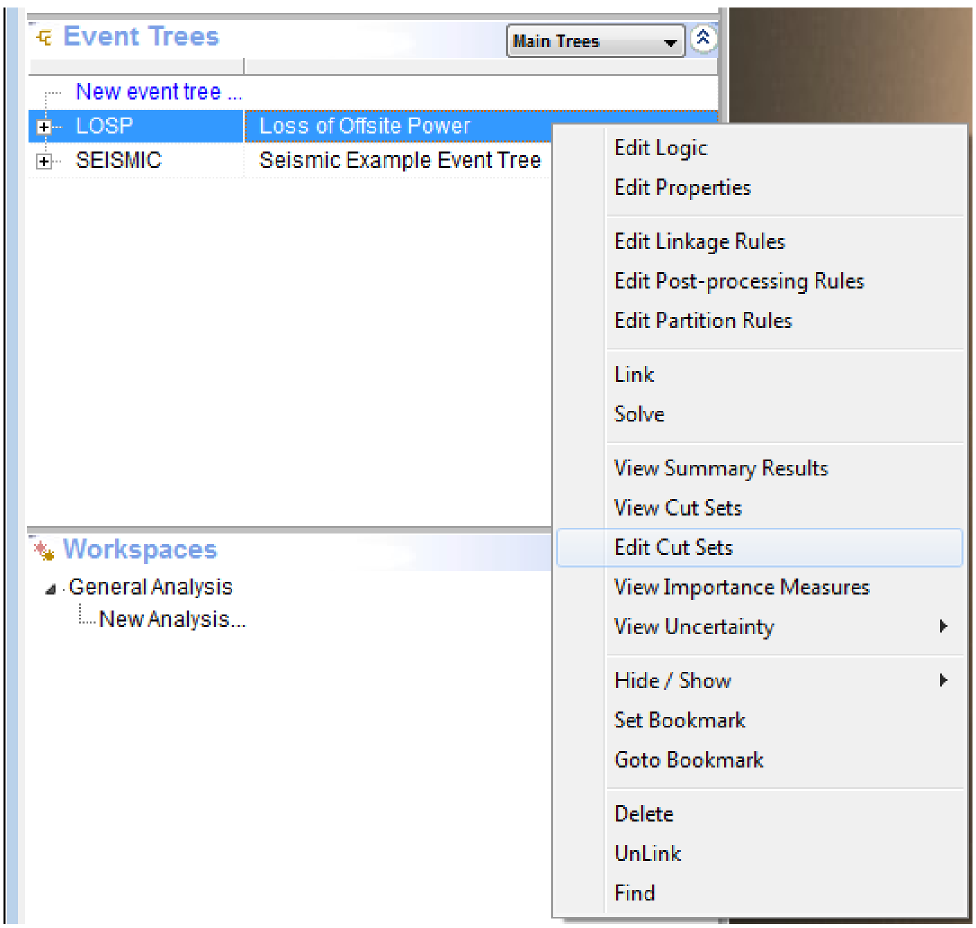


10.1. The Edit Cut Sets Option

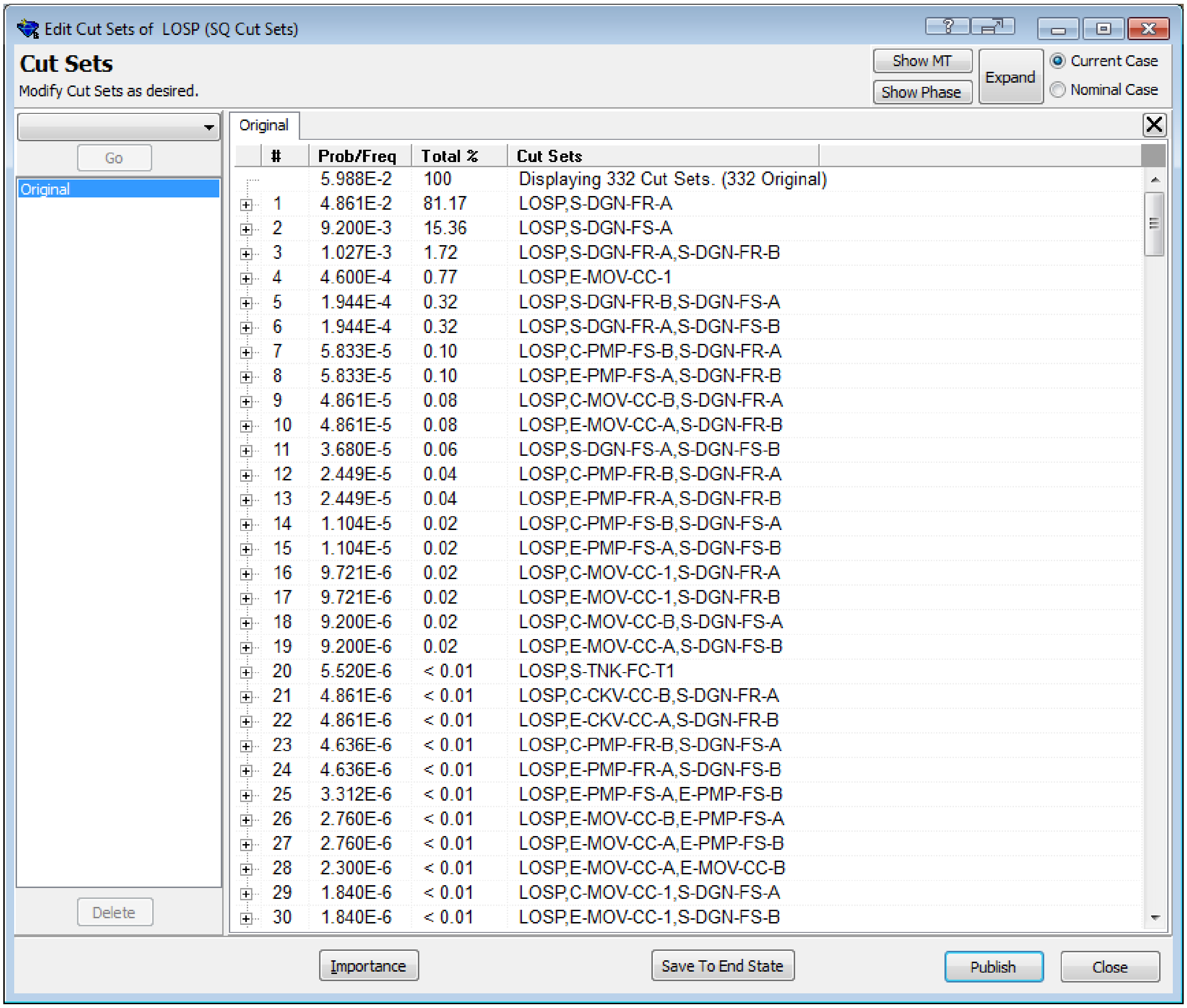
The edit cut sets option is available in the Fault Tree, Event Tree, and End State list menus.

Highlight the desired fault tree, event tree(s) [and/or sequences], or end states, right click and select the **Edit Cut Sets** option.

As an example, the Edit Cut Set option will use the LOSP event tree cut sets. These cut sets have been solved at a truncation of 1.0E-15.



From the LOSP event tree cut sets, the minimal cut set upper bound approximation frequency is 5.988E-2 and there are a total of 332 cut sets.

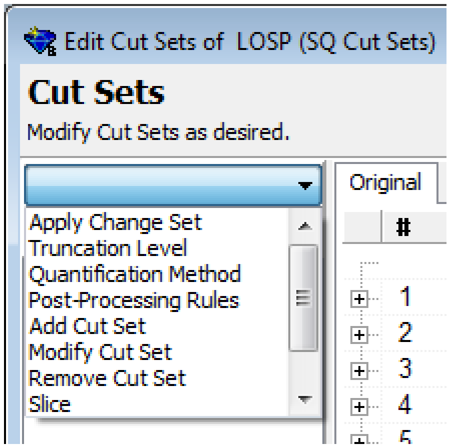


The Edit Cut Set option screen lists all of the cut sets, which can be manipulated using the many options provided. This screen allows the analyst to:

- Display the current case cut sets
- Display the nominal case cut sets (if they have been saved)
- Show the model type for the basic event (Show MT), show the phase for the basic event (Show Phase), and expand all of the cut sets to show each of the basic events and their information.
- The “Importance” button will provide the importance information for each basic event that is in the current display.
- The current displayed cut sets can be saved to an user defined end state.
- The displayed cut sets can be published into one of the formats (html, rtf, excel spreadsheet, or pdf).

10.1.1. The Modify Cut Set Options

The drop down option provides the list of modifications that can be applied to the current displayed cut sets. Each option will be discussed along with the outcome of the option.



Apply Change Set

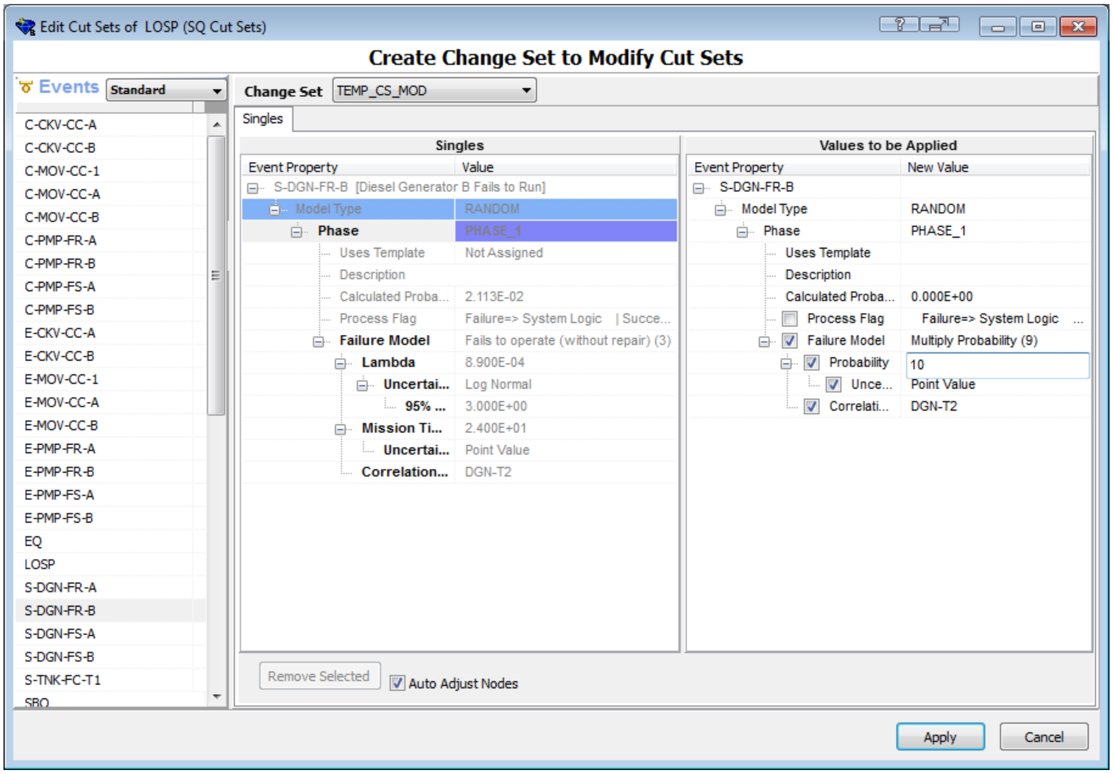
- ♦ The apply change set functions the same as the change set option. The difference is this becomes a temporary change set that is applied the current display of cut sets.

- ♦ Click the drop down option, select the Apply Change Set option and then select the Go button 

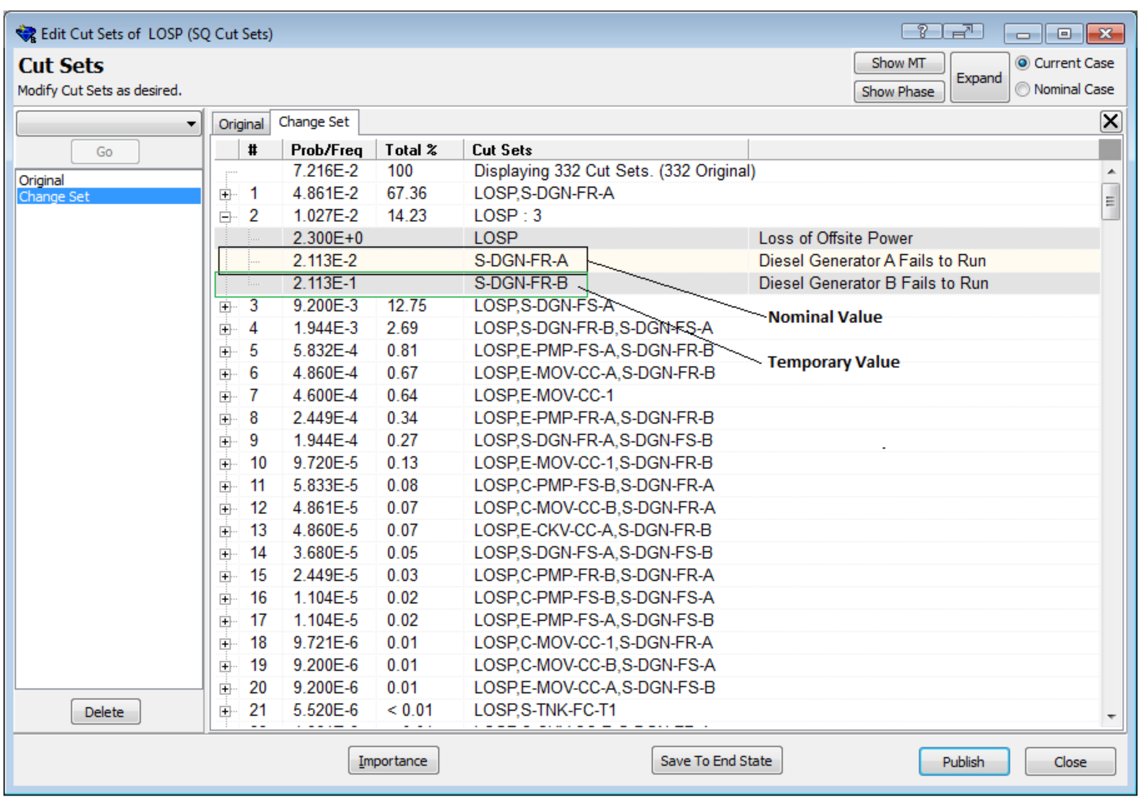
.

- ♦ Drag and drop the basic event of interest into the **Singles** column and then make the change of interest in the **Values to be Applied** column.

  - ◊ For this example, the diesel generator B fails to run (S-DGN-FR-B) probability will be increased by a factor of 10.

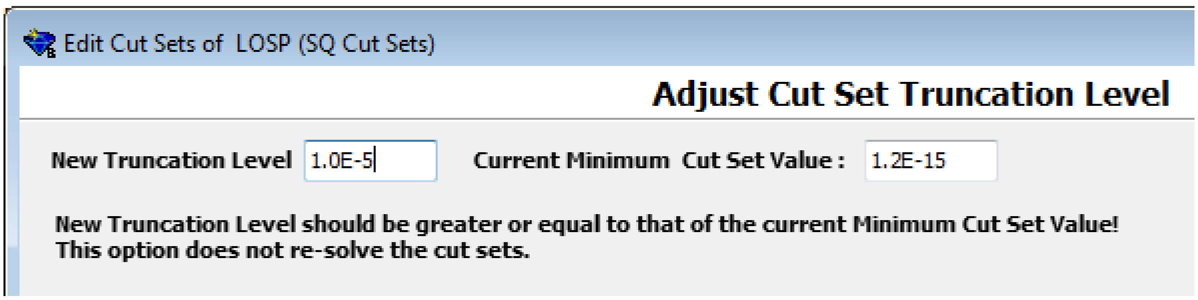


◊ Select Apply , the change set will be applied to the cuts.

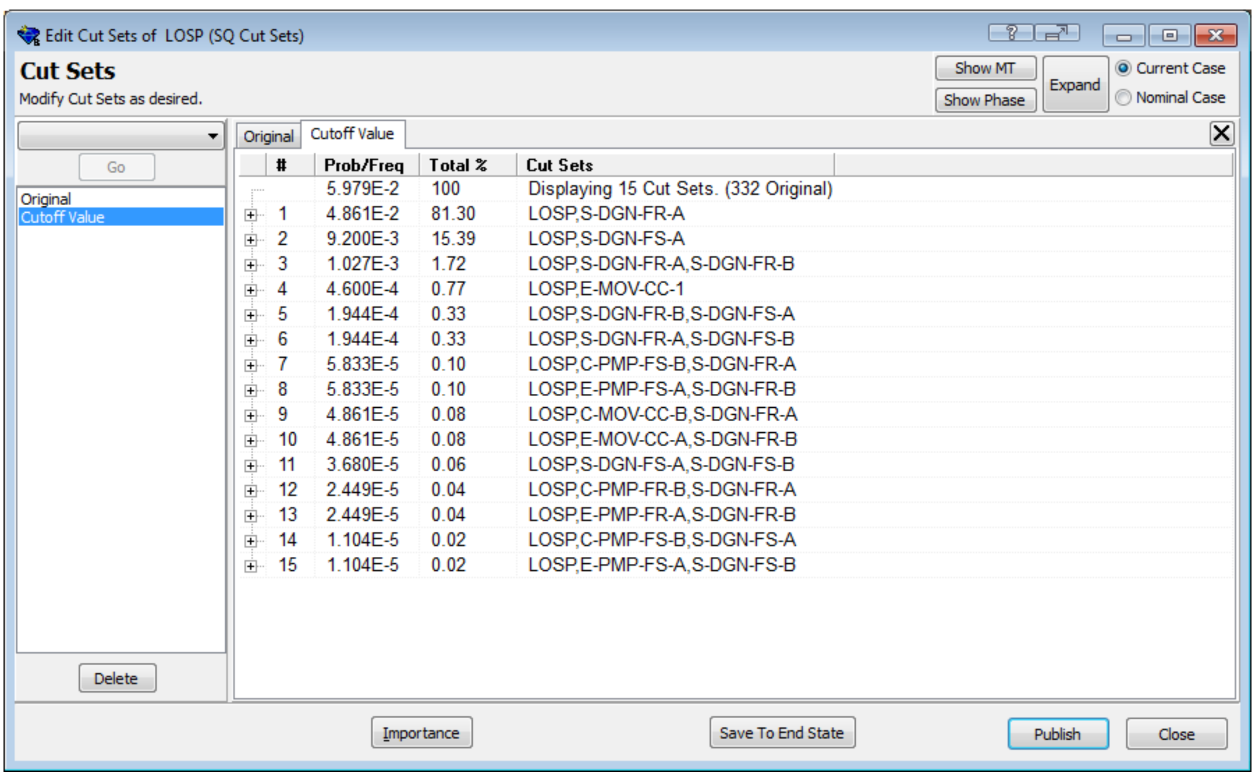


Truncation Level

- The truncation level allows the analyst to re-quantify the current cut sets at a new higher truncation.
- Click the drop down option, select the Truncation Level option and then select the Go button Go.



- For this example, the truncation level was increased to 1.0E-05 versus the default of 1.0E-15. This display option shows the value of the minimum cut set.

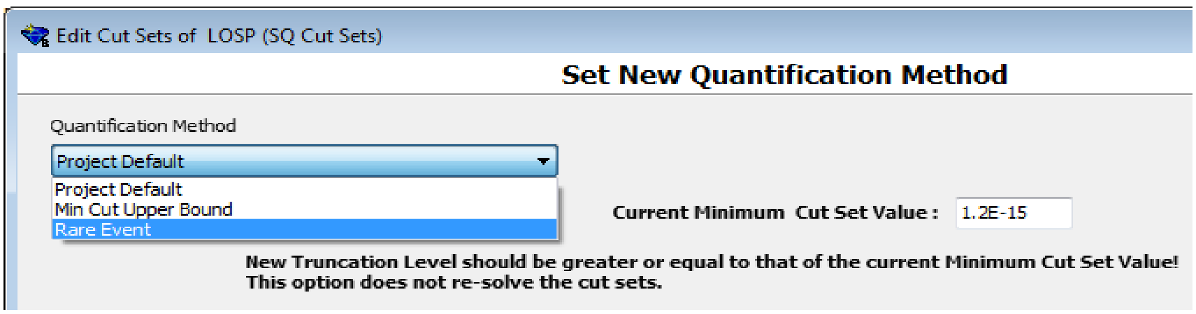


Quantification Method

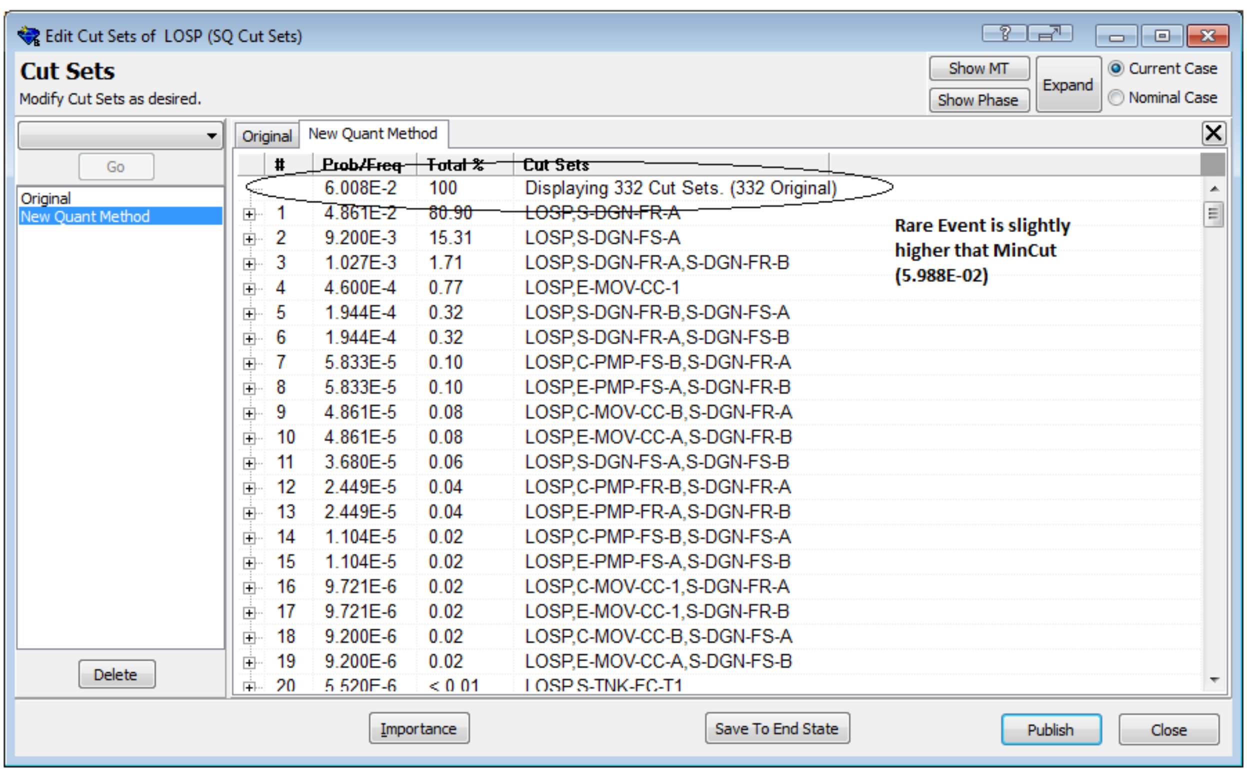
♦ The quantification method provides the analyst the option to use the minimal cut sets upperbound approximation or rare event approximation on the currently displayed cut sets.

♦ Click the drop down option, select the Quantification Method option and then select the Go button 

.



♦ For this example, the quantification was changed to rare event. This display option shows the value of the minimum cut set.



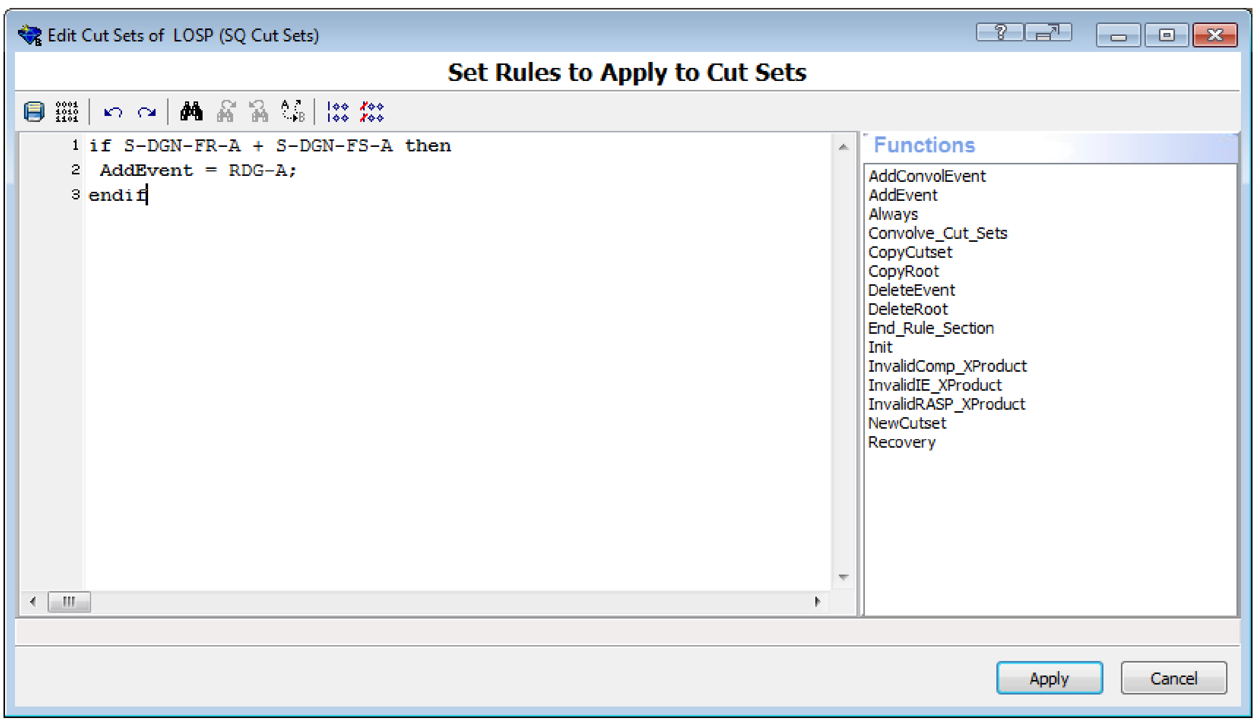
Post-Processing Rules

- The post-processing rules operate the same as discussed in Section 5. The difference is this rule is temporary and will only be applied to the currently displayed set of cut sets.

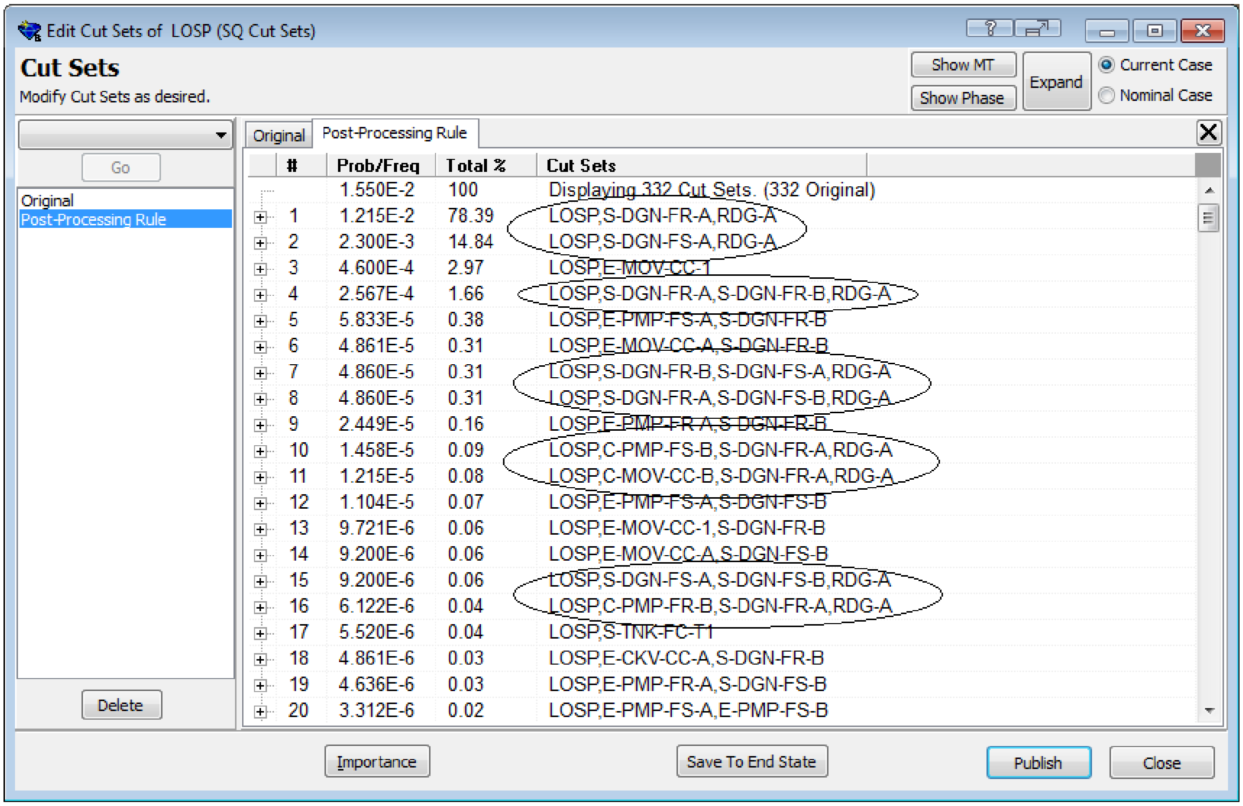
- Click the drop down option, select the Post-Processing Rules option and then select the Go button Go.

- The rule is created in the same format as discussed in Section 5. Once the rule has been created, select the compile option to make sure the rule is correct.

  ◊ For this example, a new basic event will be added every cut set that contains diesel generator A fails to run and start (S-DGN-FR-A and S-DGN-FS-A).



  ◊ Select Apply Apply, the change set will be applied to the cuts.

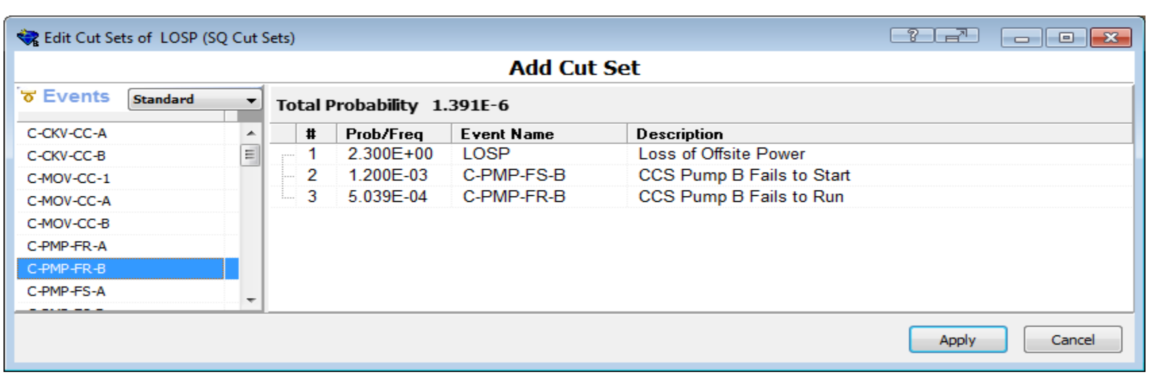


Add, Modify, and Remove Cut Set

- These options work on individual cut sets. The analyst will select one of these options and then highlight the cut set of interest. Select the Go button and SAPHIRE will perform the requested option.

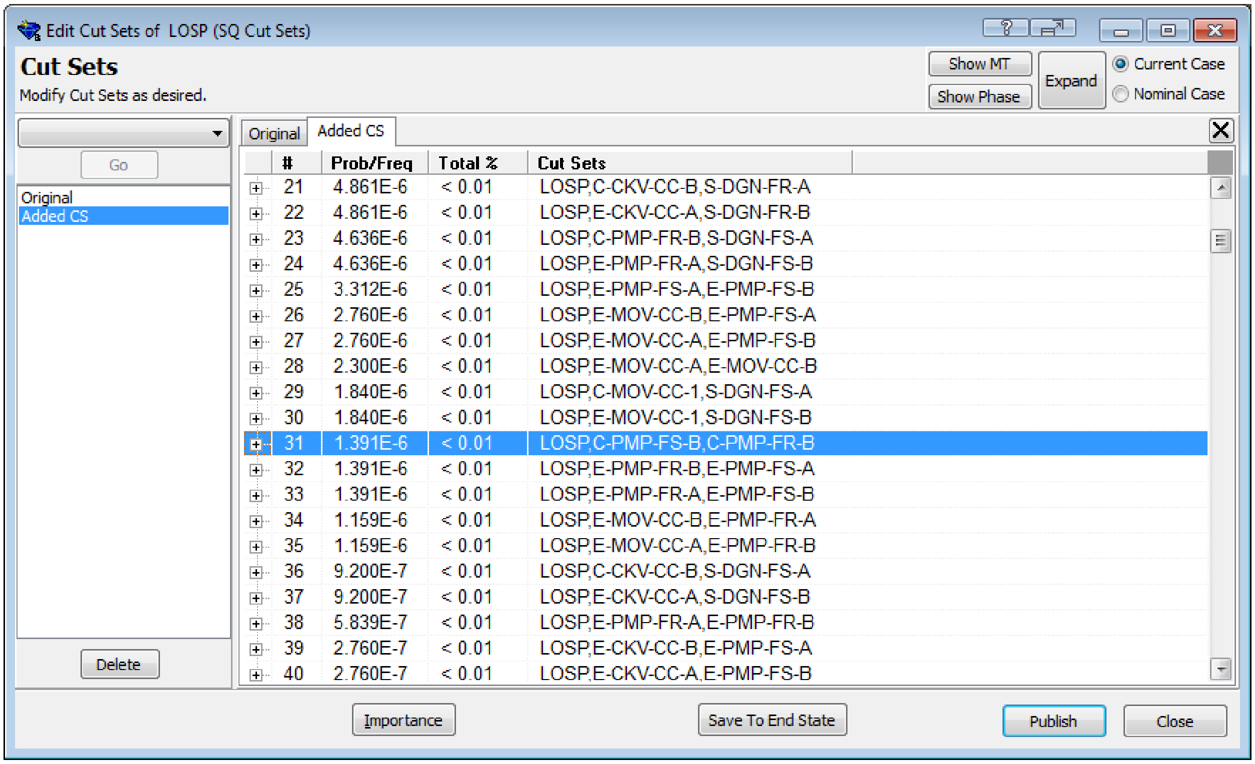
  - ◊ The **Add Cut Set** option allows a new cut set to be added directly into the list of cut sets.

  - ◊ Since no Boolean reduction goes on (unless specified) any cut set can be created by dragging an event from the list and placing it into the open field. SAPHIRE will then calculate the cut set frequency (probability) and place it into correct area depending upon sorted frequency (probability).



◊ Select Apply 

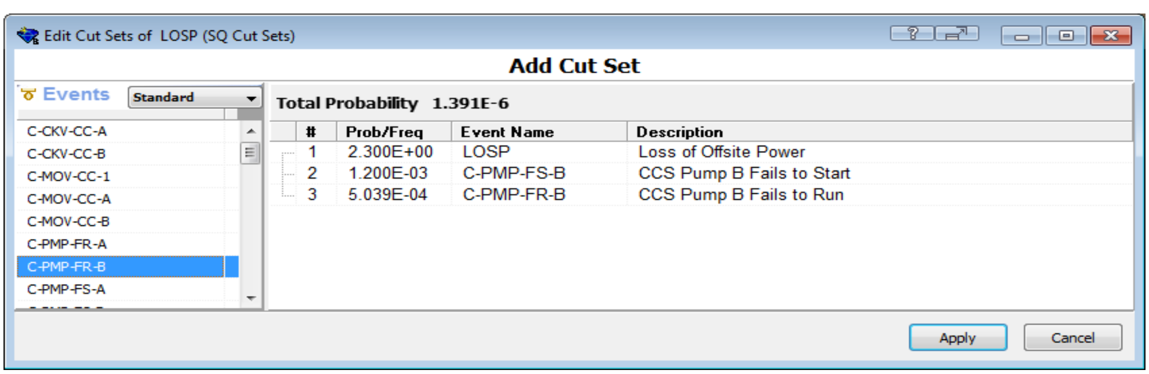
, the change set will be applied to the cuts.



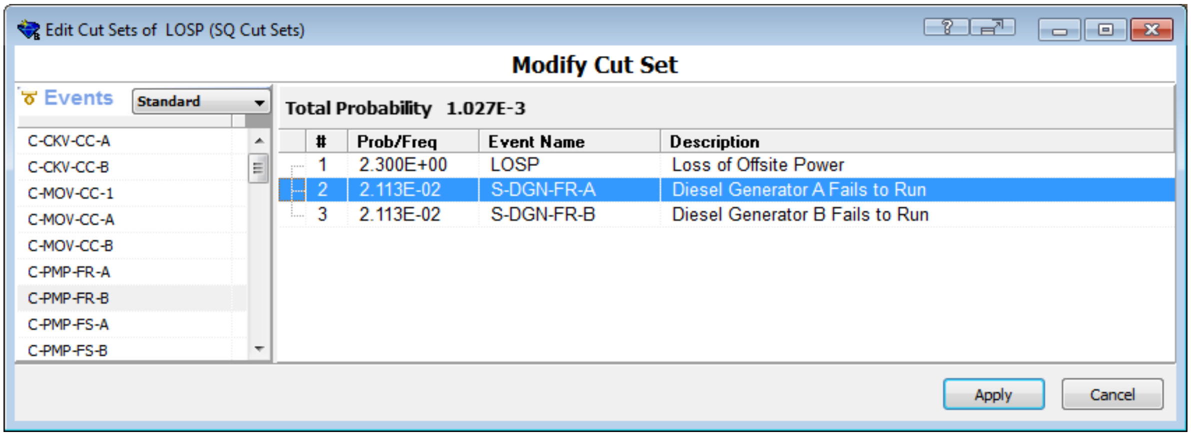
◊ The **Modify Cut Set** option allows an existing cut set to be modified (i.e., a basic event removed or a recovery event added directly into the cut set.

◊ Since no Boolean reduction goes on (unless specified) any cut set can be modified and create a non-minimal cut set.

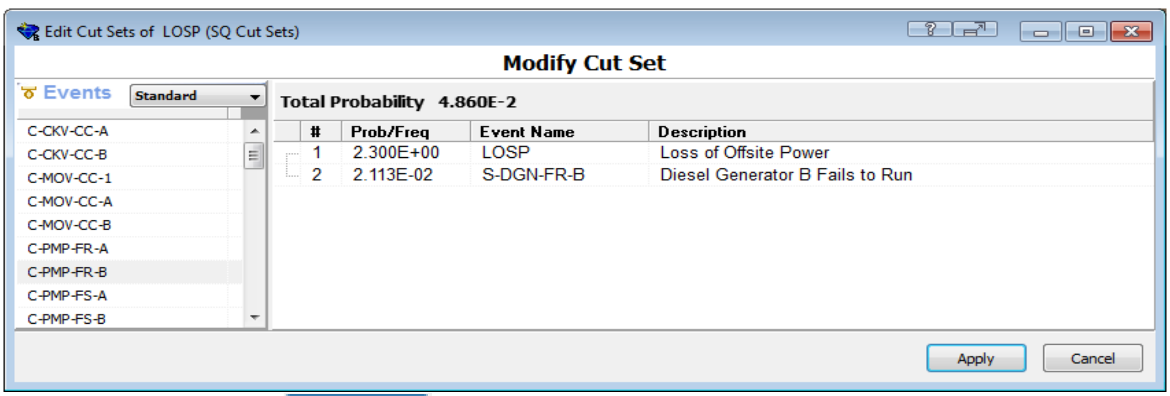
◊ The following example will remove (S-DGN-FR-A) from the cut set of interest.



◊ Highlight the cut set and from the drop down option select Modify Cut Set and select Go.

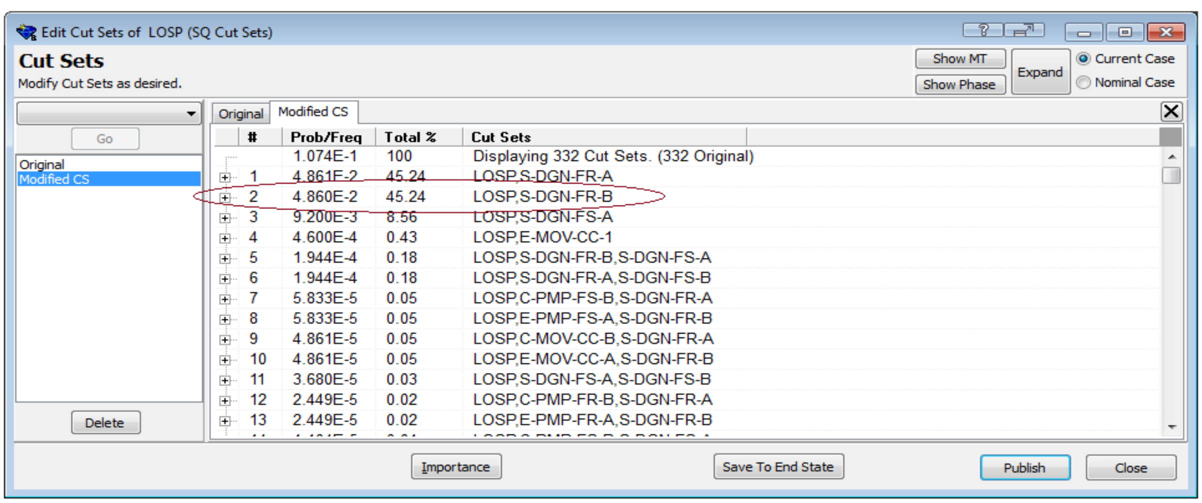


◊ Highlight the basic event that needs to be deleted from the cut set and then right mouse click and select from the pop-up menu "Delete".



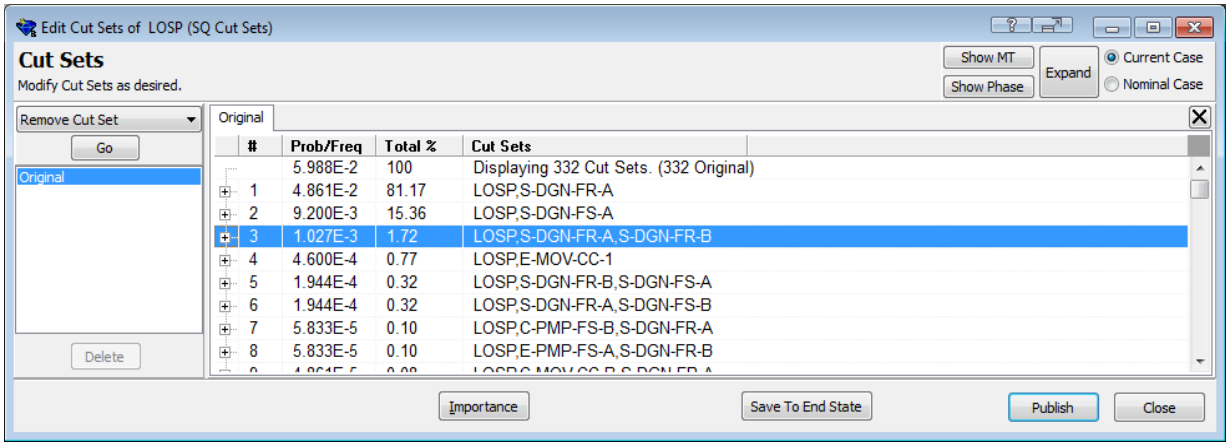
◊ Select Apply 

, and the newly modified cut set will now show up in the cut set list at the sorted frequency (probability).



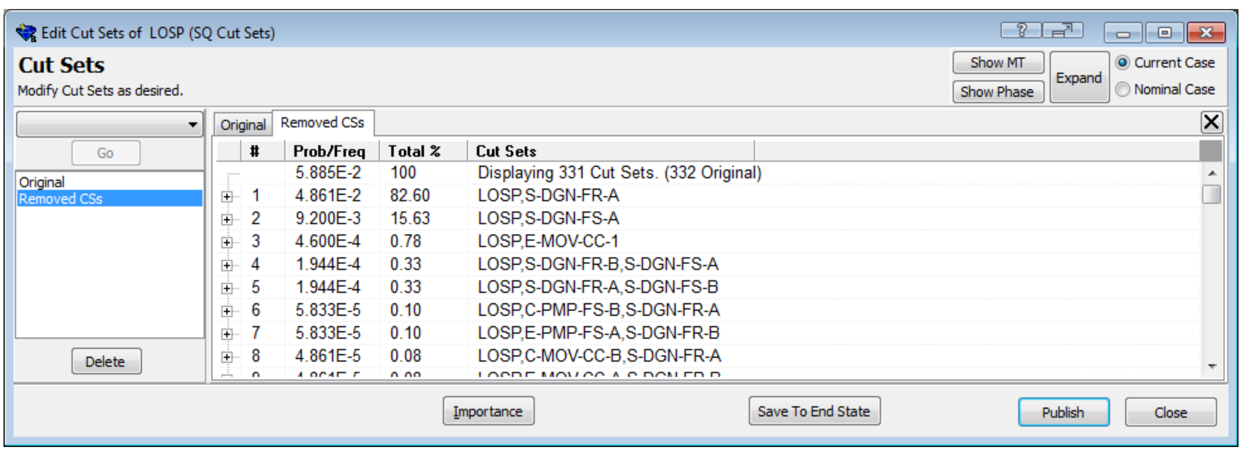
◊ The **Remove Cut Set** option removes the selected cut set from the displayed set of cut sets.

◊ Highlight the cut set that needs to be removed from the list and select Remove Cut Set from the drop down menu.



◊ Select the Go button 

and the cut set will be removed from the list.



Slice and Invert Slice Cut Sets

♦ The slice option work allows the analyst to parse the displayed cut sets up into different groups based on search criteria. The invert slice option will then display those cut sets that did not meet the search criteria.

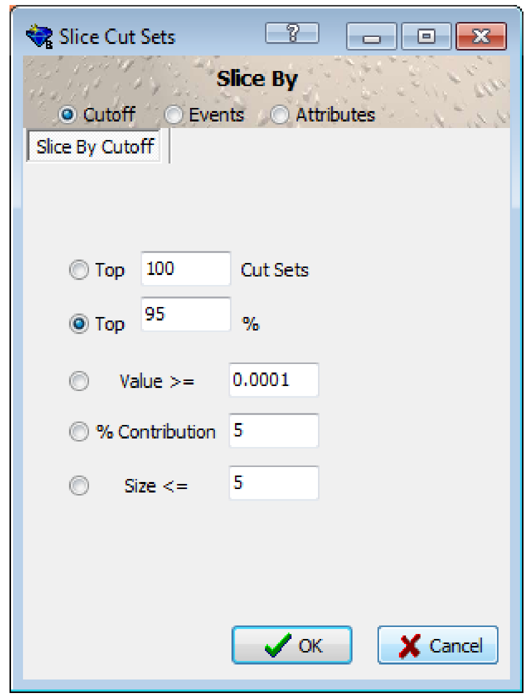
◊ The **Slice Cut Set** option will create a new group of cut sets based on the search criteria.

◊ Click the drop down option, select the Slice option and then select the Go button 

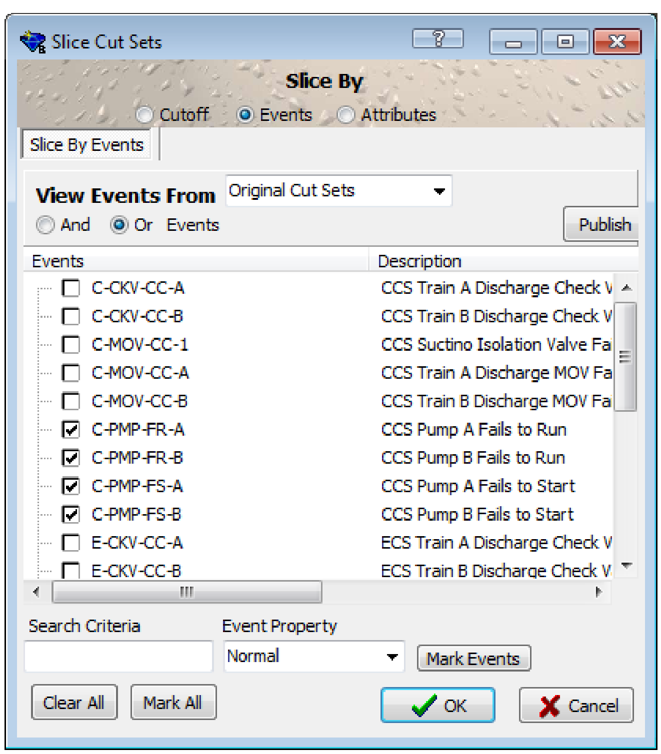
.

◊ The cut sets can be parsed based on different cut off options:

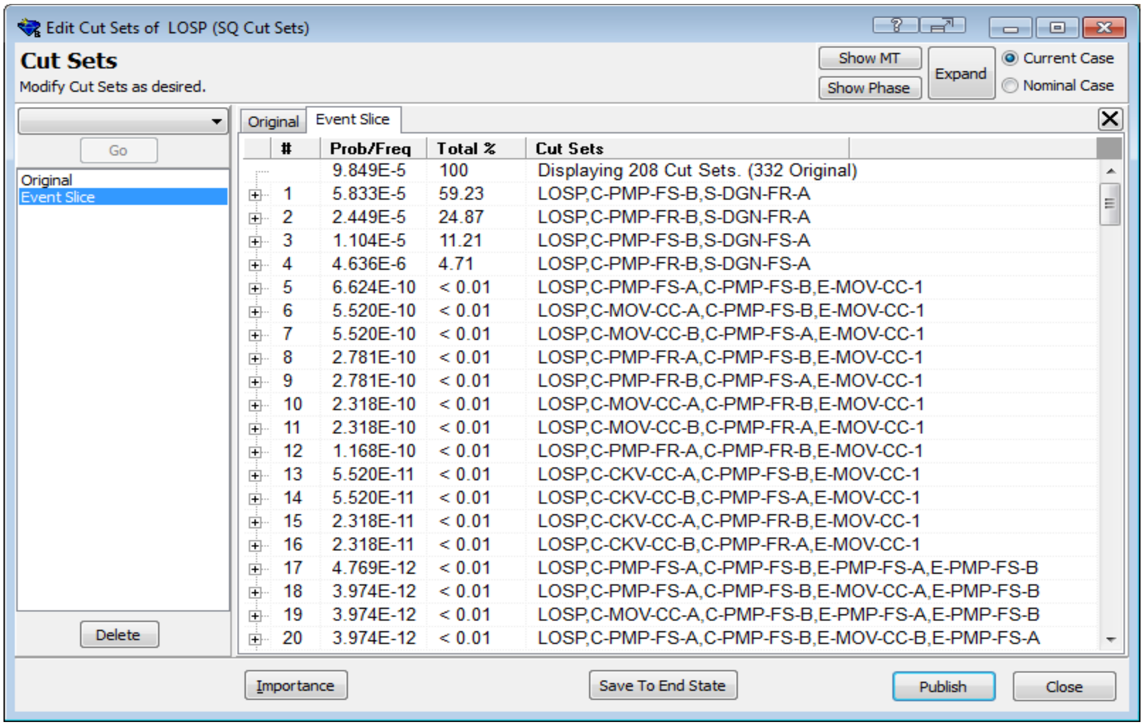
1. Top X Cut Sets (i.e., top 10 dominant cut sets)
2. Top Y % (i.e., top 90 percent dominant cut sets)
3. Value (cut sets that have a frequency (probability) greater than some specified value [i.e., cut sets greater than 1E-5])
4. % Contribution (those cut sets that contribute greater than some specified value to the overall [i.e., cut sets contribute greater than 5 percent])
5. Size (cut sets that contain at least that number of basic events or less [i.e., size of 3 will display only cut sets that have 3 or fewer basic events])



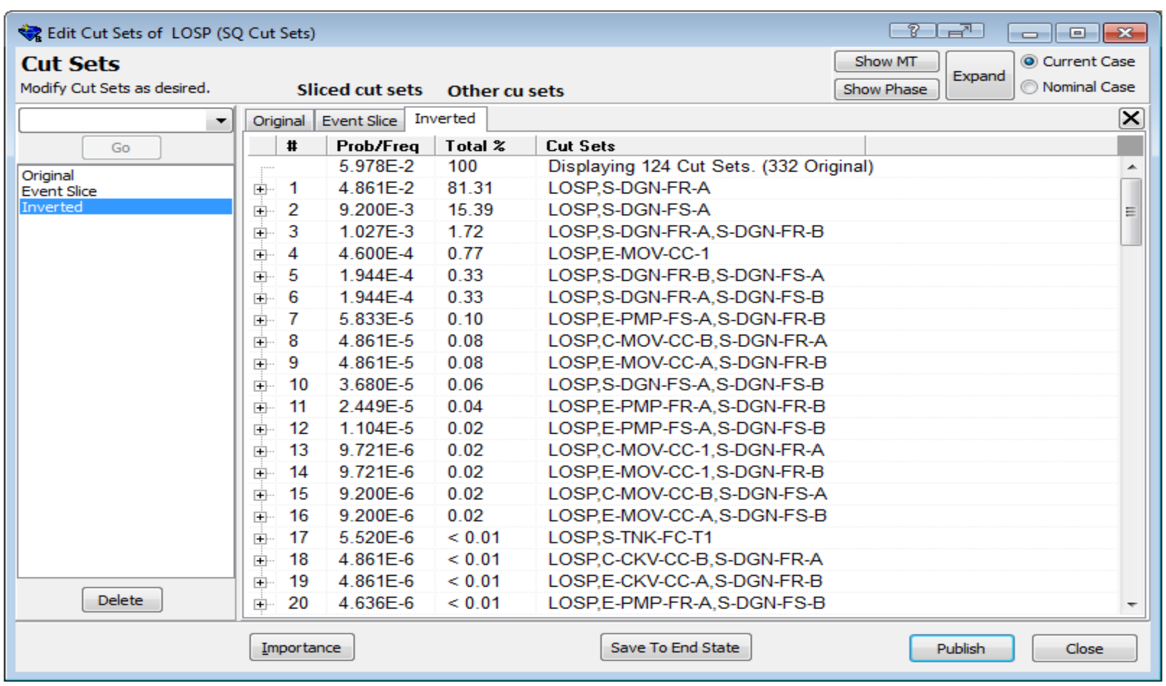
◊ The cut sets can be parsed by an individual basic event



◊ For this example, the C-PMP-(all) components are selected and will be parsed into one group of cut sets.



◊ The invert cut sets can now be displayed. This option will list all those cut sets that did not meet the search criteria (i.e., all cut sets that do not contain C-PMP-(all)).

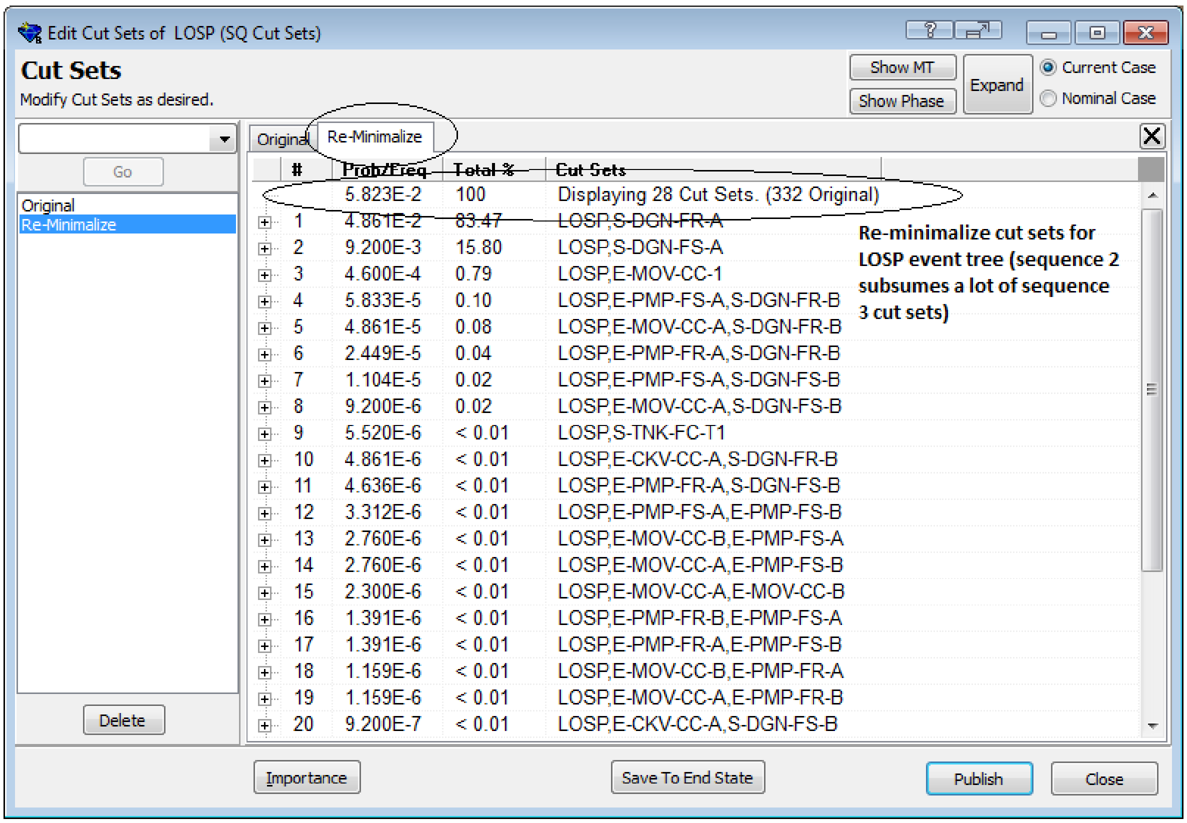


◊ The display option keeps a running list of the modified cut sets. The analyst can now move from one modification to another by selecting the Tab of interest.

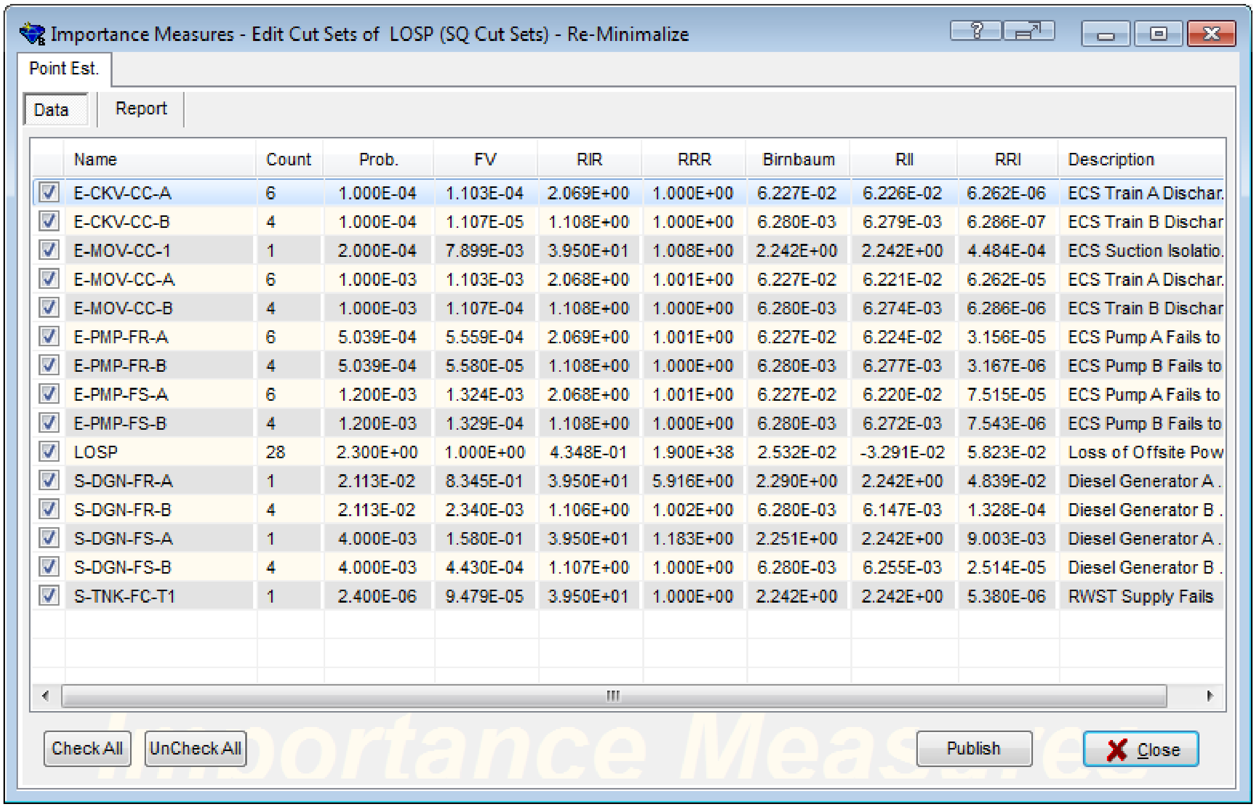
Re-minimalize Cut Sets

- The Re-minimalize option will take the displayed cut sets and perform Boolean algebra on these cut sets. This option may be required depending upon which editing option was performed.

  ◊ The **Re-minimalize Cut Set** option will remove non-minimal cut sets (based on the existing list not taking into account the sequence information [i.e., success logic]).



♦ Importance measures for the displayed set of cut sets can be generate. Select the importance button on the bottom and the basic events that are shown in the displayed set of cut sets will have their importance measures calculated.



Name	Count	Prob.	FV	RIR	RRR	Birnbaum	RII	RRI	Description
E-CKV-CC-A	6	1.000E-04	1.103E-04	2.069E+00	1.000E+00	6.227E-02	6.226E-02	6.262E-06	ECS Train A Dischar.
E-CKV-CC-B	4	1.000E-04	1.107E-05	1.108E+00	1.000E+00	6.280E-03	6.279E-03	6.286E-07	ECS Train B Dischar
E-MOV-CC-1	1	2.000E-04	7.899E-03	3.950E+01	1.008E+00	2.242E+00	2.242E+00	4.484E-04	ECS Suction Isolatio.
E-MOV-CC-A	6	1.000E-03	1.103E-03	2.068E+00	1.001E+00	6.227E-02	6.221E-02	6.262E-05	ECS Train A Dischar.
E-MOV-CC-B	4	1.000E-03	1.107E-04	1.108E+00	1.000E+00	6.280E-03	6.274E-03	6.286E-06	ECS Train B Dischar
E-PMP-FR-A	6	5.039E-04	5.559E-04	2.069E+00	1.001E+00	6.227E-02	6.224E-02	3.156E-05	ECS Pump A Fails to
E-PMP-FR-B	4	5.039E-04	5.580E-05	1.108E+00	1.000E+00	6.280E-03	6.277E-03	3.167E-06	ECS Pump B Fails to
E-PMP-FS-A	6	1.200E-03	1.324E-03	2.068E+00	1.001E+00	6.227E-02	6.220E-02	7.515E-05	ECS Pump A Fails to
E-PMP-FS-B	4	1.200E-03	1.329E-04	1.108E+00	1.000E+00	6.280E-03	6.272E-03	7.543E-06	ECS Pump B Fails to
LOSP	28	2.300E+00	1.000E+00	4.348E-01	1.900E+38	2.532E-02	-3.291E-02	5.823E-02	Loss of Offsite Pow
S-DGN-FR-A	1	2.113E-02	8.345E-01	3.950E+01	5.916E+00	2.290E+00	2.242E+00	4.839E-02	Diesel Generator A .
S-DGN-FR-B	4	2.113E-02	2.340E-03	1.106E+00	1.002E+00	6.280E-03	6.147E-03	1.328E-04	Diesel Generator B .
S-DGN-FS-A	1	4.000E-03	1.580E-01	3.950E+01	1.183E+00	2.251E+00	2.242E+00	9.003E-03	Diesel Generator A .
S-DGN-FS-B	4	4.000E-03	4.430E-04	1.107E+00	1.000E+00	6.280E-03	6.255E-03	2.514E-05	Diesel Generator B .
S-TNK-FC-T1	1	2.400E-06	9.479E-05	3.950E+01	1.000E+00	2.242E+00	2.242E+00	5.380E-06	RWST Supply Fails

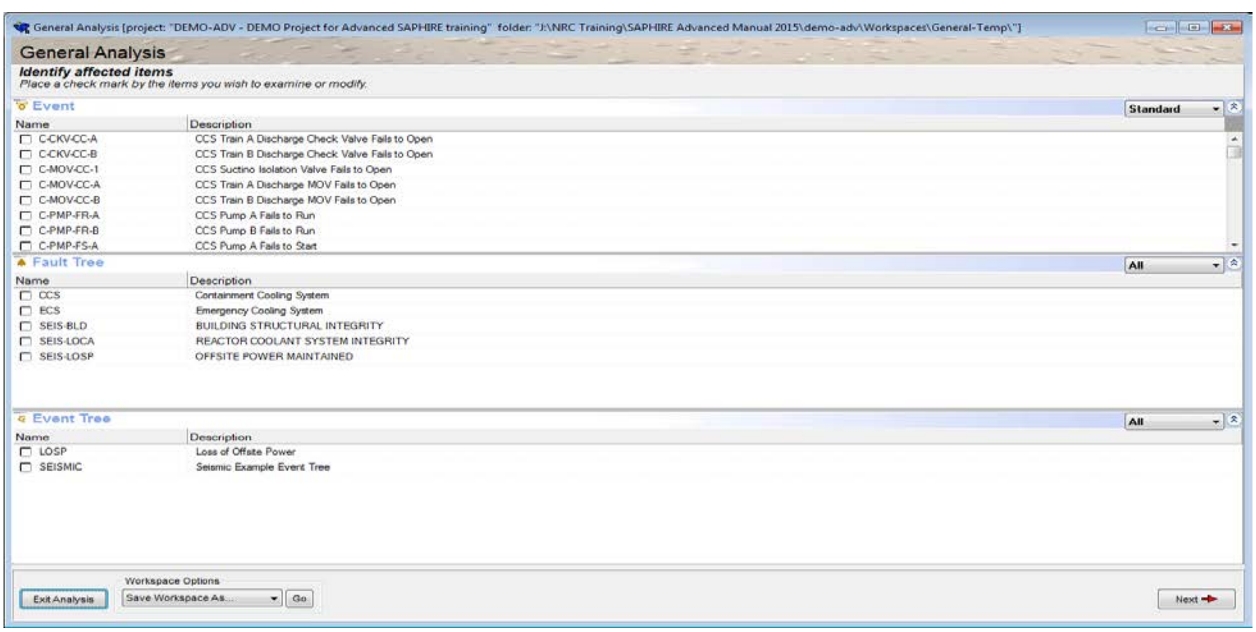
| 11 | GENERAL ANALYSIS FOR SENSITIVITY STUDIES

Section 11 discusses the General Analysis workspace. When compared to the Change Set method presented above and in the SAPHIRE Basics manual, General Analysis is more versatile for logic changes made to Fault Trees and Event Trees and less versatile for Basic Event changes. General Analysis is the only way to temporarily change the logic of Fault Trees and Event Trees. The Basic Event data is limited to changing the basic event probability and does not allow changes to the failure model for the event.

11.1. General Analysis Example

General Analysis is found in the Workspaces List Panel. To start a new General Analysis, perform the following:

- ♦ Double click on **New Analysis…** under General Analysis in the Workspaces List Panel. Alternately, right click on **New Analysis…** and choose **Open** from the pop-up.
- ♦ The General Analysis worksheet will open with lists of Basic Events, Fault Trees, and Event Trees. The DEMO General Analysis worksheet is shown here:



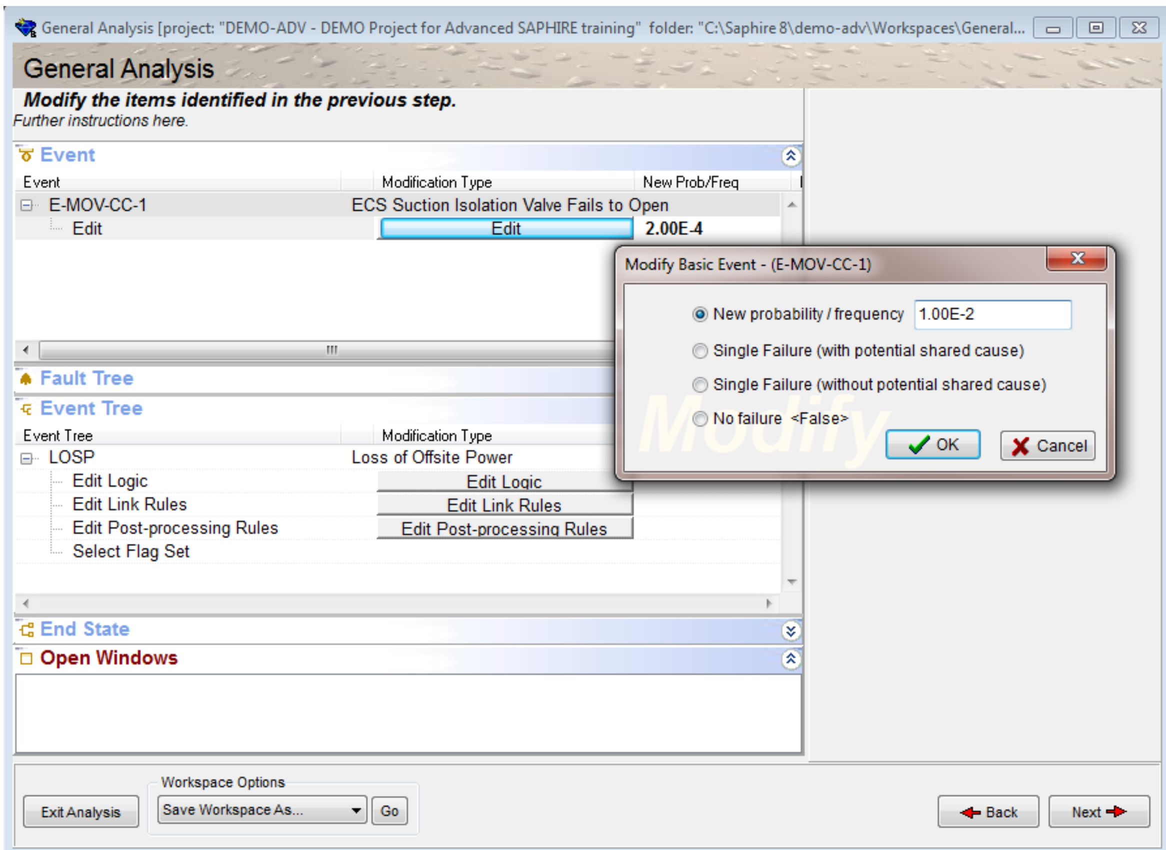
- ♦ Choose the Basic Events, Fault Trees, and Event Trees for modification by clicking on the appropriate check boxes. In this example, we will modify Basic Event E-MOV-CC-1 and the LOSP Event Tree

- ♦ Click on 

to proceed

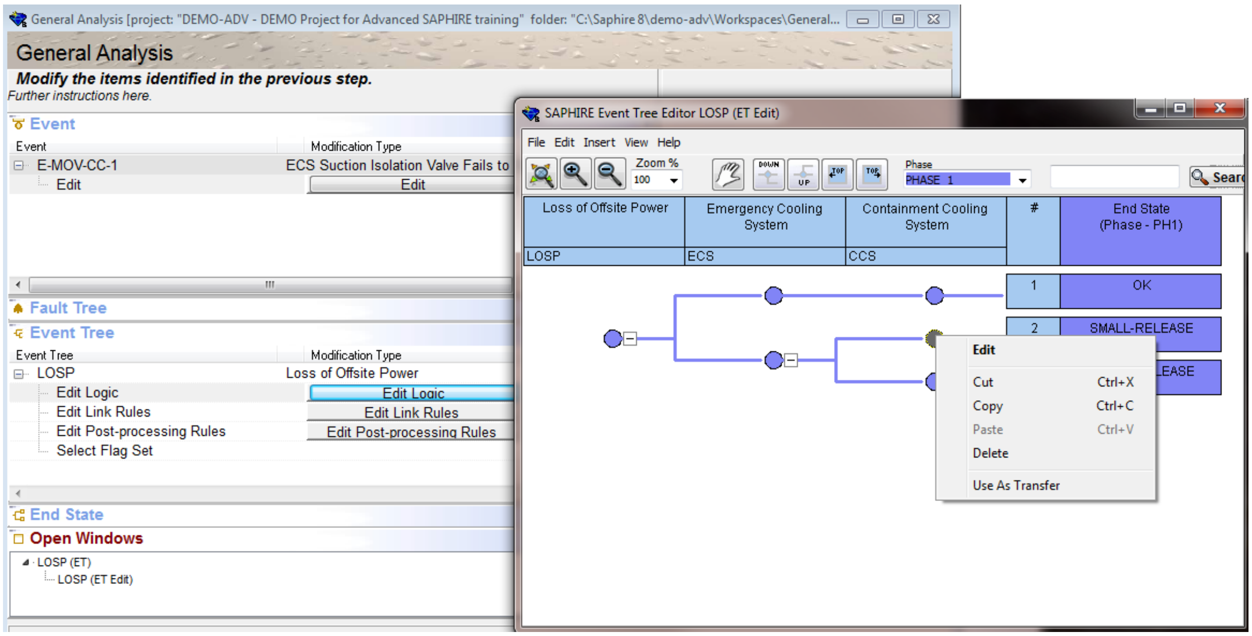
- ♦ The General Analysis Editing worksheet will appear.

  - ◊ Editing buttons are available on this screen for the items selected for edit on the prior worksheet. The image below shows E-MOV-CC-1 after selection by clicking on Edit. The basic event's probability is changed to 1.00E-2.

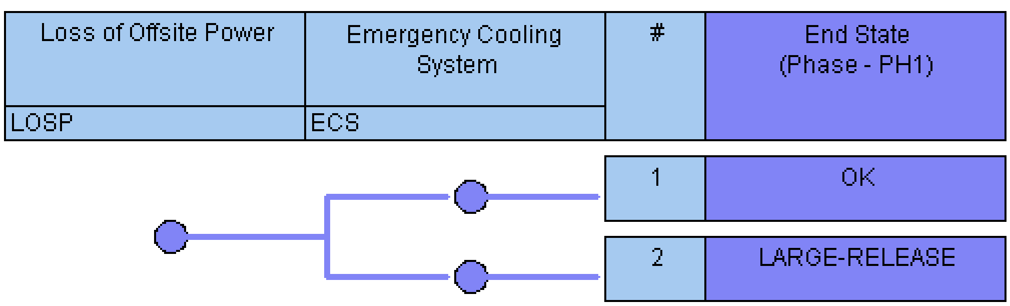


- ♦ Perform the change shown on the probability of E-MOV-1 to 1.00E-2 and then click on the Edit Logic button for the LOSP Event Tree. The Event Tree graphical editor will appear.

- ♦ Remove the SMALL-RELEASE branch for Containment Cooling System by right clicking on the branch node and selecting Delete.

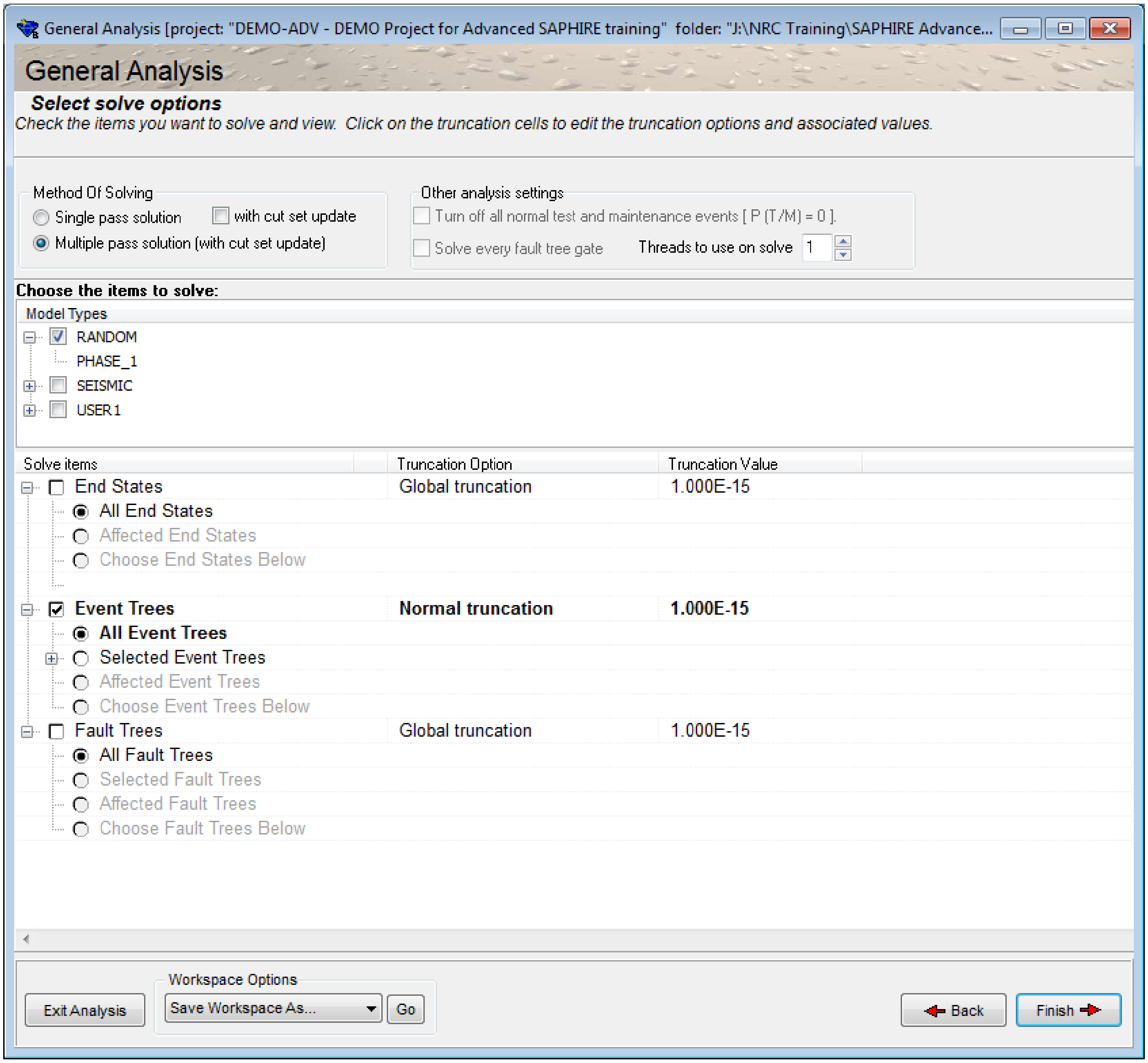


♦ Remove the Containment Cooling System top event by right clicking on the top event box and selecting Delete.



♦ Close and save the Event Tree editor. It should just have one top event, Emergency Cooling System with a single branch. The lower branch should be LARGE-RELEASE.

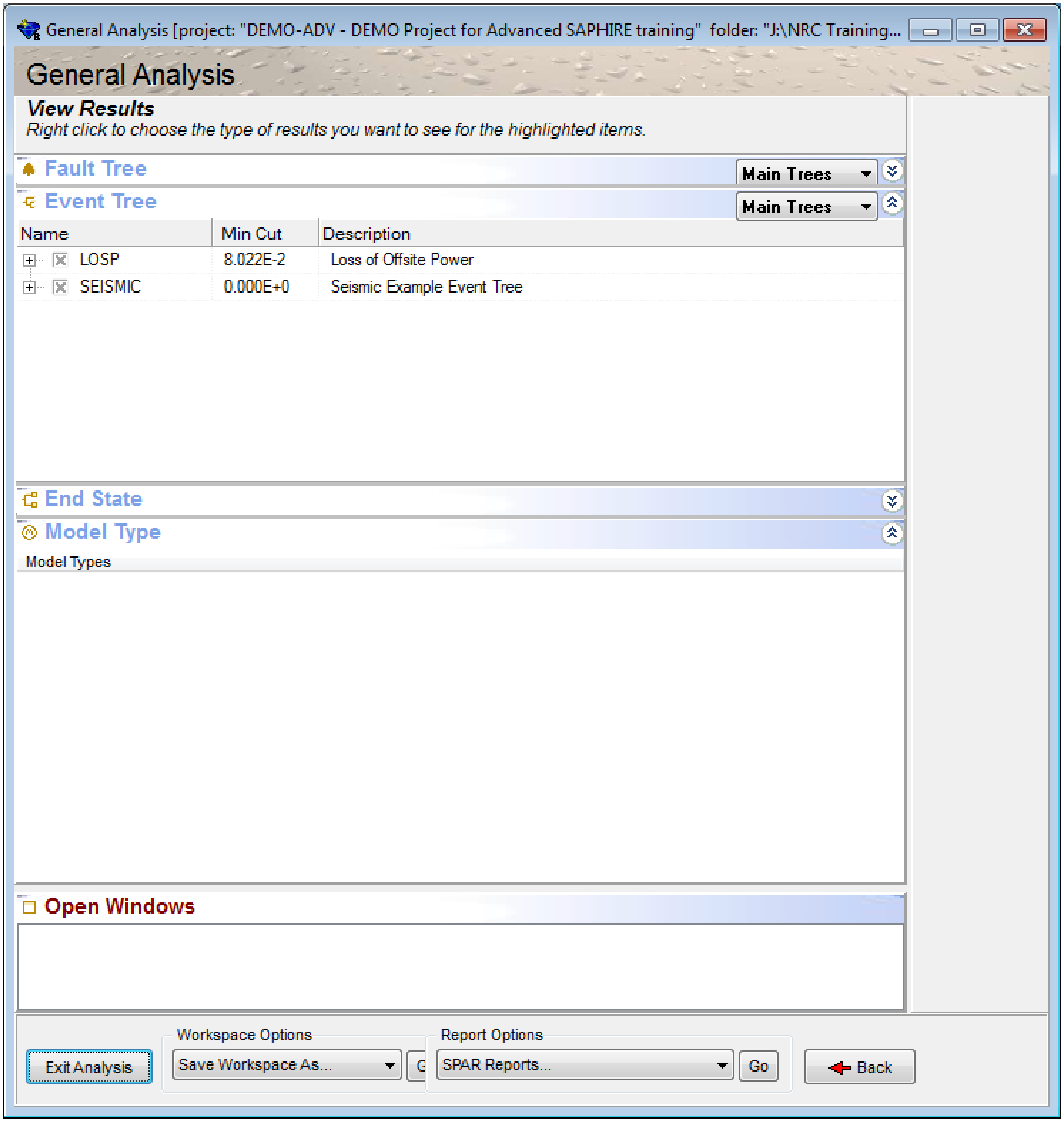
♦ Click  and the General Analysis Solve Worksheet will appear. Selections here can be made for what items to solve, their truncation values and the method of solving. A Single pass solution is faster for large models, but less accurate overall. In the case of DEMO model, the Multiple pass solution is desirable. It is ok to use all the defaults as shown:



- Click on  and then OK on the confirmation message to perform the analysis.

- The results screen will show the Min Cut for the LOSP event tree. View the Fault Trees list panel by clicking on the  to the right of the Fault Tree menu.

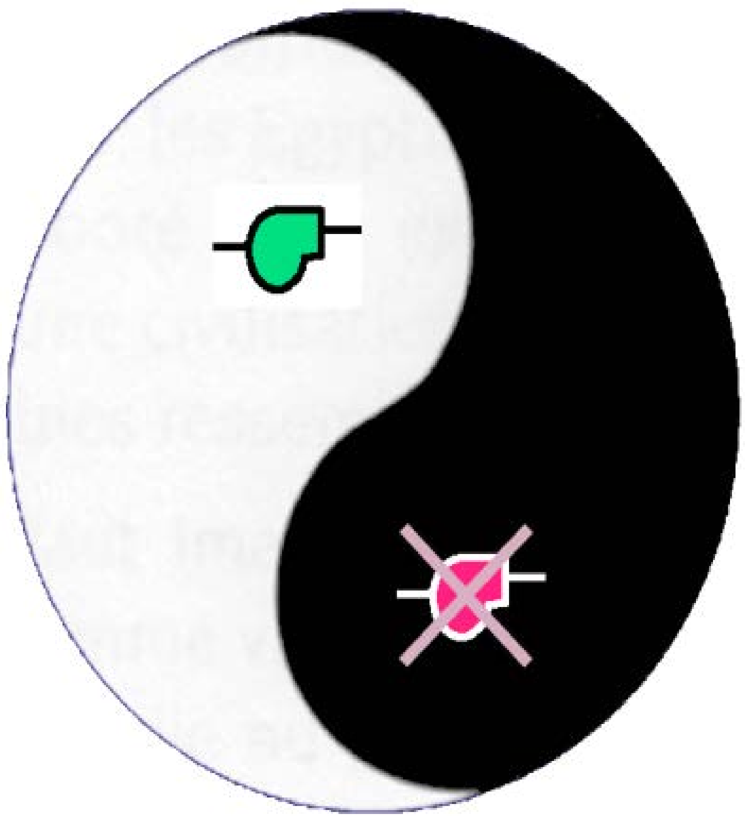
- Right clicking on LOSP or The ECS Fault Tree will provide options to view the modified cut sets, uncertainty and importance measures through summary results or cut set lists.


General Analysis [project: "DEMO-ADV - DEMO Project for Advanced SAPHIRE training" folder: "J:\NRC Training...
General Analysis
View Results
Right click to choose the type of results you want to see for the highlighted items.
Fault Tree
Main Trees
Event Tree
Main Trees
Name
Min Cut
Description
LOSP
8.022E-2
Loss of Offsite Power
SEISMIC
0.000E+0
Seismic Example Event Tree
End State
Model Type
Model Types
Open Windows
Exit Analysis
Workspace Options
Save Workspace As...
Report Options
SPAR Reports...
Go
Back

NOTES

| 12 | MUTUALLY EXCLUSIVE EVENTS

Section 12 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.



12.1. 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 or more 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 cannot be both failed and working (success) in the same cut set.

Most mutually exclusive groups include only two or more 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.

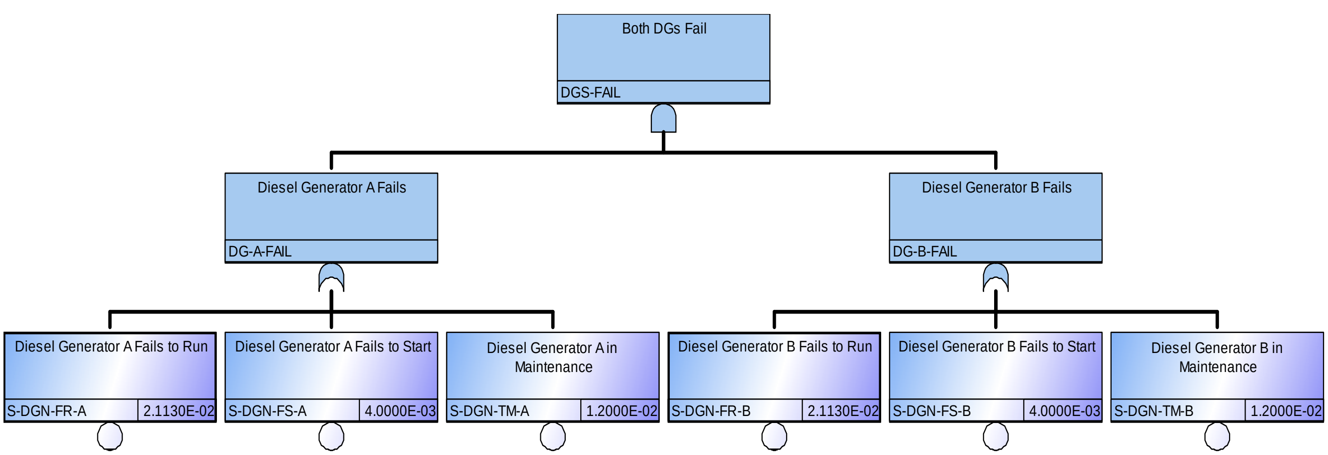
- Other unrecognized mutually exclusive events may not be evident until the analyst solves and evaluates the fault tree or sequence cut sets.

12.1.1. Mutually Exclusive Event Example

The fault tree logic below will produce a cut set containing the two maintenance events

S-DGN-TM-A * S-DGN-TM-B

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.



There are different methods to remove cut sets containing mutually exclusive events. Two methods are discussed below, in order of increasing preference:

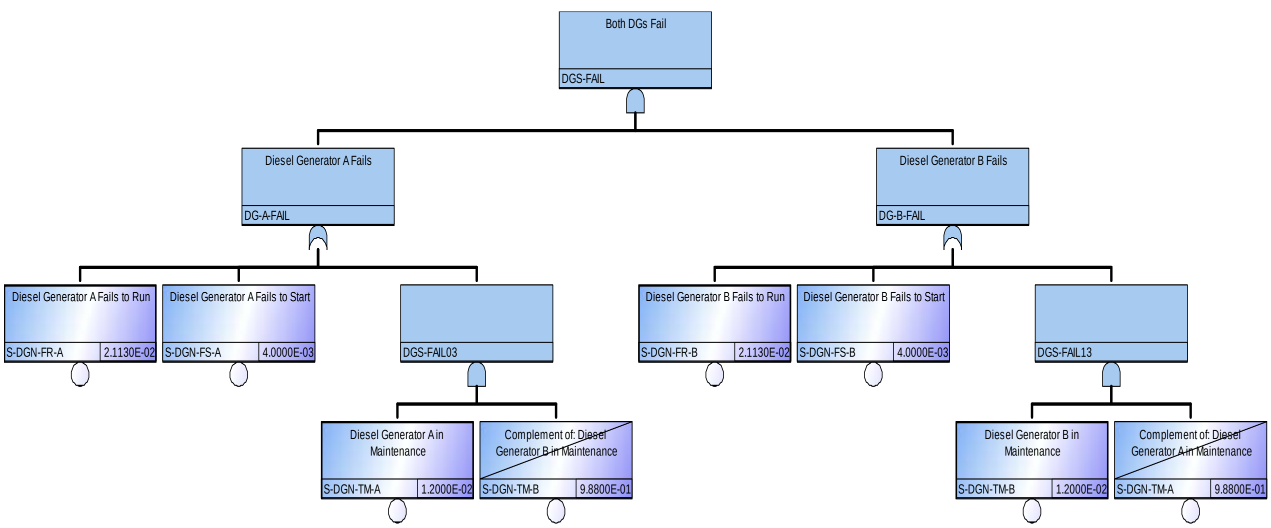
- Modify logic models (via NOT gates or complemented events) to remove prohibited combinations of events.
- Using the Post-processing Rules to define combinations of events in cut sets that would be deleted (via the **DeleteRoot** keyword).

Let us discuss these methods in turn.

12.1.1.1. 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 below. 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.



12.2. Mutually Exclusive Event Removal via Post-processing Rules

Post-processing 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.

During cut set generation, the post-processing rules may be automatically applied. Thus,

- ♦ No changes to logic models are needed.
- ♦ No manual manipulations to cut sets are required.

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.

Apply or edit the *FAULT TREE* Post-processing Rules:

- ♦ To edit the Post-processing Rules for a particular fault tree, highlight the fault tree name in the Fault Trees list panel, right click the mouse and select **Edit Post-processing Rules**.
- ♦ To edit the Post-processing Rules for all fault trees, select from the project main menu **Project → Edit Rules → FT (Post-processing)**.

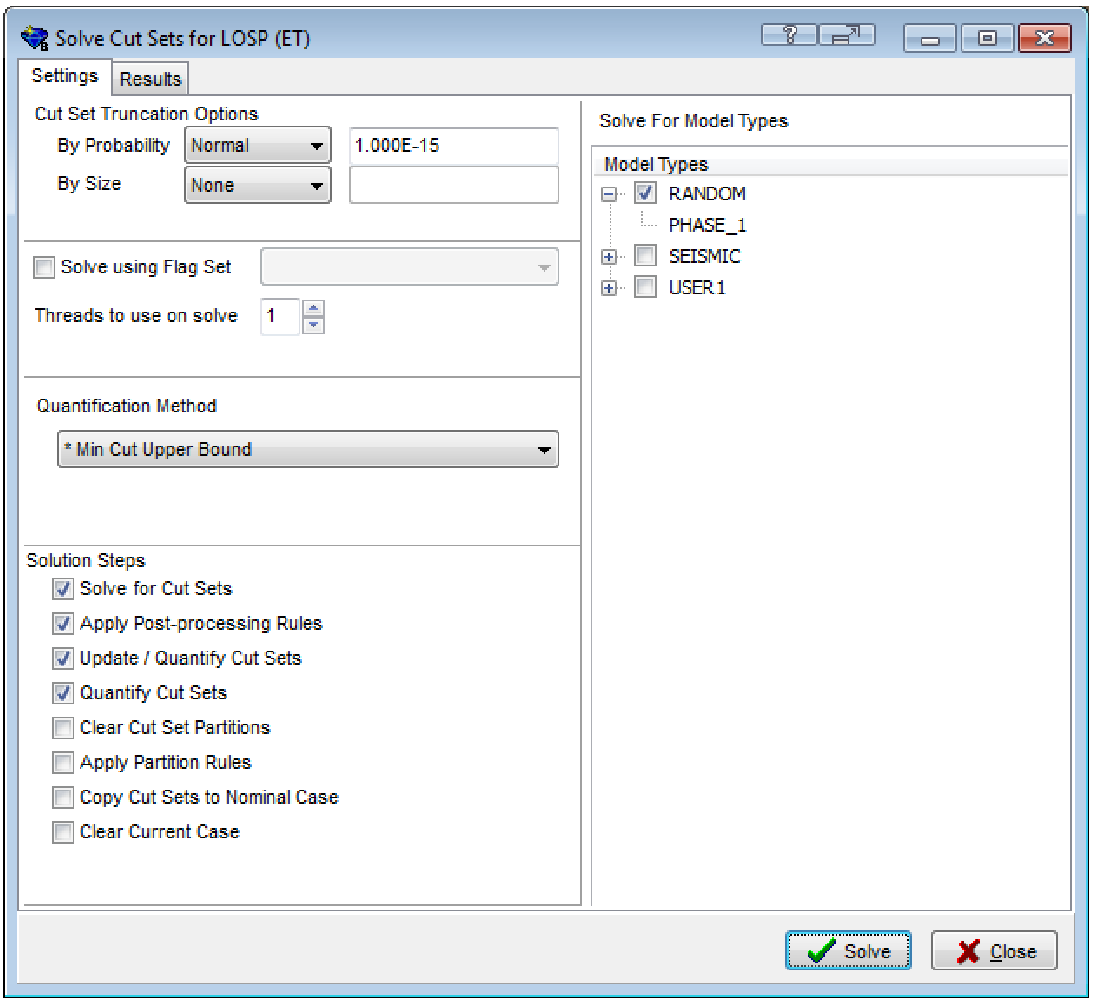
Apply or edit the *Event Tree* Post-processing Rules:

- ♦ To edit the Rules for a particular event tree, highlight the event tree name in the Event Trees list panel, right click, and select **Edit Post-processing Rules**.
- ♦ To edit the Post-processing Rules for all sequences, select from the project main menu **Project → Edit Rules → ET (Post-processing)**.

To demonstrate the uses of the Post-processing Rules, the example below shows how the rules can be used to remove the cut set containing both diesel generators failing to start from the DEMO project. Add a rule for the LOSP event tree **Post-processing Rules** as follows:

| This rule searches for both diesel generators failing
if S-DGN-TM-A * S-DGN-TM-B **then**
DeleteRoot; | Delete the cut set matching the search criteria
endif

Apply this Post-processing Rule when we solve the LOSP event tree for sequence cut sets (using no truncation).



Compare cut sets for sequence 3:

Sequence 3 changed from an original value of 3.688E-3/yr to a value of 3.357E-3/yr.

Cut Set Report - LOSP-3 (ET)	DEMO Project for Advanced SAPHIRE training
2/5/2015 9:35:29 AM LOSP-3 (ET)	Slice of - Cut Set Results

#	Cases	Prob/Freq	Total%	Cut Set
Total		3.688E-3	100	Displaying 328 Cut Sets. (328 Original)
1	C	1.027E-3	27.86	LOSP,S-DGN-FR-A,S-DGN-FR-B
2	C	5.833E-4	15.82	LOSP,S-DGN-FR-B,S-DGN-TM-A
3	C	5.833E-4	15.82	LOSP,S-DGN-FR-A,S-DGN-TM-B
4	C	3.312E-4	8.98	LOSP,S-DGN-TM-A,S-DGN-TM-B
5	C	1.944E-4	5.27	LOSP,S-DGN-FR-B,S-DGN-FS-A
6	C	1.944E-4	5.27	LOSP,S-DGN-FR-A,S-DGN-FS-B
7	C	1.104E-4	2.99	LOSP,S-DGN-FS-B,S-DGN-TM-A
8	C	1.104E-4	2.99	LOSP,S-DGN-FS-A,S-DGN-TM-B
9	C	5.833E-5	1.58	LOSP,C-PMP-FS-B,S-DGN-FR-A
10	C	5.833E-5	1.58	LOSP,E-PMP-FS-A,S-DGN-FR-B

Cut Set Report - LOSP-3 (ET)	DEMO Project for Advanced SAPHIRE training
2/5/2015 9:39:45 AM LOSP-3 (ET)	Slice of - Cut Set Results

#	Cases	Prob/Freq	Total%	Cut Set
Total		3.357E-3	100	Displaying 327 Cut Sets. (327 Original)
1	C	1.027E-3	30.60	LOSP,S-DGN-FR-A,S-DGN-FR-B
2	C	5.833E-4	17.38	LOSP,S-DGN-FR-B,S-DGN-TM-A
3	C	5.833E-4	17.38	LOSP,S-DGN-FR-A,S-DGN-TM-B
4	C	1.944E-4	5.79	LOSP,S-DGN-FR-B,S-DGN-FS-A
5	C	1.944E-4	5.79	LOSP,S-DGN-FR-A,S-DGN-FS-B
6	C	1.104E-4	3.29	LOSP,S-DGN-FS-B,S-DGN-TM-A
7	C	1.104E-4	3.29	LOSP,S-DGN-FS-A,S-DGN-TM-B
8	C	5.833E-5	1.74	LOSP,C-PMP-FS-B,S-DGN-FR-A
9	C	5.833E-5	1.74	LOSP,E-PMP-FS-A,S-DGN-FR-B
10	C	4.861E-5	1.45	LOSP,C-MOV-CC-B,S-DGN-FR-A

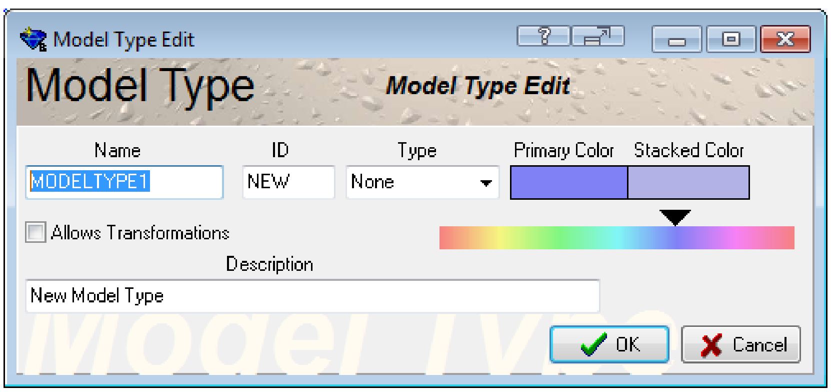
Note: Only *one* cut set was removed from sequence 3, but it happened to be the 4th dominant cut set.

| 13 | USING MODEL TYPES

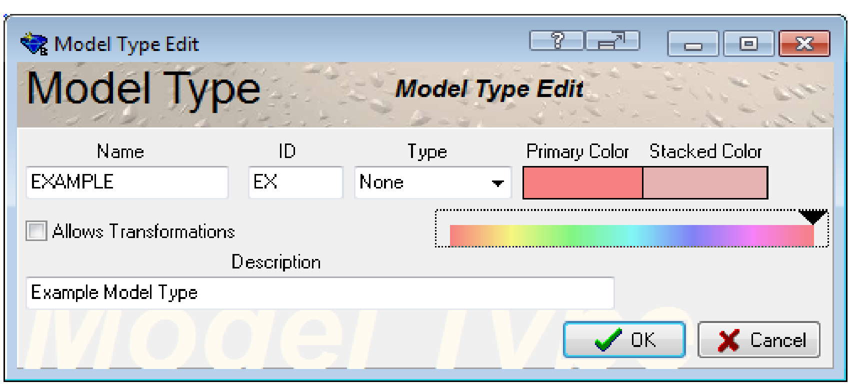
Section 14 introduces the different model types that can be used in SAPHIRE. Model types are designed to contain different failure potentials for components and different modeling options for user defined assessments. The following section will discuss the development of model types and how these can be used for evaluating seismic, fire, flood and user defined types.

13.1. User Defined Model Type

♦ To add a new Model Type, select **Project** → **Model Types** and click the **Go** option with the **Add Model Type** drop down option selected.



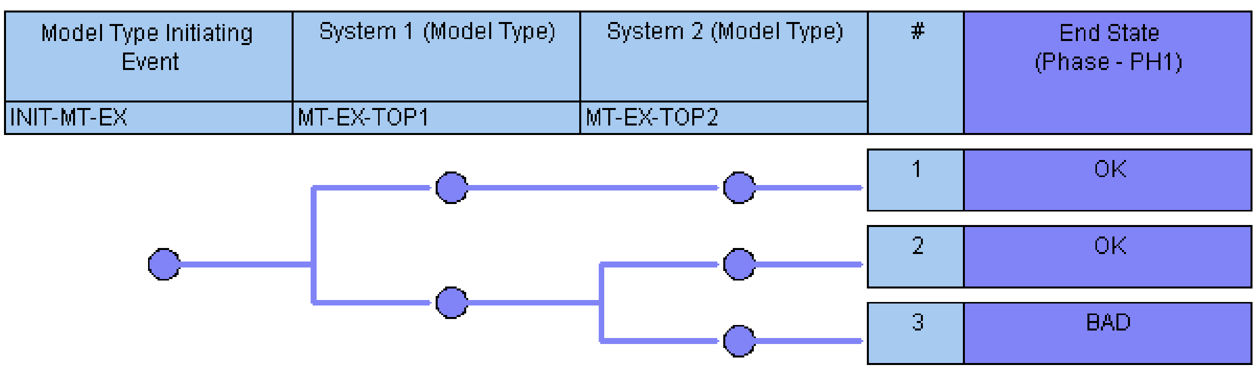
♦ Type in the Model Type and provide this model type with an ID, which will be used when displaying the basic events in cut sets, etc. Then specify a description and color for the model type.



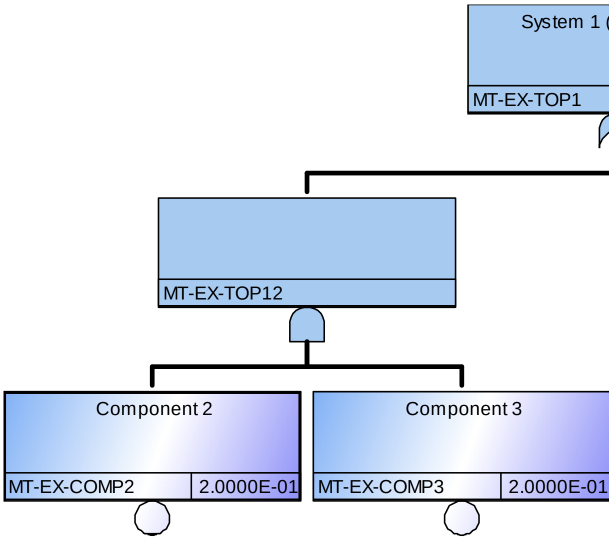
- This newly added model type (EXAMPLE) can now be developed.
- The event tree(s) fault tree(s) are developed as discussed in the SAPHIRE Basics manual.
- The difference in developing a new model type versus the default Random model type is selecting this model type for each specific event. This is how SAPHIRE knows where to segregate the information for specific analyses.

13.1.1. Logic Models

- Event tree
  - The event tree is developed as discussed in the SAPHIRE Basics manual. Start with the initiating event and then identify the systems (tops) needed to respond.
  - The initiating event will be stored in the basic event listing and needs to be modified to specify which susceptibilities (model types) it is available for when performing analyses.

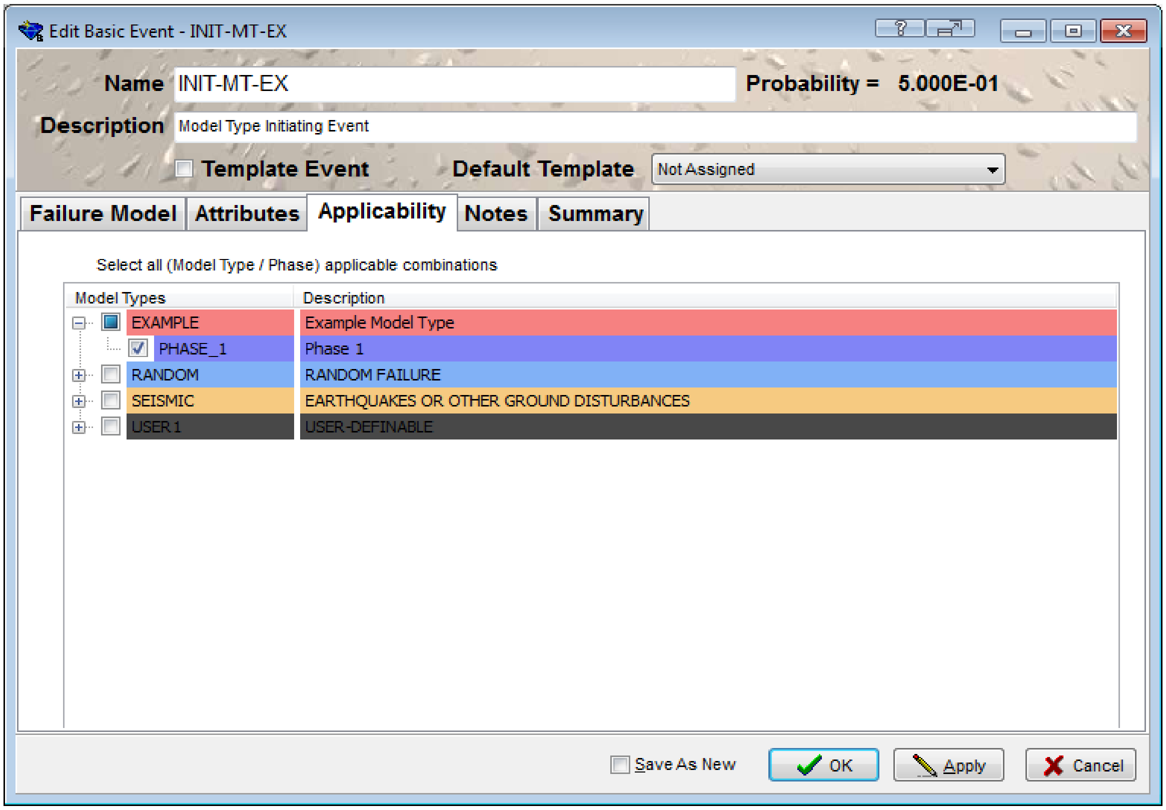


- Fault tree
  - The fault tree is developed as discussed in the SAPHIRE Basics manual.
  - The basic events (components and human errors) will be stored in the basic event listing and need to be modified to specify which susceptibilities (model types) it is available for when performing analyses.
  - The fault tree shown below shows the colors specified for Random model type. The colors will change once the basic event is specified as an EXAMPLE model type.

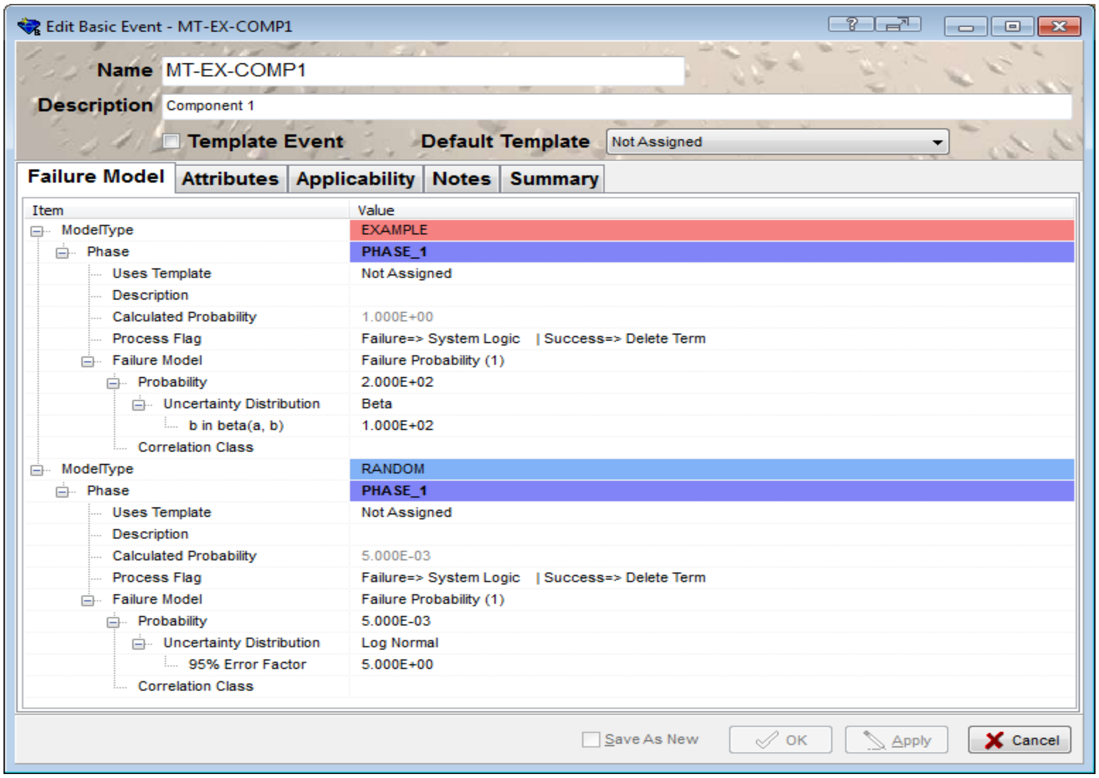


13.1.1. Basic Event Modifications

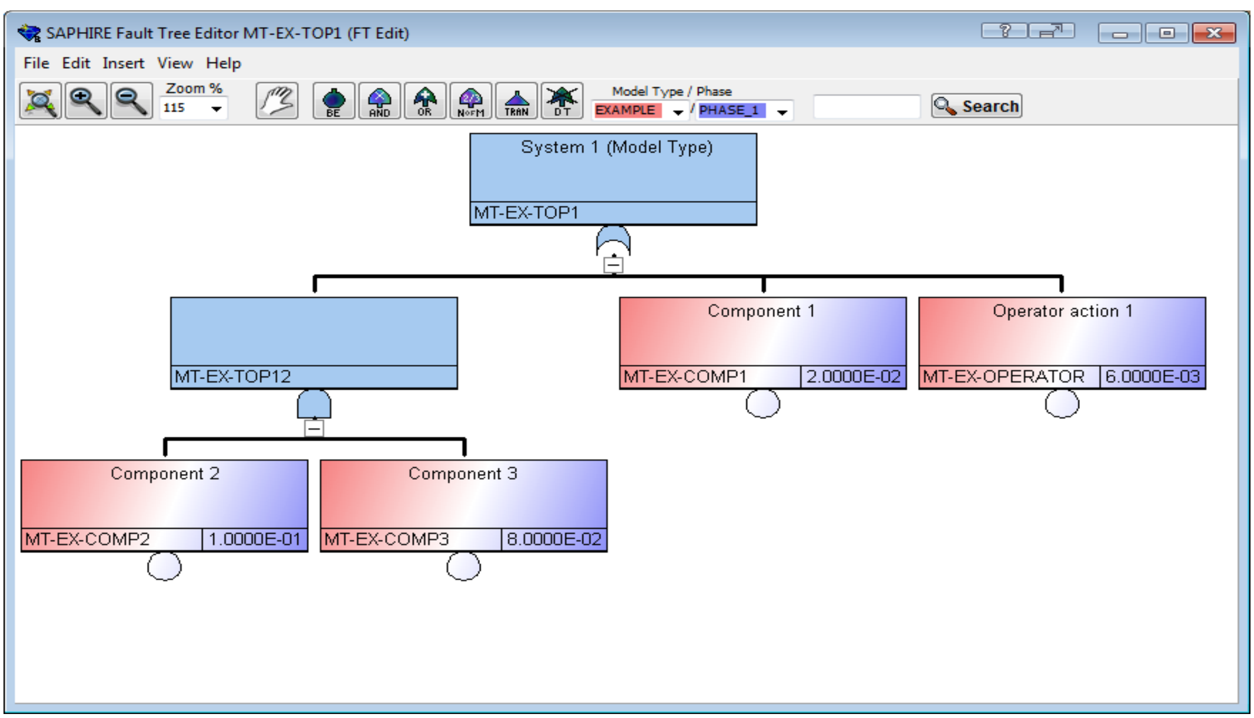
- Initiating event
  - To modify the initiating event, select it in the Basic Event list panel and then double click or highlight the initiating event, right click the mouse and select **Edit Basic Event**.
  - Specify the frequency to be used for this event and uncertainty information and then select the Applicability Tab.
  - Under the **Applicability** tab, check the EXAMPLE model type, this tells SAPHIRE that this event can be evaluated in this model type (to uncheck the Random option, uncheck the phase_1 option and SAPHIRE will correctly update the information).



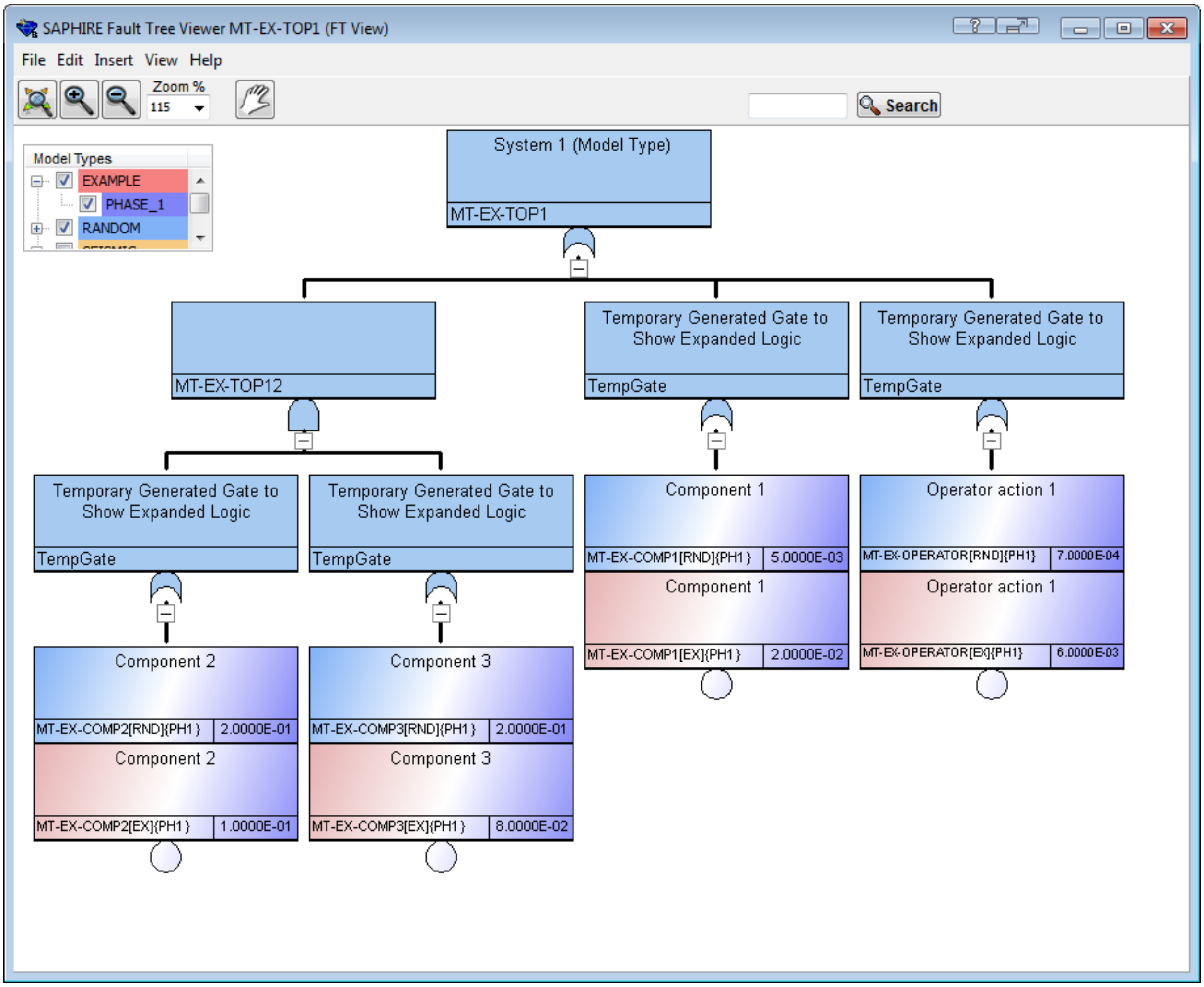
- Basic events
  - To modify the basic events, select the basic event of interest in the Basic Event list panel and then double click or highlight the initiating event, right click the mouse and select **Edit Basic Event**.
  - Select the **Applicability** tab and check EXAMPLE model type. If this basic event (component or human action) is susceptible to the RANDOM model type, leave this model type checked.
  - Now specify the specific failure probability for this basic event. A different probability can be used for the EXAMPLE model type versus the RANDOM model type. This allows for flexibility and specific type of analyses.



- The fault tree graphics can show the different susceptibilities of the basic events
    - ◊ Under this view option, only one Model Type can be displayed. To change the display from the default of Random, the fault tree must be solved in that Model Type.



◊ or all the different Model Types can be viewed together by selecting **File → View Expanded Model Types**

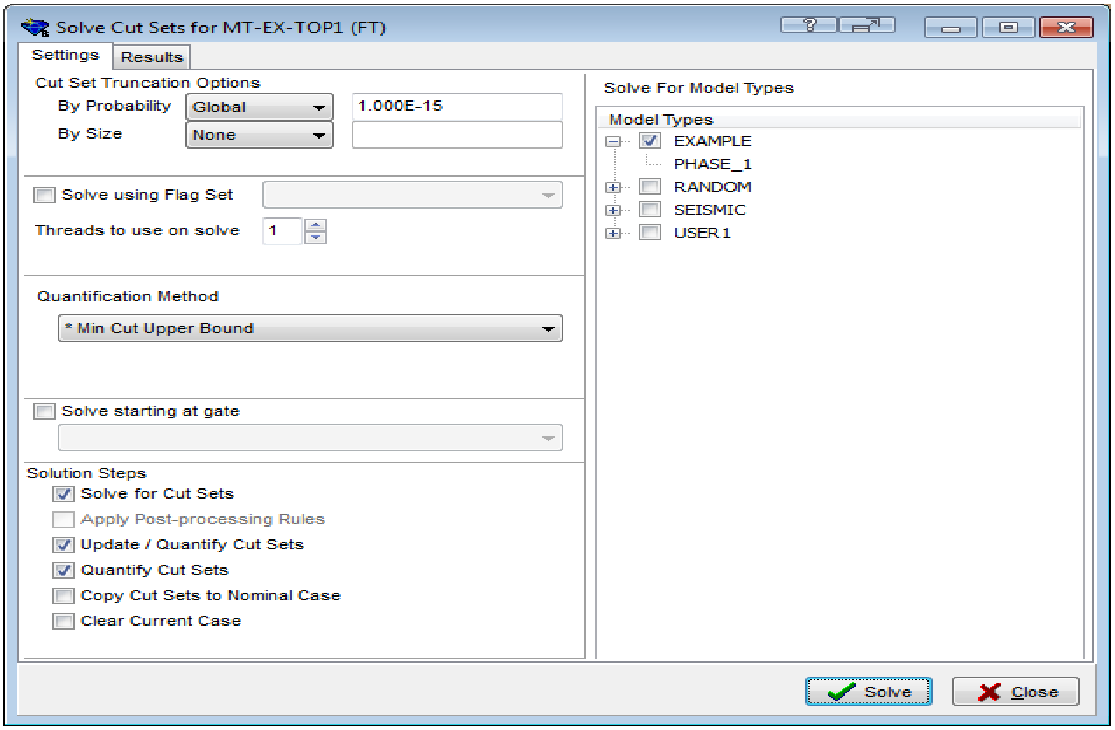


13.2. Solve Logic Models Using New Model Type

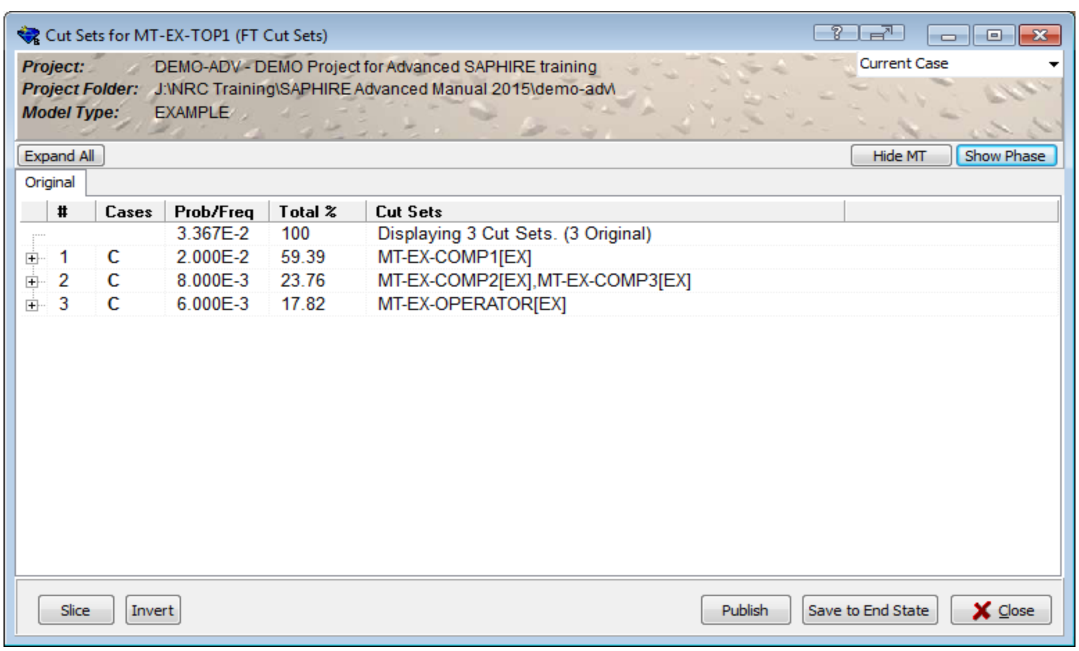
13.2.1. Fault Trees

- Highlight the Fault Tree to be analyzed, then right click and select **Solve**.
- To solve the Fault Tree and have the results stored in the new Model Type (EXAMPLE) in the **Solve for Model Types** column check the EXAMPLE box and de-check the RANDOM box.

- The truncation options work the same, except now the results will be stored in the new Model Type field within SAPHIRE.

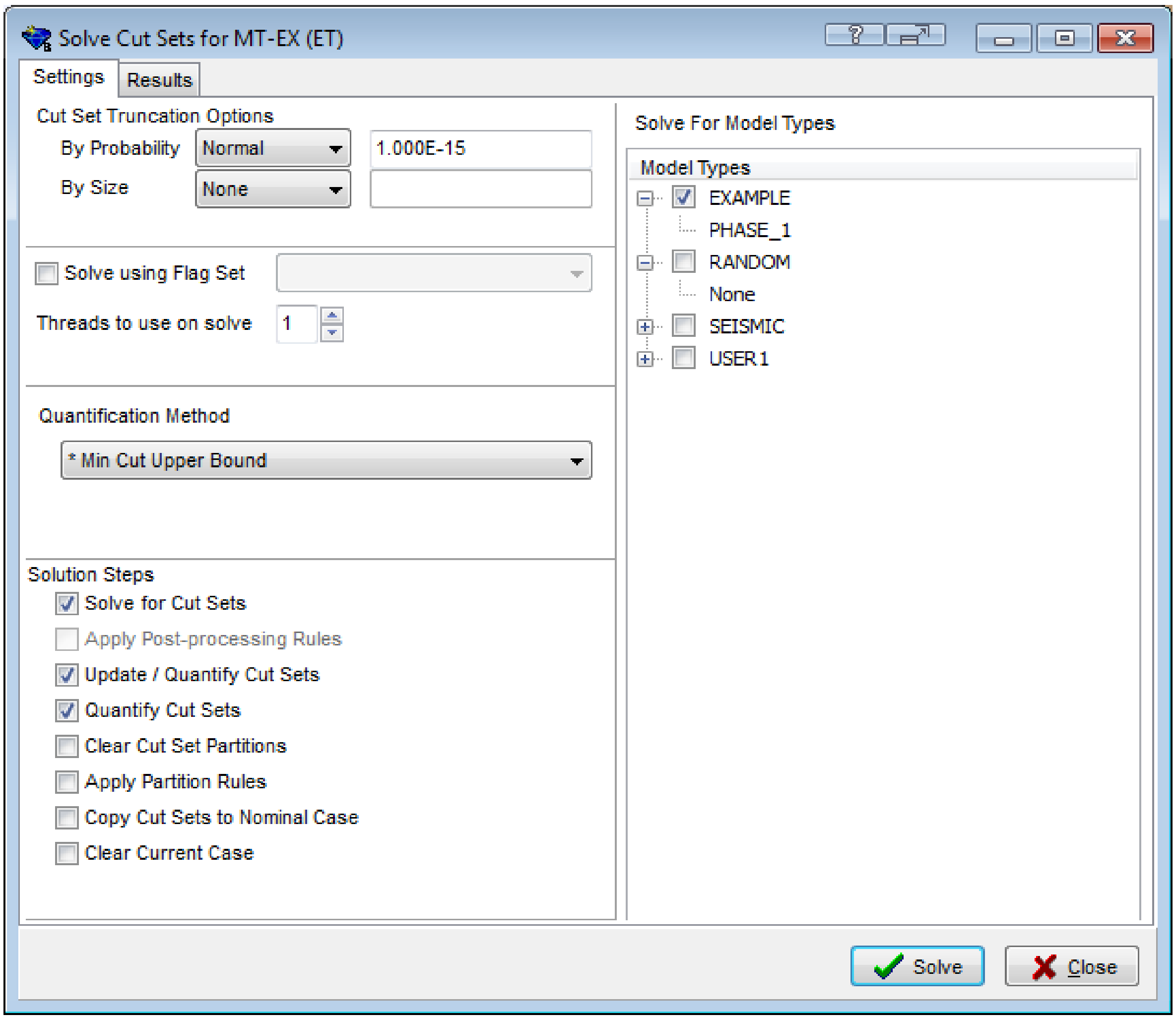


- The cut sets can be viewed by selecting the Cut Sets button on bottom of the Results tab or with the Fault Tree highlighted right click and select View Cut Sets.



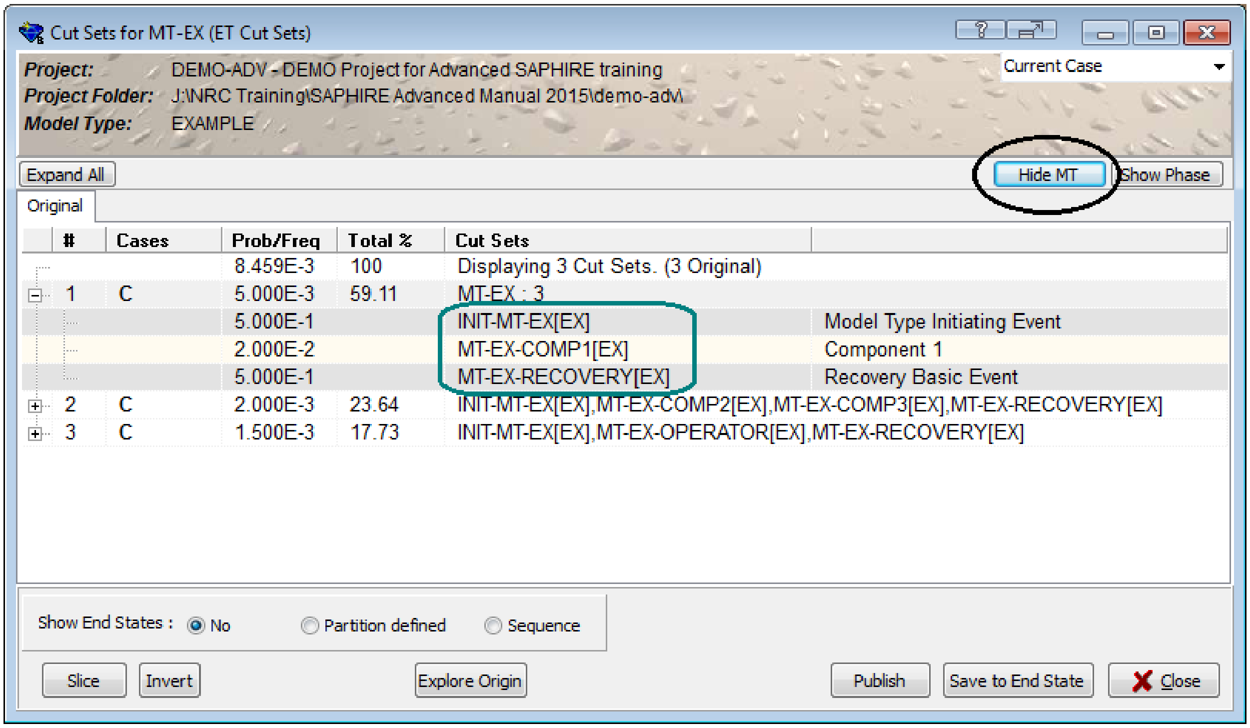
13.2.2. Event Trees

- Highlight the Event Tree to be analyzed, then right click and select **Solve**.
- To solve the Event Tree and have the results stored in the new Model Type (EXAMPLE), check the EXAMPLE box in the **Solve for Model Types** column and de-check the RANDOM box.



- The truncation options work the same, except now the results will be stored in the new Model Type field within SAPHIRE.
- The cut sets can be viewed by selecting the Cut Sets button on bottom of the Results tab or with the Event Tree highlighted, right click and selct View Cut Sets.

- To show the Model Type, the Show MT button in the upper right hand corner needs to be selected. SAPHIRE will then add [EX] at the end of each basic event that is applicable to the Example Model Type. (SAPHIRE does not show the model types unless this option is selected.)



13.3. FIRE and FLOOD Model Types

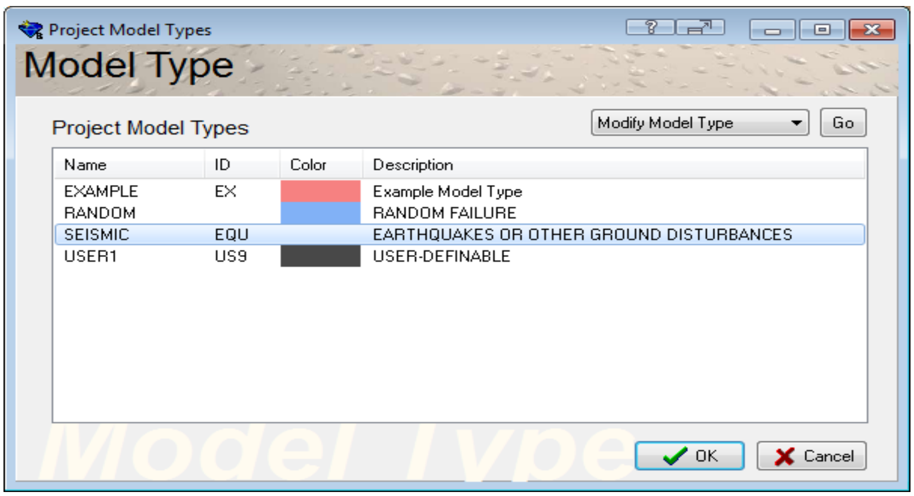
- The Fire and Flood Model Types are very similar to the EXAMPLE Model Type just discussed.
- The Model Type is selected and the logic structures (event trees and fault trees) are developed with specific information pertaining to these analyses types.
- The initiating event and basic events (components and operator actions) are modeled the same as above except now the FIRE or FLOOD model type needs to be selected along with these components failing randomly.
- There are no special calculations that must be done at this time; therefore, these Model Types (analyses) can be handled and evaluated similarly to the EXAMPLE defined Model Type.

13.4. SEISMIC Model Type

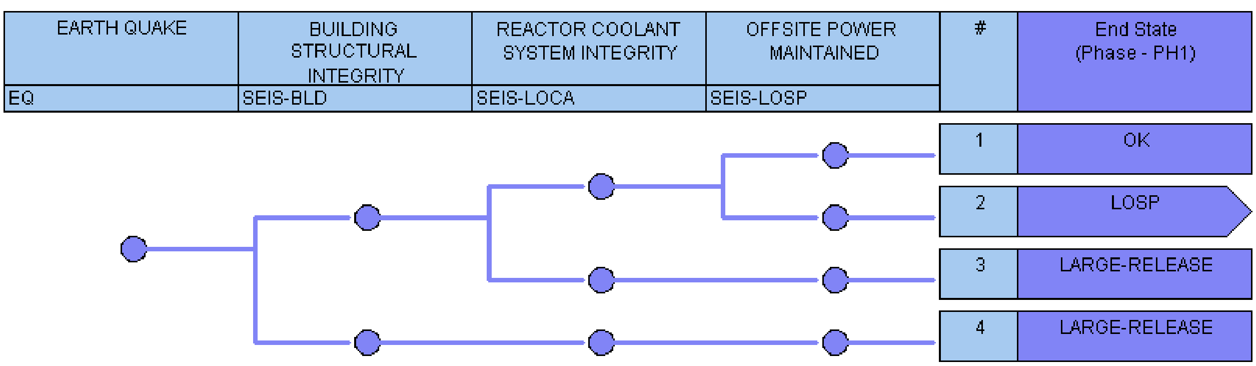
- The Seismic Model Type is a specific Model Type, which utilize built-in equations to calculate the probability a component fails given a certain magnitude of earthquake or greater. Therefore, SAPHIRE has specific calculation types for this analysis.

- To set up the Seismic Model Type for analyses

  - ◊ Add the new Model type as discussed above.

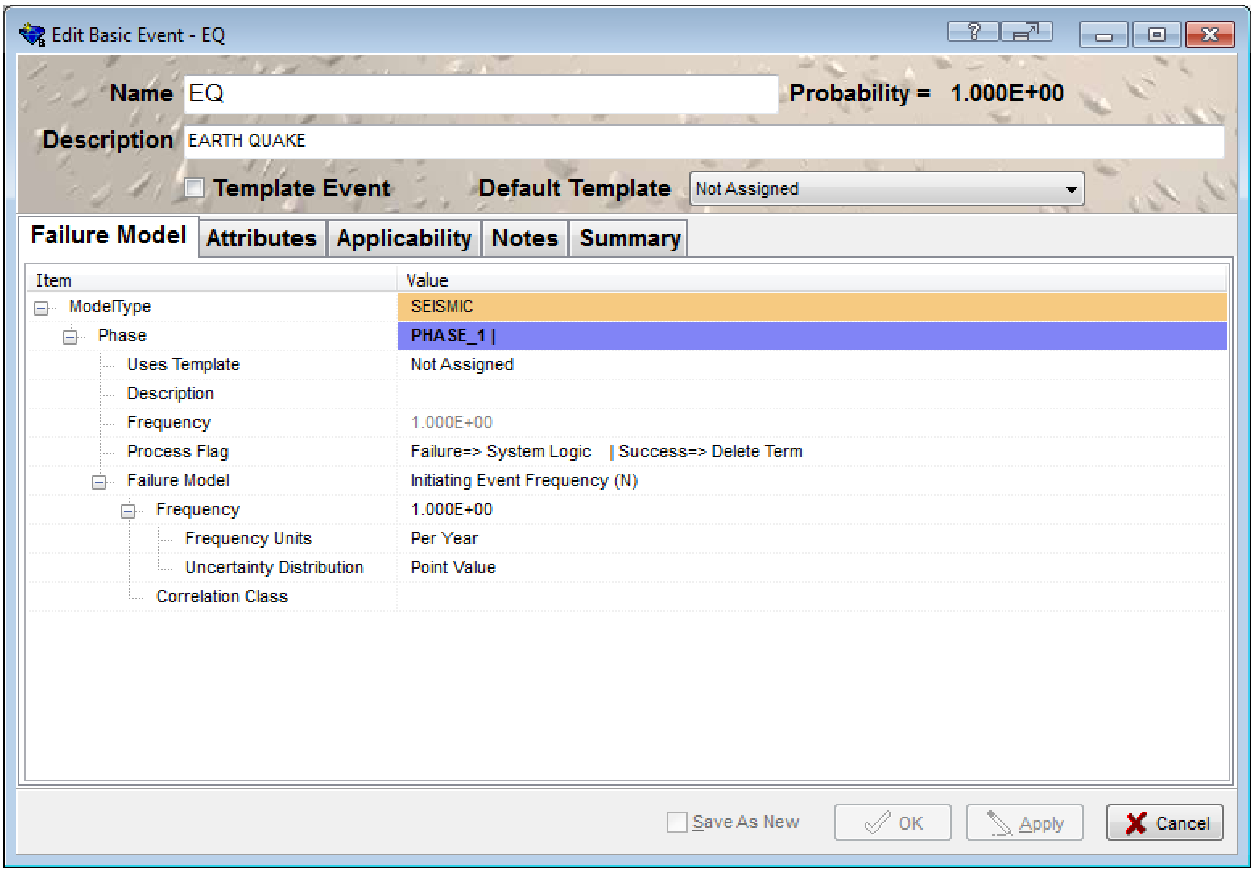


  - ◊ Develop the logic structure (event trees and fault trees) as needed based on plants response to a seismic event.



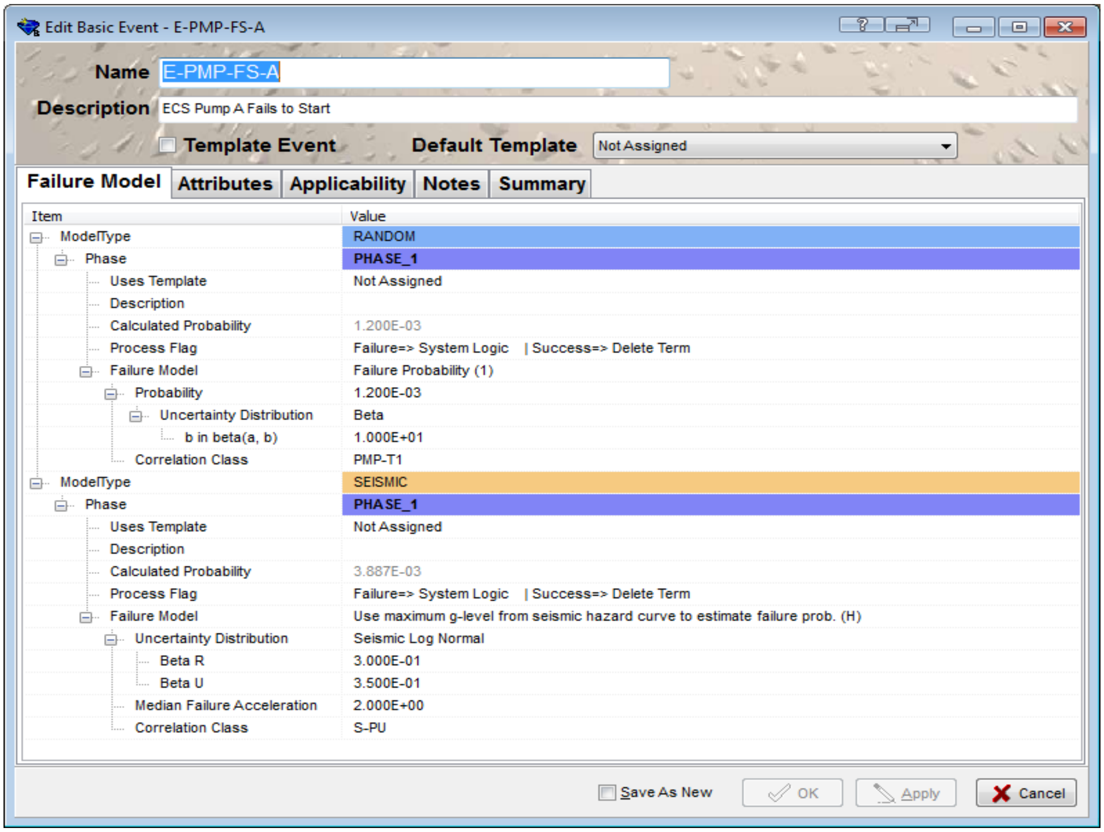
♦ The seismic analysis can be done by setting up the components as a single ground motion or develop a histogram of ground motions that allows for the evaluation to account for all potential ground motions (frequency of occurrence) at a particular plant.

◊ The initiating event needs to be identified as a seismic event in the **Applicability** tab and the selection of either a single ground motion event or a histogram. If a single ground motion is to be evaluated, then just specify the exceedance frequency of that specific ground motion. Otherwise, leave the frequency as 1.0 and SAPHIRE will use the developed histogram.

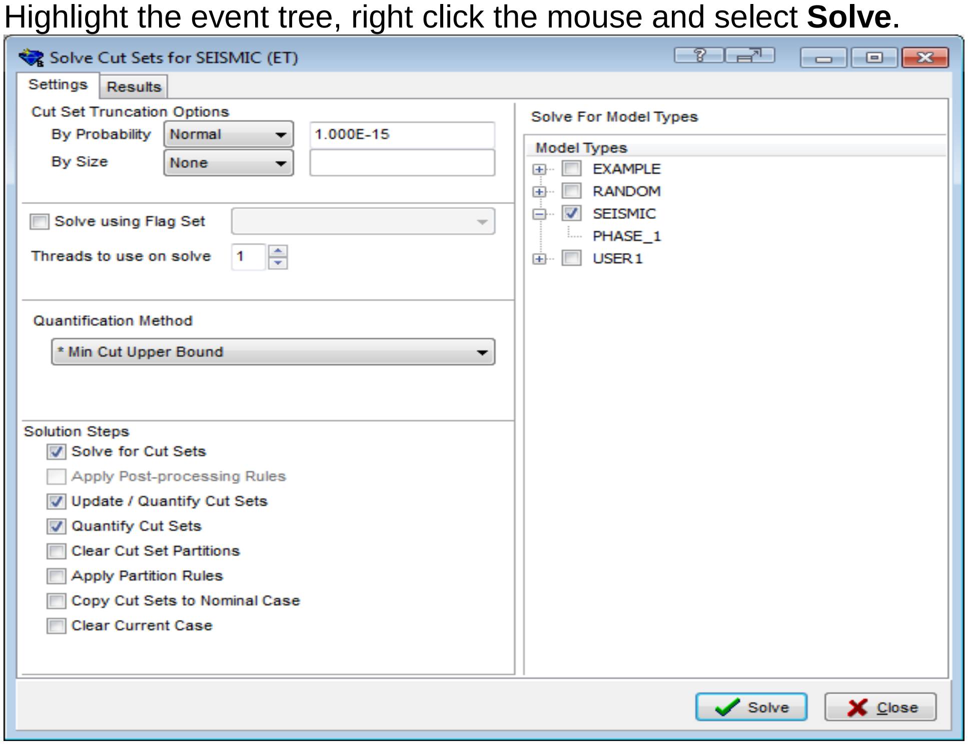


◊ The seismic susceptible basic events need to be modified to include all of their appropriate information.

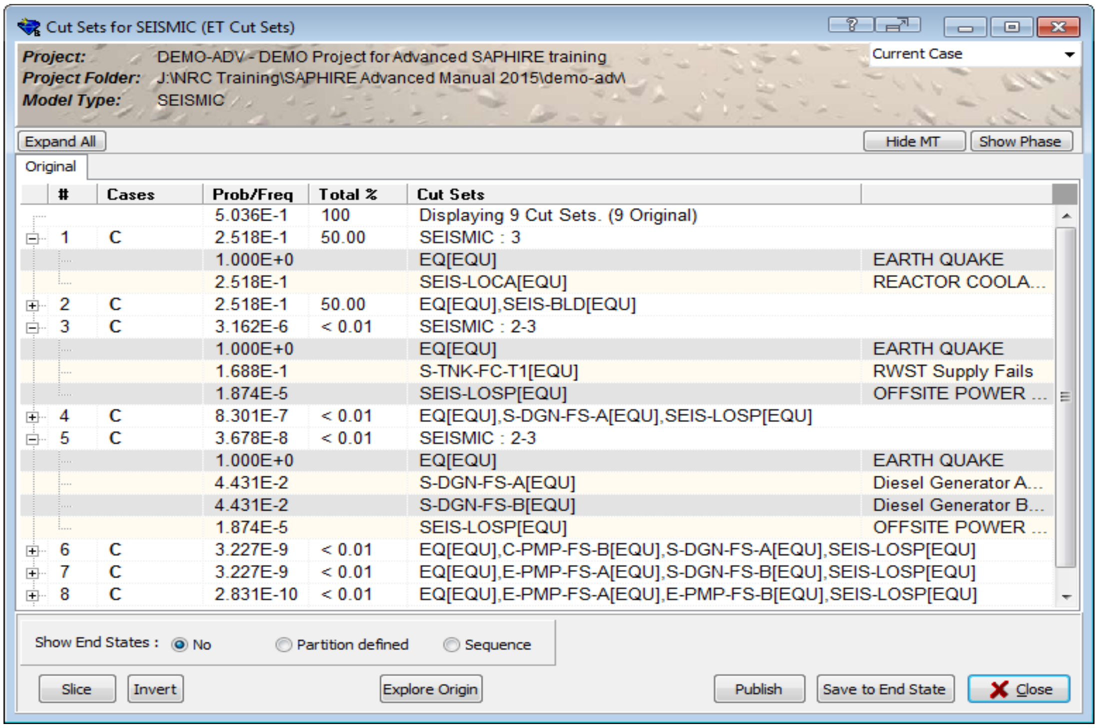
◊ If the evaluation is only based on a single ground motion, then the basic event's calculation type is set to "G - User defined seismic g-level". Otherwise, select the "H - use maximum g-level from histogram".

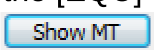


- ♦ Once all of the basic events are designated with the appropriate calculation type, the fault trees and/or event trees can be evaluated.
- ♦ Highlight the event tree, right click the mouse and select **Solve**.



- The results from solving the SEISMIC event tree (using the histogram data input) are listed below.



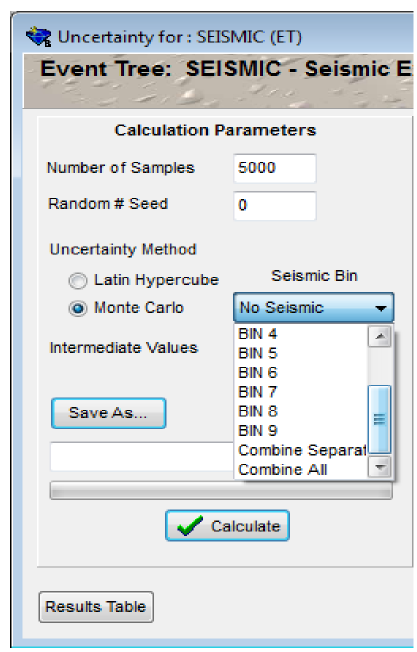
- To get the [EQU] to show up, click the Show MT button .

- The cut sets above are based on using the hazard curve input into SAPHIRE as a histogram. Therefore, the initiating event frequency is 1.0 and the basic events' probabilities are based on the highest g-level ground motion.

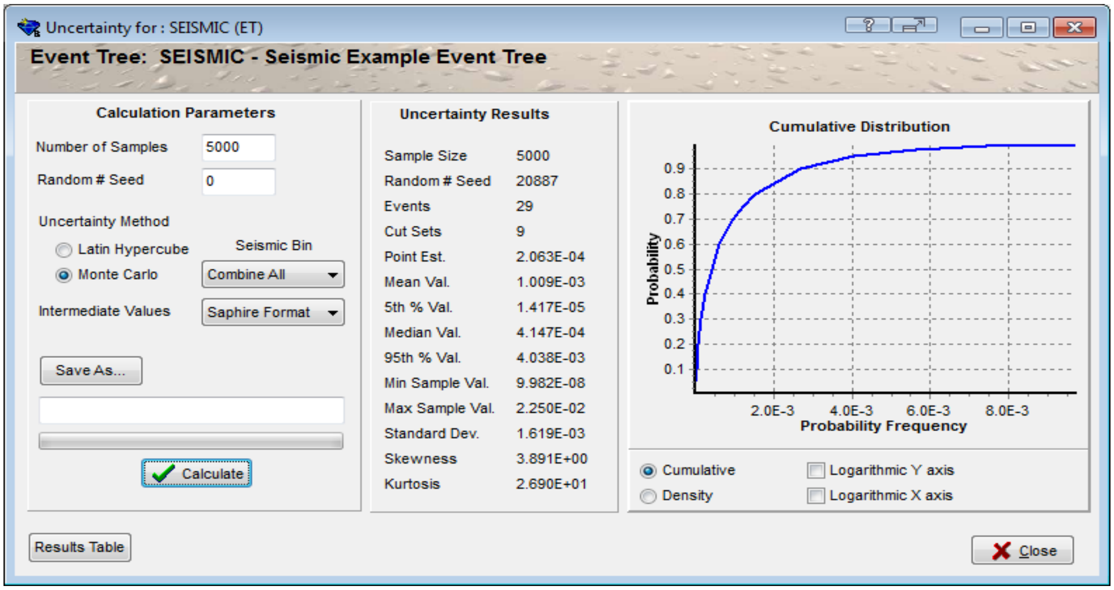
- To get result, an uncertainty analysis is required.

  ◊ To perform a seismic uncertainty analysis, highlight the event tree, right mouse click and select **View Uncertainty** → (Group by Event Trees or Single Sequence) [depending upon final result needed].

  ◊ Input the necessary parameters, # of Samples, Random Seed #, Monte Carlo or Latin Hypercube and then which seismic bin or all combined to get the overall result.



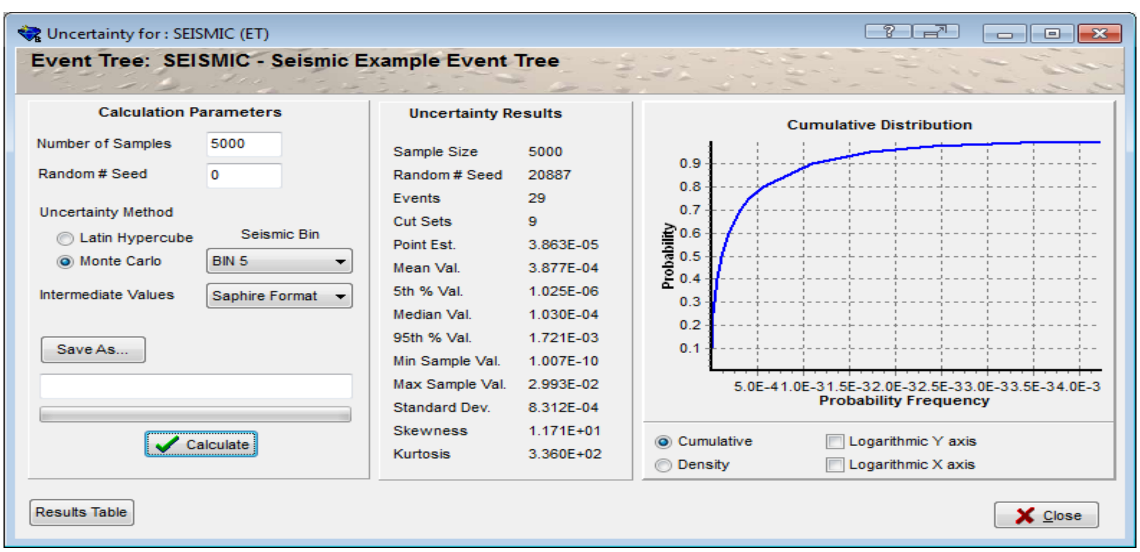
◊ Click the Calculate button and SAPHIRE will perform the sampling from the hazard curve and basic events to give an overall result.



♦ To analyze a single ground motion, this can be achieved two different ways:

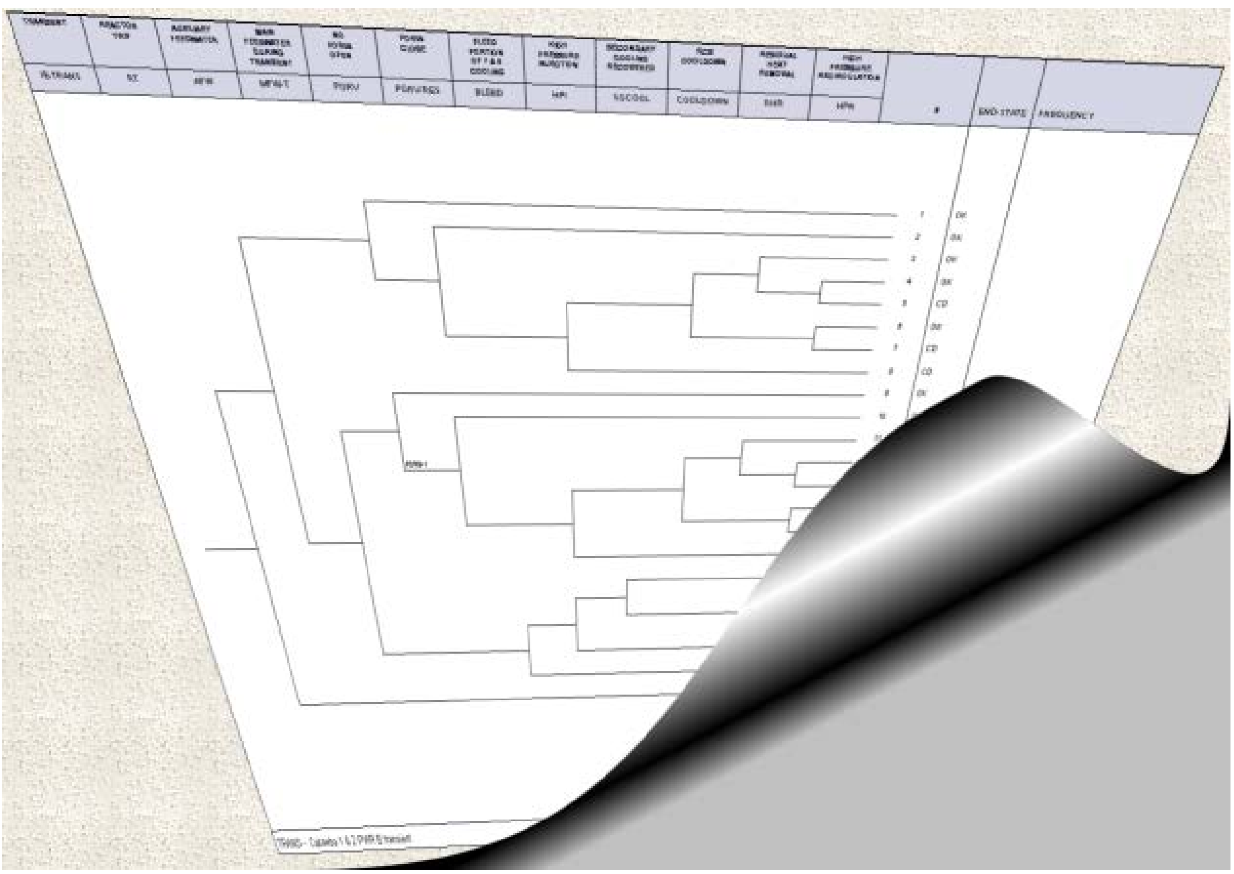
◊ The exceedance frequency needs to be manually put into the initiating event at that specific ground motion and all of the basic events failure model needs to be set to a G: ground motion calculation type and the specific g-level ground motion is specified. Then solve the event tree and the results will reflect that information.

◊ The uncertainty analysis allows for specific bins to be analyzed. Select the bin that pertains to a specific ground motion and perform the uncertainty analysis.



| 14 | THE LARGE EVENT TREE METHODOLOGY

Section 15 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.



14.1. Large Event Tree Methodology Introduction

There are two basic approaches for accident sequence quantification:

1. Fault-tree linking (covered in the SAPHIRE Basics, demonstrated in the DEMO project, and described in previous sections).
2. 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.

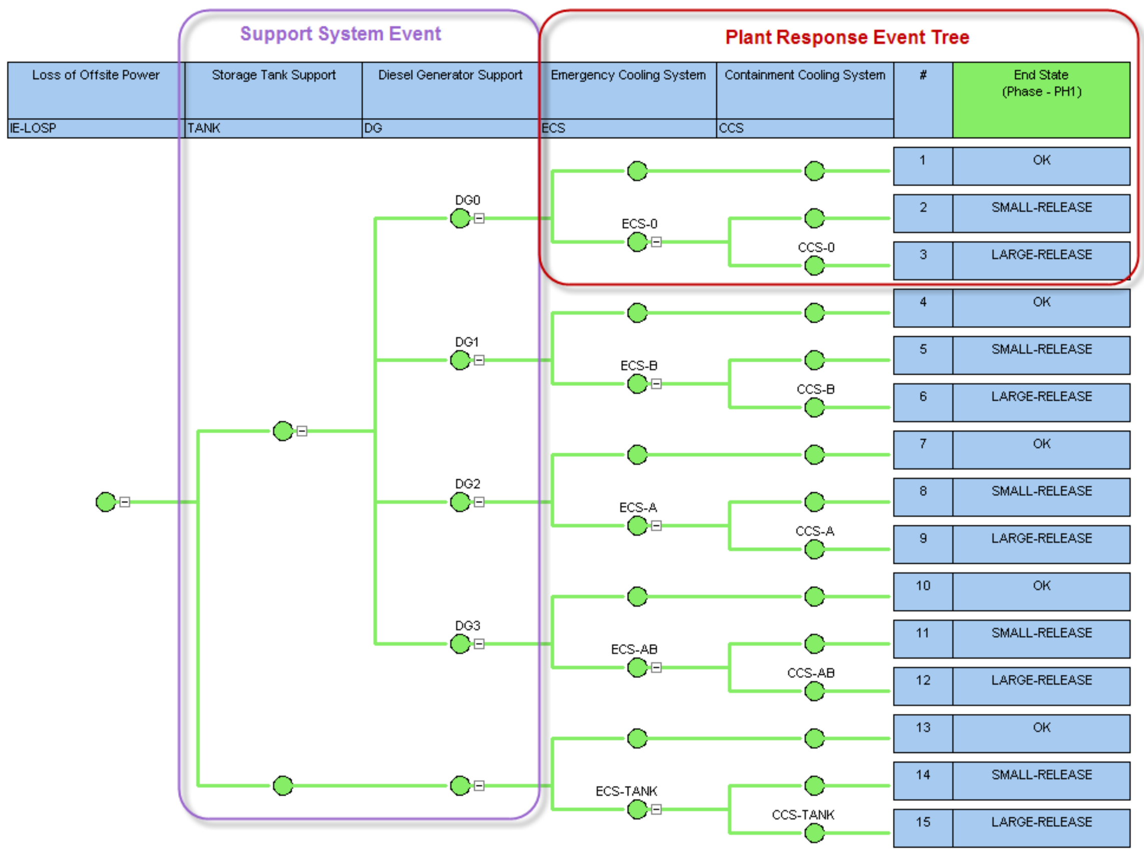
Important features of the large event tree approach (with regard to model construction and use) are:

- ♦ 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 14.4.)
- ♦ The use of multiple-split branching in the event tree, such as 3-split or 4-split branching. (Described in Sections 4 and 14.4.)
- ♦ The use of truncation when linking event trees because of the large number of sequences that could be generated. (Described in Section 14.5.)

14.2. 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:

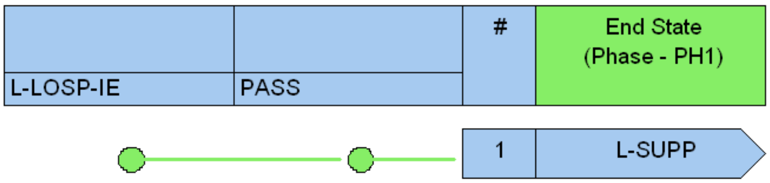
    - ◊ Create an event tree that contains the initiating event and transfers to the appropriate support system event tree. (Note: SAPHIRE requires that at least 2 tops are present in each event tree; however, there does not need to be any branching.)

    - ◊ 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.

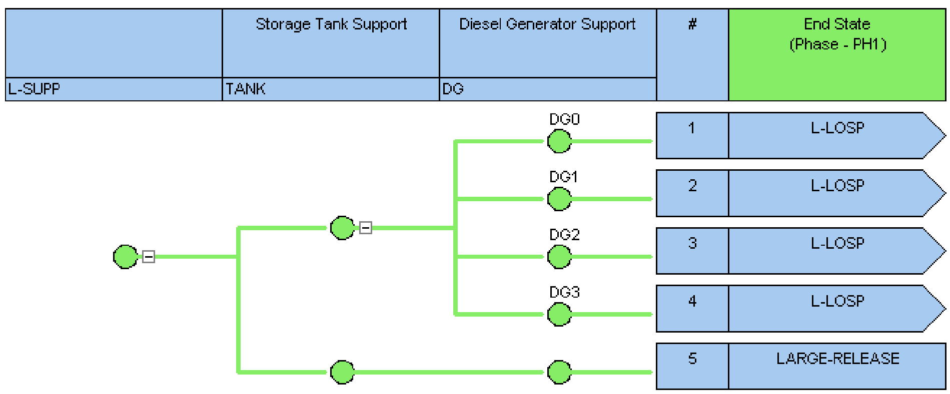
        - Rules can be written in the "Link Event Tree" rule editor to enact this transfer (see Section 4).

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

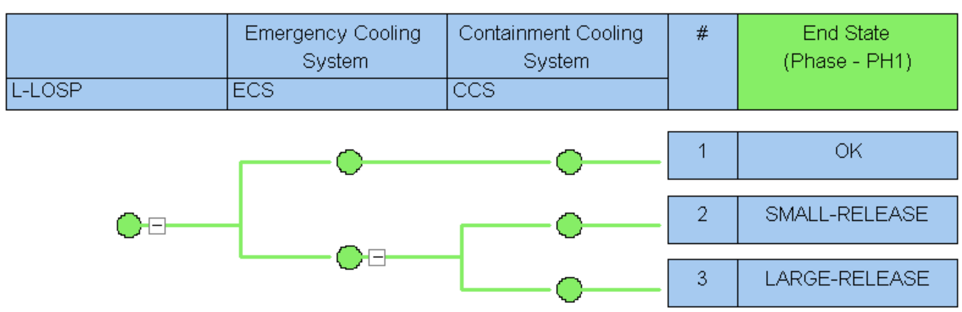
The "Initiating Event" Tree



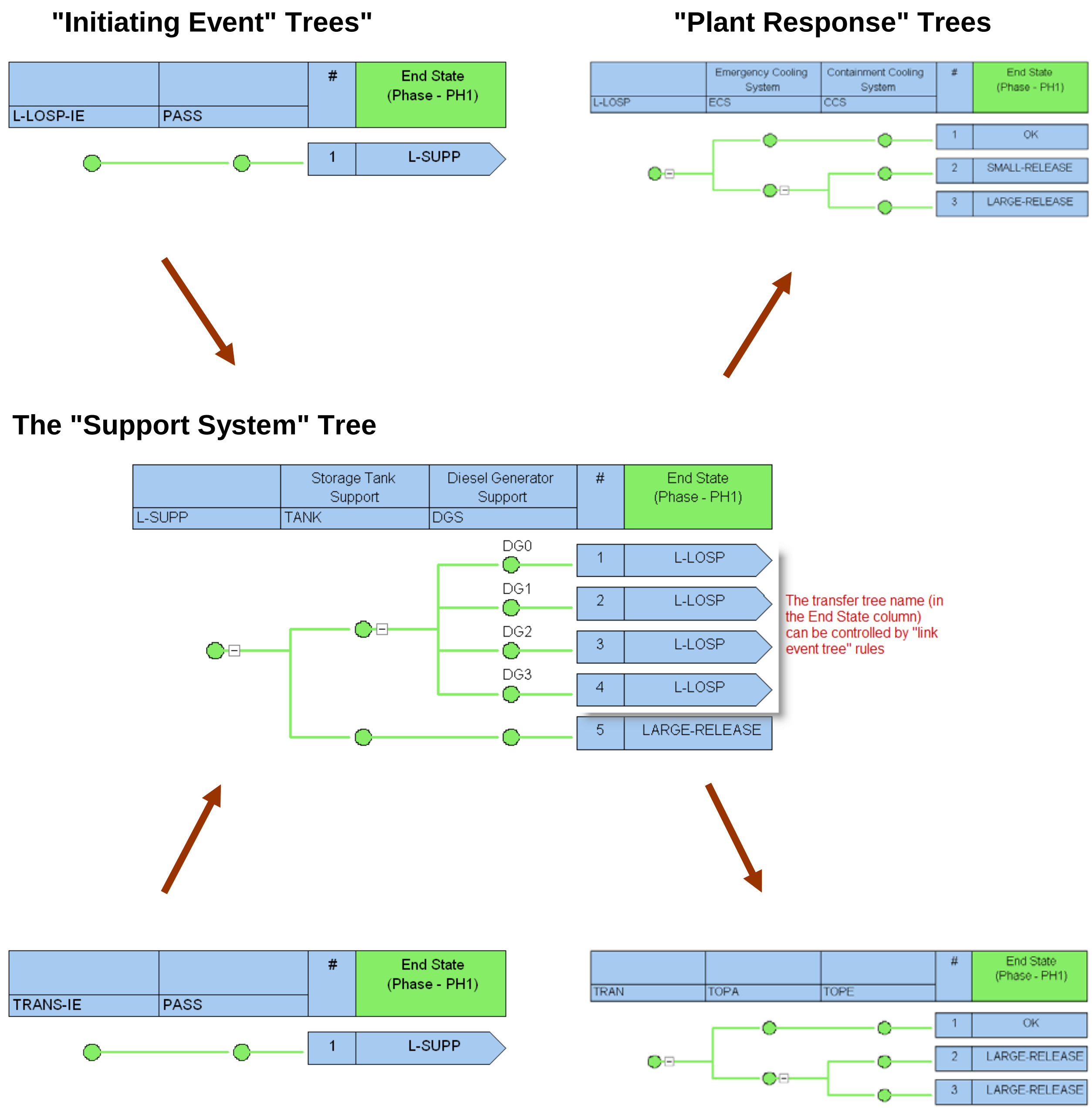
The "Support System" Tree



The "Plant Response" Tree



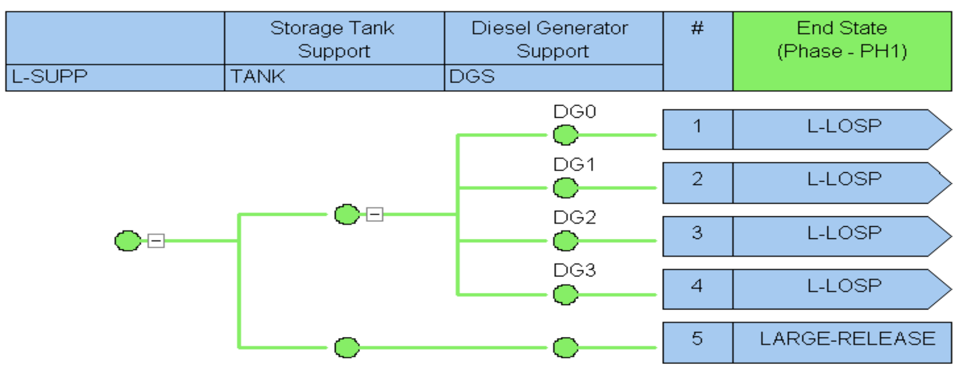
Notice that more than one initiating event can call the same support system tree. And the support system tree could transfer to different plant response trees.



14.2.1. 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 cannot 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 that 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).

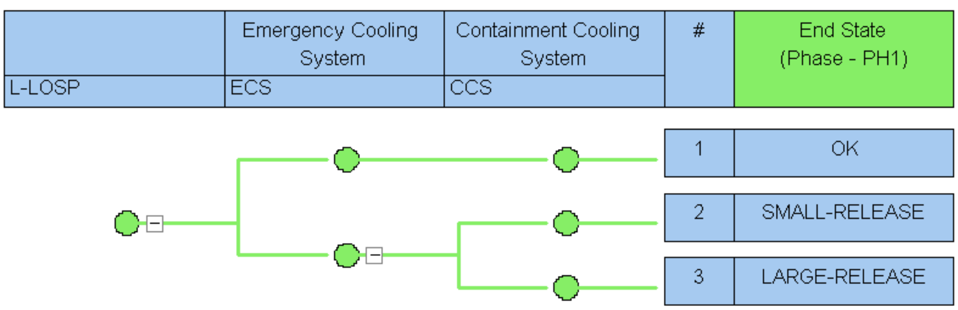
◊ The fourth and final branch represents the failure of both DG-A and DG-B (top event assignment is DG3).

♦ (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.

14.2.2. 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).

14.3. 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 and TANK. These new fault trees are solved and then assigned the “S” calculation type.)

14.3.1. Assign the split-fraction probabilities directly

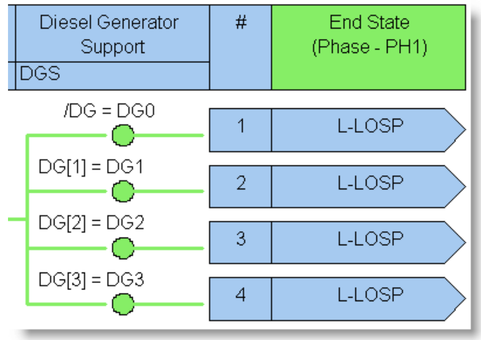
- ♦ To assign the split-fraction probability directly, the probability is entered via double clicking on the basic event in the Basic Event List Pane and using the **Basic Events** list panel (because SAPHIRE recognizes top events as developed events).

- ♦ 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.

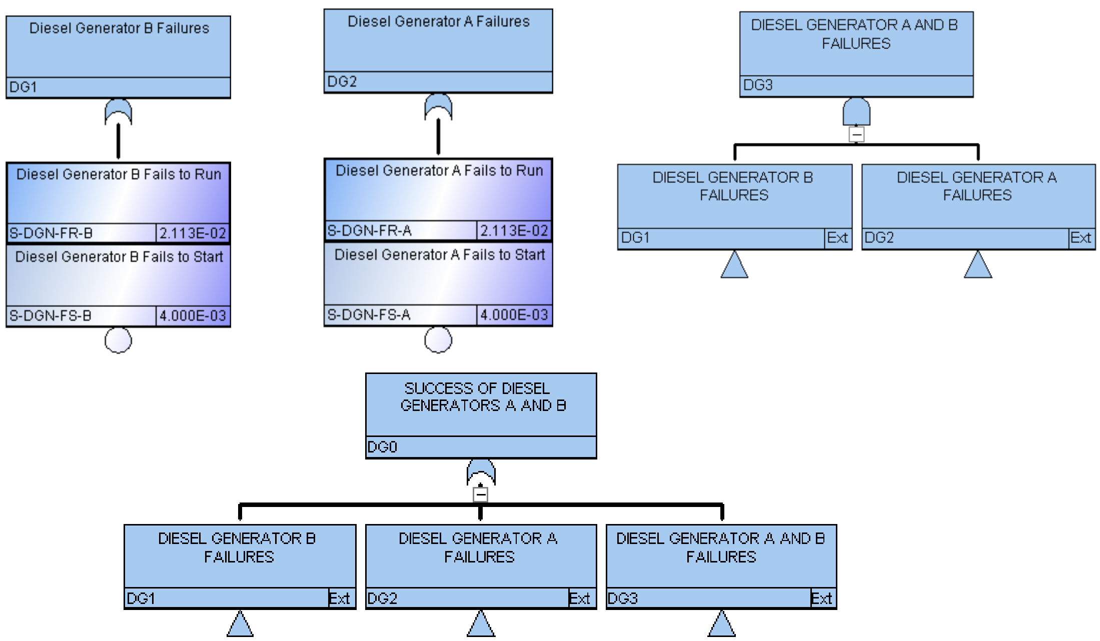
- ♦ The split-fraction probability for each DG branching is calculated using the following fault trees.

    - ◊ 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 DGS top event in detail. The top branch (/DGS) 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.

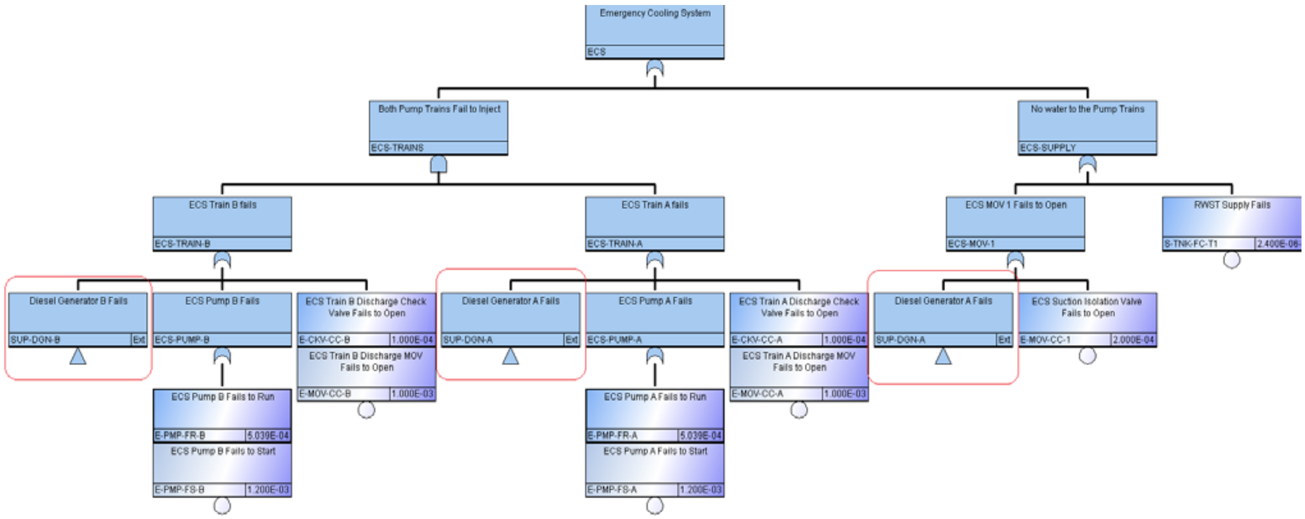


- The developed fault trees representing the different states of the two diesel generators can now be solved via highlighting the **Fault Tree**, invoking the menu through right clicking, and then using the **Solve** option with no truncation.

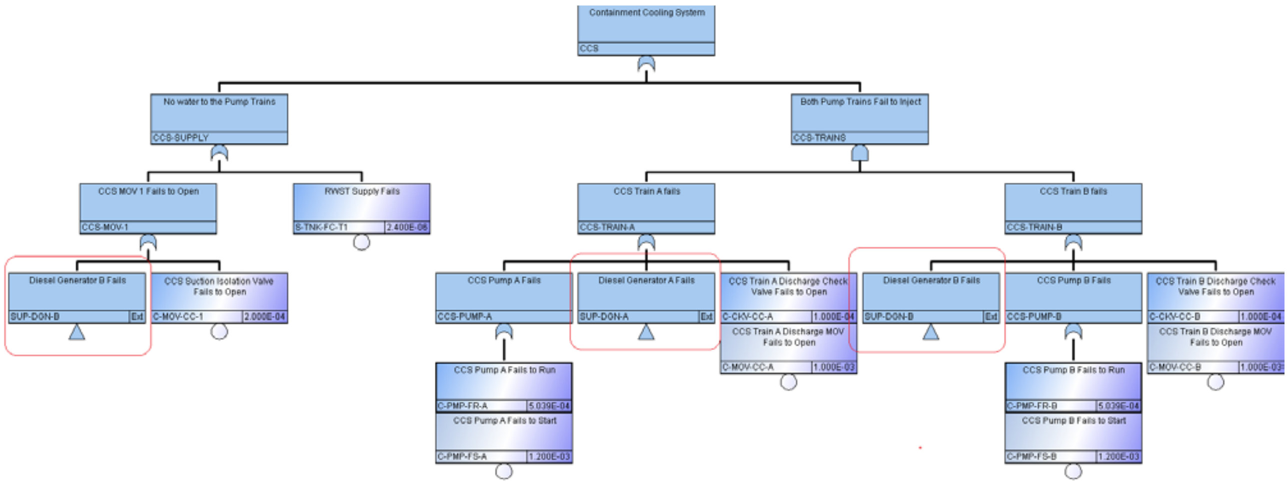
- The probability calculated from each fault tree can now be assigned to the fault tree top events in the **Basic Events** list panel.

- 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).

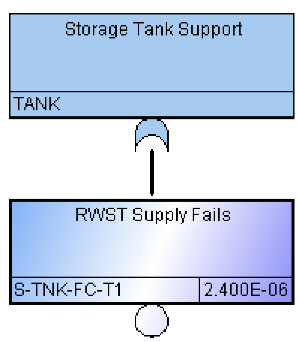


DG Status	***ECS System Name***	***Split-Fraction***
No DGs failed	ECS-0	2.079E-4
Only DG-A failed	ECS-A	1.0
Only DG-B failed	ECS-B	3.001E-3
Both DGs failed	ECS-AB	1.0



DG Status	***CCS System Name***	***Split-Fraction***
No DGs failed	CCS-0	2.079E-4
Only DG-A failed	CCS-A	3.001E-3
Only DG-B failed	CCS-B	1.0
Both DGs failed	CCS-AB	1.0

♦ A simple fault tree to represent the TANK top event needs to be created in order to determine its split fraction probability.



14.3.2. Assign probabilities using "S" calculation

♦ To assign the top event probabilities using the "S" calculation, the following steps are required.

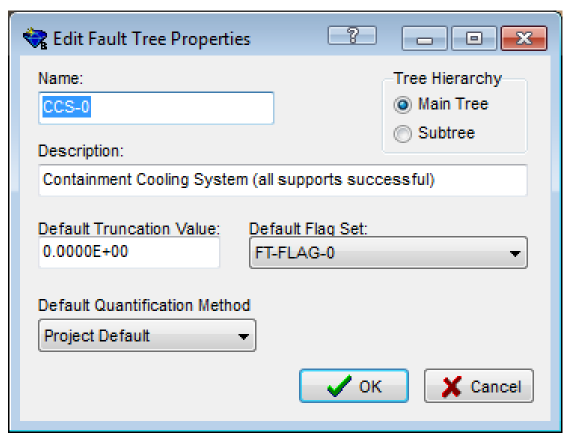
1. Multiple copies of the ECS and CCS fault trees need to be created with the names listed above (i.e., ECS-0, ECS-A, ECS-B, and ECS-AB). This can be accomplished by opening the root fault tree (ECS for instance) and using the fault tree main menu **File → Save As** option.

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	S-DGN-FS-A	FALSE
	S-DGN-FR-A	FALSE
	S-DGN-FS-B	FALSE
	S-DGN-FR-B	FALSE
	S-TNK-FC-T1	FALSE
FT-FLAG-A	S-DGN-FS-A	TRUE
	S-DGN-FR-A	TRUE
	S-DGN-FS-B	FALSE
	S-DGN-FR-B	FALSE
	S-TNK-FC-T1	FALSE
FT-FLAG-B	S-DGN-FS-A	FALSE
	S-DGN-FR-A	FALSE
	S-DGN-FS-B	TRUE
	S-DGN-FR-B	TRUE
	S-TNK-FC-T1	FALSE
FT-FLAG-AB	S-DGN-FS-A	TRUE
	S-DGN-FR-A	TRUE
	S-DGN-FS-B	TRUE
	S-DGN-FR-B	TRUE
	S-TNK-FC-T1	FALSE

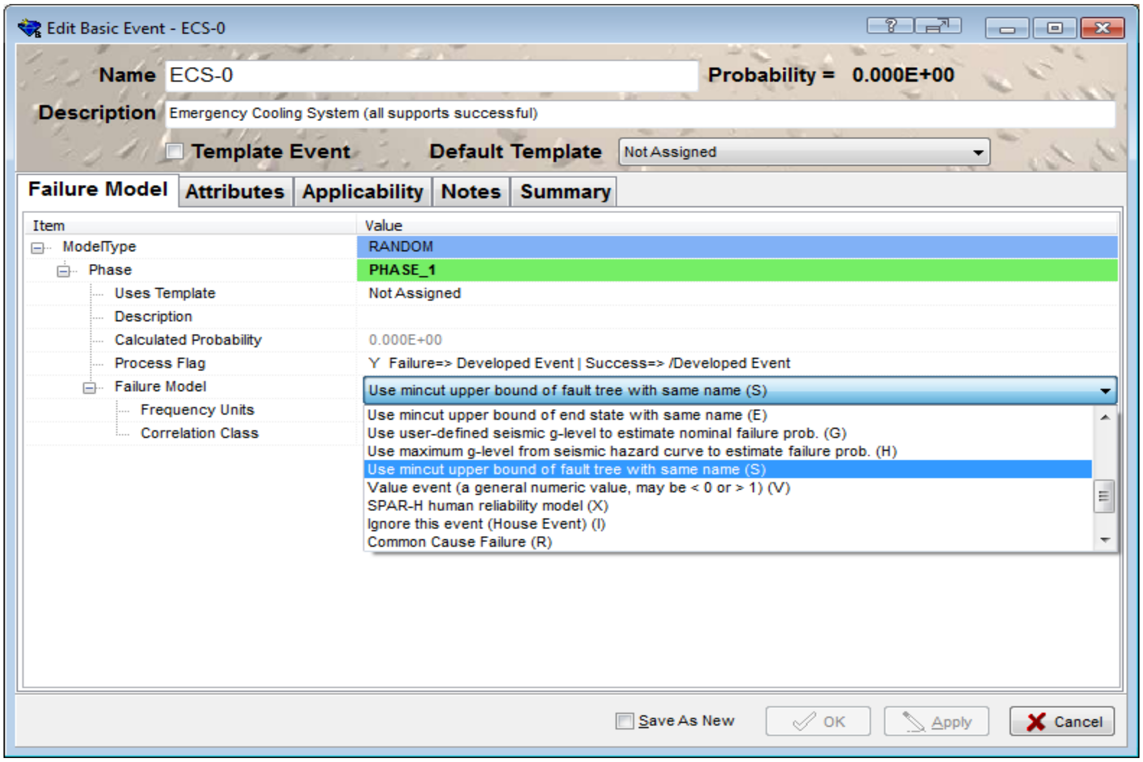
3. Assign the fault tree flag sets to the appropriate fault tree for the calculation process. This is performed using the open fault tree's main menu **Edit → Properties** option. Open each fault tree individually, click **Edit → Properties** to open up the editor, and then select the **Default Flag Set** to 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 the **Edit 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 (highlight all fault trees, right click, then **Solve**) with no truncation.

♦ 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.

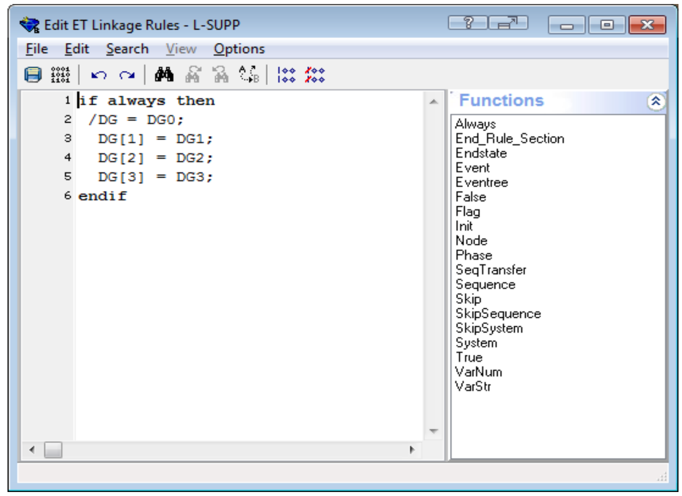
14.4. Using "Link Event Tree" Rules to Assign Split-Fractions

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.

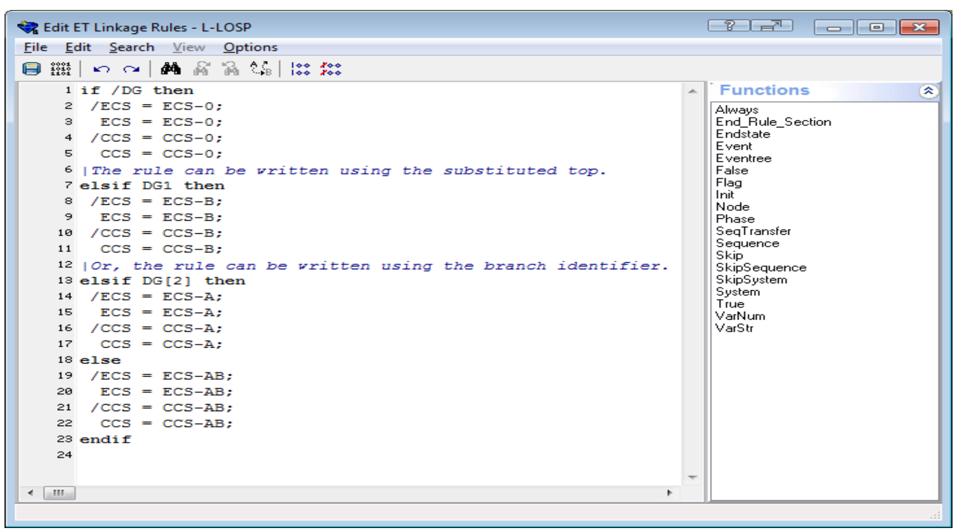
14.4.1. Assigning the Support System Split-Fractions

- 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.



14.4.2. 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.

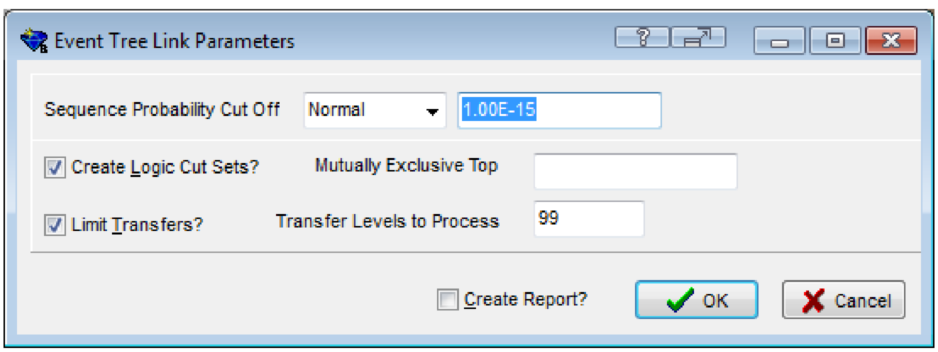


14.5. Truncating Sequences during Event Tree Linking

- 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 14.3).

14.5.1. Generating the Sequence "Cut Sets" During Event Tree Linking

- The L-LOSPIE event tree (from Section 14.2) 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**.
- 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** option, we need to specify two options.
    1. Select either Normal or Conditional truncation from the pull-down menu 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.

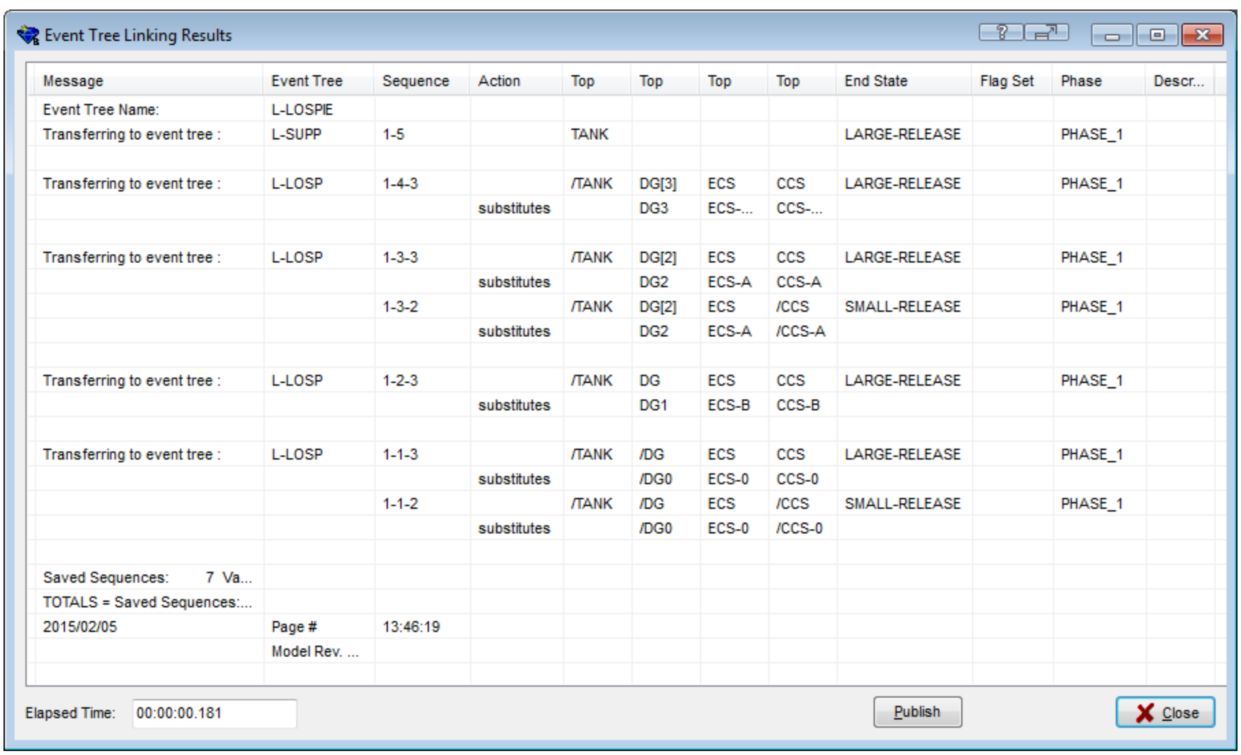
- The **Conditional** option will truncate the sequence once its value is below the “cut off” value specified.

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:



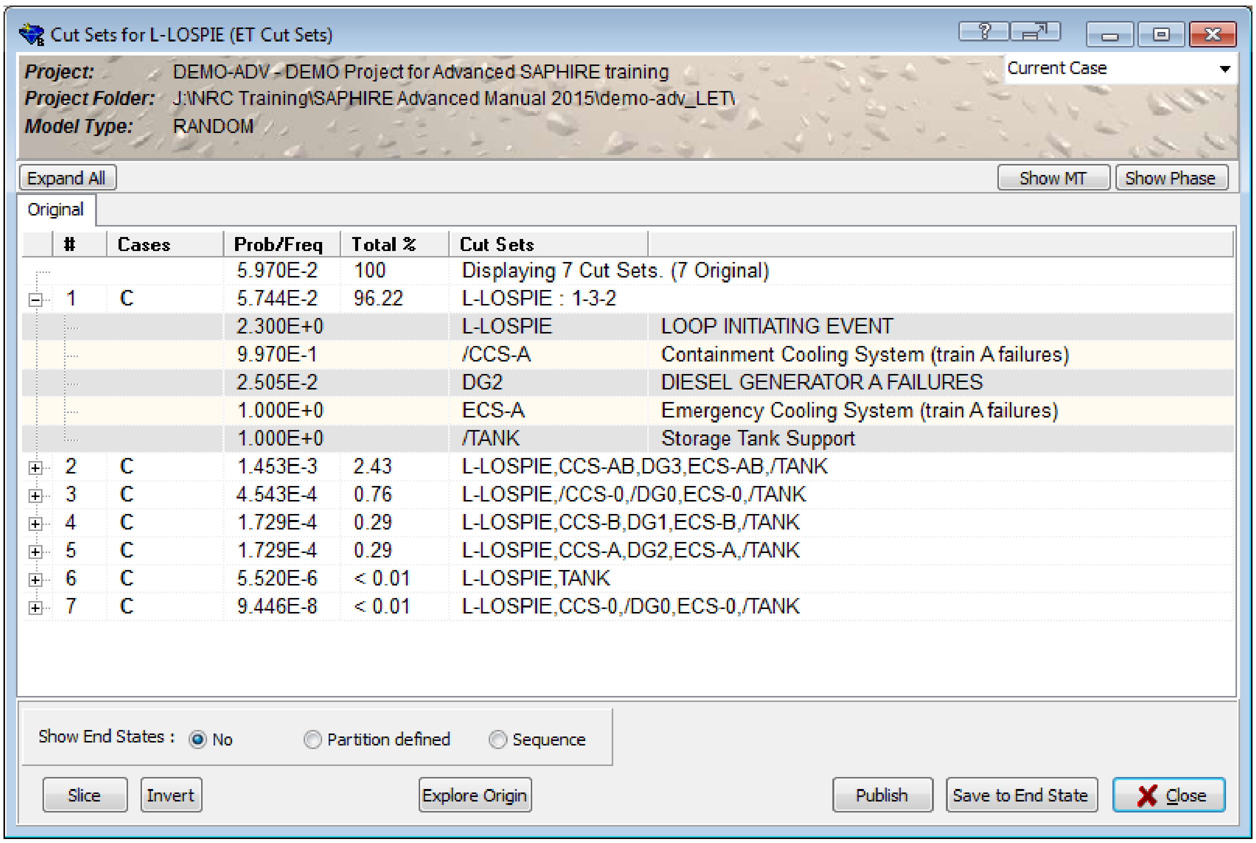
Message	Event Tree	Sequence	Action	Top	Top	Top	Top	End State	Flag Set	Phase	Descr...
Event Tree Name:	L-LOSPIE										
Transferring to event tree :	L-SUPP	1-5		TANK				LARGE-RELEASE		PHASE_1	
Transferring to event tree :	L-LOSP	1-4-3		/TANK	DG[3]	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG3	ECS-...	CCS-...				
Transferring to event tree :	L-LOSP	1-3-3		/TANK	DG[2]	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG2	ECS-A	CCS-A				
		1-3-2		/TANK	DG[2]	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		DG2	ECS-A	/CCS-A				
Transferring to event tree :	L-LOSP	1-2-3		/TANK	DG	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG1	ECS-B	CCS-B				
Transferring to event tree :	L-LOSP	1-1-3		/TANK	/DG	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		/DG0	ECS-0	CCS-0				
		1-1-2		/TANK	/DG	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		/DG0	ECS-0	/CCS-0				
Saved Sequences: 7 Va...											
TOTALS = Saved Sequences:...											
2015/02/05	Page #	13:46:19									
	Model Rev. ...										

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.

The following presents the cut sets of the L-LOSPIE event tree. Notice that the cut sets are the sequences generated during the link process.



NOTES

Appendix A – Link, Post Processing & 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 =	Post Processing	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;	Post Processing	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=	Post Processing	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;	Post Processing	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

End_Rule_Section	Link/Post-processing	Provides a break from one rule group to the next rule group in layered rules.	

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 cannot define a macro in event tree
			|"A" and expect to use it in event tree "B."

NewCutset;	Post Processing	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

Keyword or symbol	Type	Definition	Example Usage

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 =	Post Processing	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

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

SkipSequence
SkipSystem
	Link	Keywords with the same function as Skip( ), however it provides further delineation between sequence and system with the use of categories.	if “search criteria” then
			SkipSequence;
			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 Post Processing rules or partition rules.	if system(TOP EVENT) * “other search criteria if needed” then
			perform action on each sequence;
			endif

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

Keyword or symbol	Type	Definition	Example Usage

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 (placeholder not functional)	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

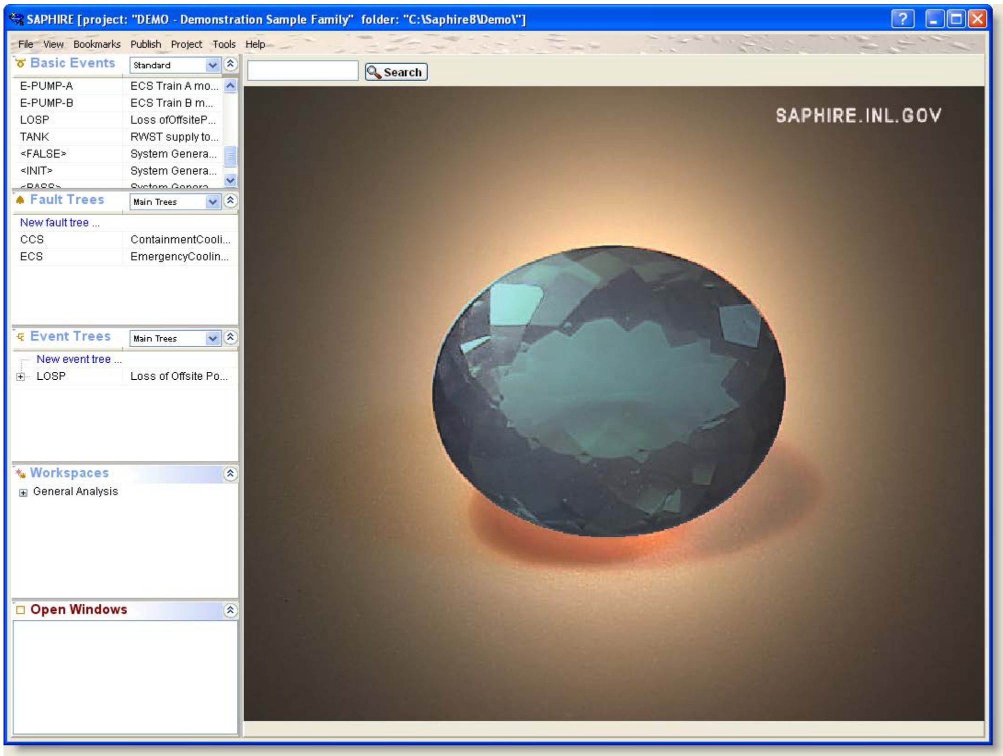
Discard (placeholder not functional)	Slice	Keyword to group the cut sets that meet the search criteria together for display in the "Excluded From Slice".	If "search criteria" then
			discard;
			endif

Convolve_Cut_Sets	Post Processing	Keyword that indicates the original cut set (i.e., that cut set that satisfied the search criteria) will be adjusted by a correction factor. The keyword for the correction factor in the rule is **AddConvolEvent**.	If Convolve_Cut_Sets then
			AddConvolEvent;
			endif

InvalidRASP_XProduct	Post Processing	Keyword that indicates the original cut set (i.e., that cut set that satisfied the search criteria) will be removed from the cut set. This is used when fully expanded R-Type CCF events are used in the PRA model	If InvalidRASP_XProduct then
			DeleteRoot;
			Endif

LerfFactor	Link	Keyword to assign LERF factors to sequences that meet the search criteria. These factors are used for SPAR models when performing SDP analyses.	if init(event tree) then
			lerffactor = 1.0; |numerical value between 0 < 1
			elsif RPS then
			endif

SAPHIRE 8 Advanced Workbook



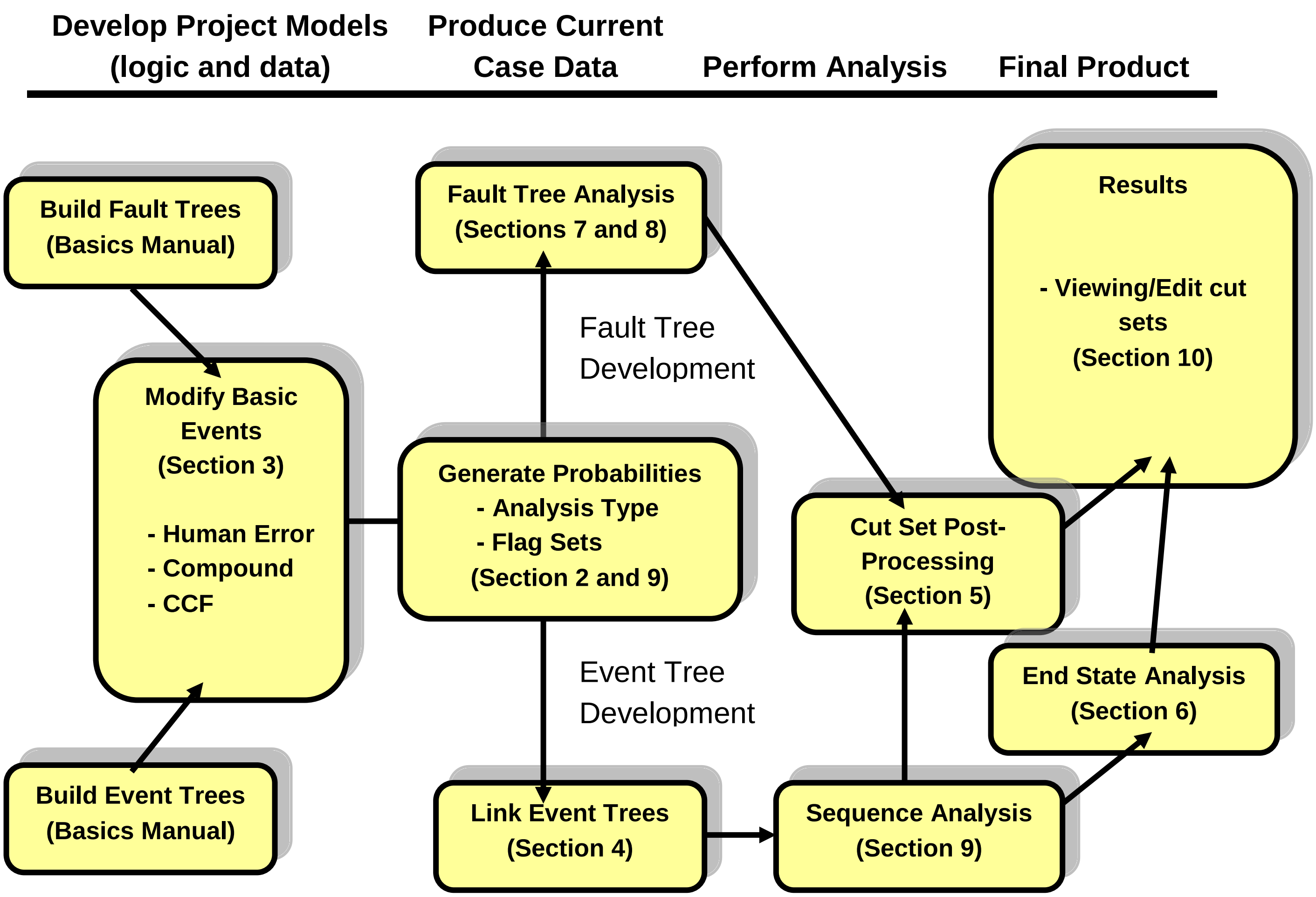
Curtis Smith
James Knudsen
Kurt Vedros

Idaho National Laboratory

February 2016

Idaho National Laboratory

SAPHIRE – The "Big Picture"



NOTICE

CONTENTS

NOTES

| 1 | INTRODUCTION

Section 1 contains an introduction to the SAPHIRE Advanced course workbook

This SAPHIRE Advanced course workbook accompanies the SAPHIRE Advanced training course manual. The workbook provides instructions for modeling exercises that will provide students with hands-on experience using advanced SAPHIRE features. The workbook provides model information for the workshop problems, describes the exercises to perform, and provides the results that should be achieved during the exercise.

The SAPHIRE Advanced training course manual and SAPHIRE Reference Manuals should be used to provide additional instructions as needed to perform the exercises.

Many of the workshop exercises will utilize simplified versions of a reactor and containment cooling systems. A diagram of this system is shown in Figure 1. The fault trees and event tree related to this system are contained in the DEMO-ADV project. Additional projects or modifications to DEMO-ADV will be utilized as needed for some of the workshops.

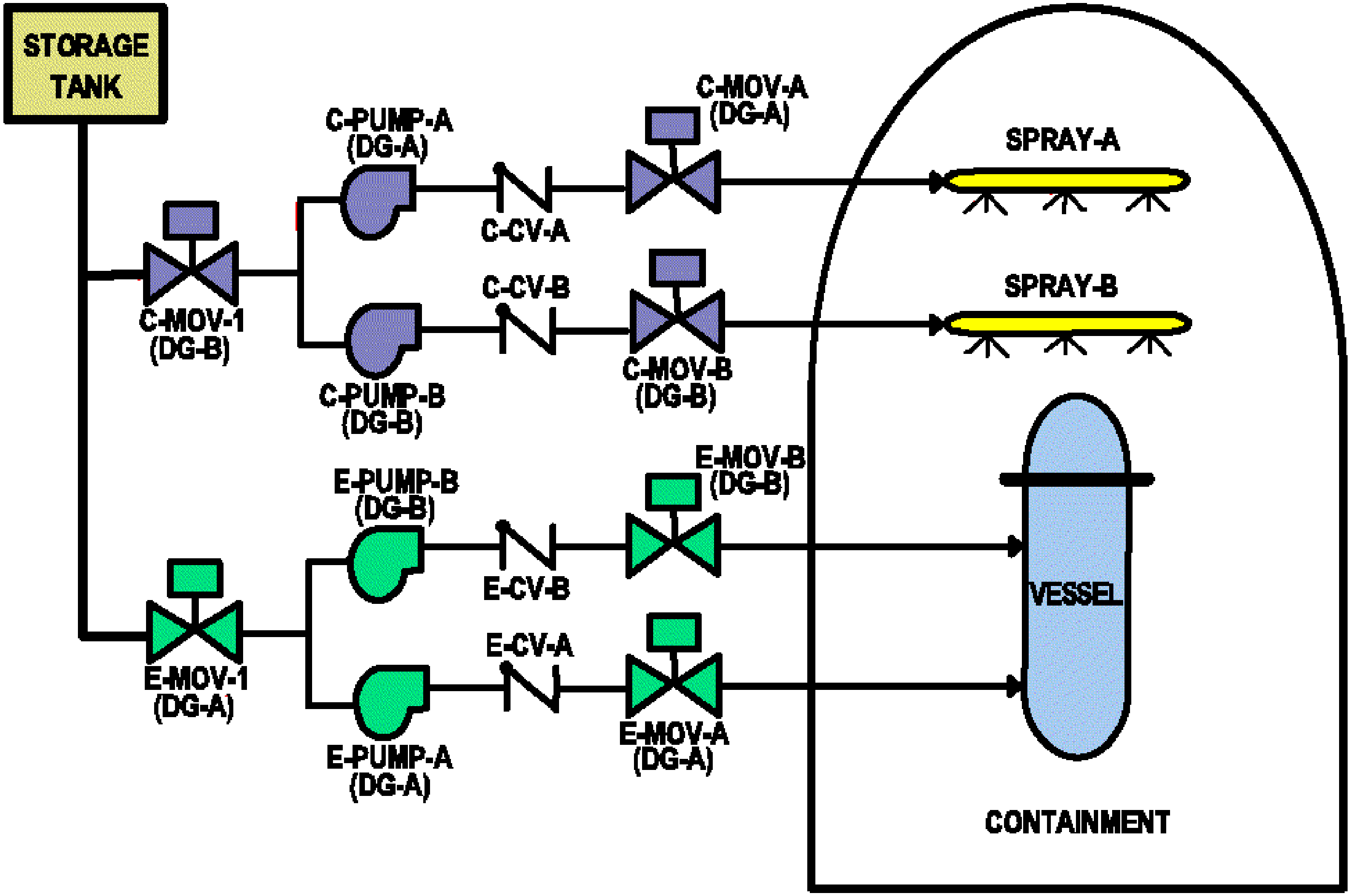


Figure 1. System diagram of the coolant systems to be used in the workshop exercises. Note that electrical support system dependencies are shown in brackets ( ).

| 2 | Database Concepts

Workshop Objectives

The SAPHIRE Data Base Concepts workshop objectives are to understand how to change the model type and to practice making Flag Sets.

The following advanced training manual sections are referenced:

- ♦ Section 2: Database Concepts
  - ◊ Projects, model types, and general flag set creation
- ♦ Section 7: Solving Fault Tree Cut Sets
  - ◊ Flag Set creation and use with fault trees
- ♦ Section 9: Solving Event Tree Sequence Cut Sets
  - ◊ Flag Set creation and use with event trees

Workshop Instructions

1. This workshop consists of two sections. First, the DEMO-ADV project will be used to switch between different Model Types (random and seismic). Second, two flag sets will be created in the DEMO-ADV project that will be utilized in a later workshop.

2. Select the DEMO-ADV project using the **Current Project** pull-down menu from the entry screen or **File → Open existing project…** if SAPHIRE is already open. (Make sure the DEMO-ADV project has been installed on your computer.)

3. Solve the ECS and CCS Fault Trees for the Random Model Type.

    a. In the **Fault Trees** list panel, right click on the desired fault tree(s) and select the **Solve** option. Place a check in the checkbox in ***only*** the **Random Failure Model Type** checkbox and solve at the default 1.00E-15 truncation.

    b. View the cut sets for the ECS and CCS fault trees by highlighting the fault tree, right click and selecting the **View Cut Sets** option.

4. Now, change the model type to **SEISMIC** by placing a check ***only*** in the **SEISMIC** Model Type checkbox on the fault tree solve form and proceed to solve for both ECS and CCS fault trees. View the cut sets for the ECS and CCS fault trees that used the seismic model type.

5. Solve the SEISMIC and LOSP event tree sequences using the RANDOM Model Type and then view the cut sets for the sequences using the **View Cut Sets** option of the event tree menu invoked by right-clicking on the event tree.

6. Change the Model Type to **SEISMIC** by placing a check only in the **SEISMIC** Model Type checkbox on the event tree solve form and proceed to solve for both SEISMIC and LOSP event trees. View the cut sets for the seismic sequences that used the seismic Model Type.

7. Perform seismic uncertainty analysis, highlight the SEISMIC event tree and select the **View Uncertainty** option and then specifying a specific g-level earthquake to determine the overall seismic result.

Creating Flag Sets

We will now use the DEMO-ADV project to create two Flag Sets (named FLAG-SET-SBO1 and FLAG-SET-SBO2). Note that these Flag Sets will be needed in later workshop exercises.

1. To make a Flag Set, enter the main menu **Project → Flag Sets** to open the **Flag Set List** form

2. Create a new flag sets by selecting **Add Flag Set** from the pull-down menu on the Flag Set List form and clicking the **Go** button. Enter the Flag Set names and descriptions provided below (two Flag Sets will be added and they will be modified in subsequent steps).

Name	***Description***
FLAG-SET-SBO1	"Both diesel generators (A & B) are failed"
FLAG-SET-SBO2	"Just diesel generator B is failed"

3. To make a data change in a Flag Set, highlight the Flag Set, select **Modify Flag Set** from the pull-down menu and click the **Go** button.
   - Drag and drop the basic events required into the **Singles** pane
   - Make the changes in the **Values to be Applied** pane
   - Click **OK** when finished

4. For the two Flag Sets that were just added, make the changes indicated below to the RANDOM model type field.

Name	Events to be Changed	Changes
FLAG-SET-SBO1	S-DGN-FS-A S-DGN-FS-B	Failure Model set to “T” (by clicking the Random Model Type only)
FLAG-SET-SBO2	S-DGN-FS-B	Calculation type set to “T” (by clicking the Random Model Type only)

5. These flag sets will be assigned to event tree sequences in a later workshop. However, a Flag Set can be assigned to a sequence via the **Sequence Properties** menu (not recommended):
   - Expand the sequences under the event tree in the event tree list panel
   - Highlight the desired event tree sequence
   - Right click on a highlighted sequence and select **Edit Properties**
   - The Flag Set name is entered in the field labeled **Default Flag Set** pull-down menu for the selected sequence

| 3 | Basic Event Information

Workshop Objectives

The Basic Event Information workshop accompanies Section 3 in the Advanced SAPHIRE training manual. The workshop objectives are to become familiar with the construction and use of compound events and human reliability worksheet.

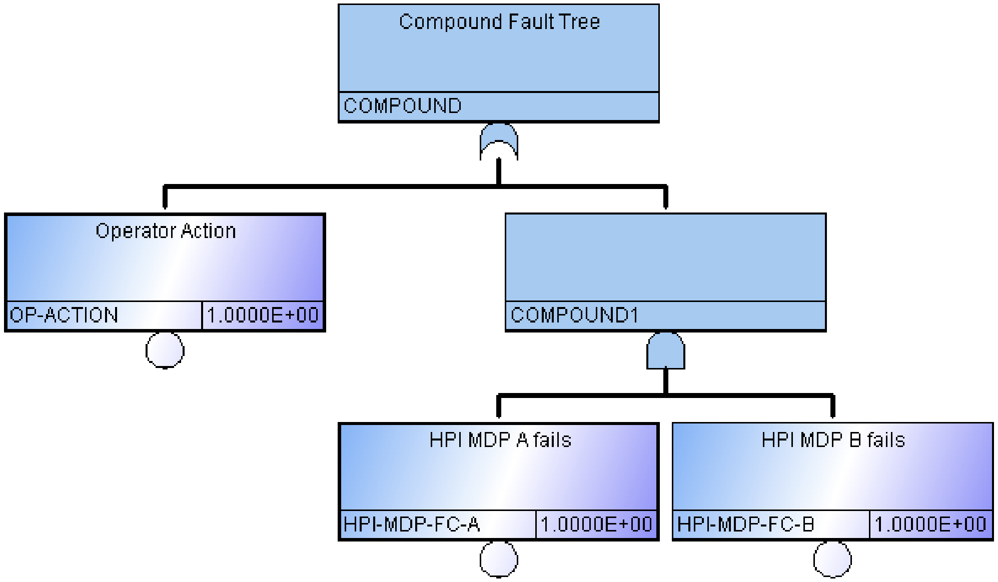
Workshop Instructions

This workshop consists of two sections. First, a new fault tree will be developed that will use compound events. Second, a basic event representing an operator action will be added to the fault tree.

Compound Basic Event - Using the DEMO-ADV project, a new fault tree needs to be developed which has three basic events, two basic events being a compound basic event and the third being a human reliability event. The compound events are comprised of other basic events that need to be added to the database.

1. Create a fault tree called **COMPOUND**. The fault tree top gate is an “OR” gate with and basic event and an “AND” gate. The “AND” gate contains two basic events as inputs:

   HPI-MDP-FC-A
   HPI-MDP-FC-B

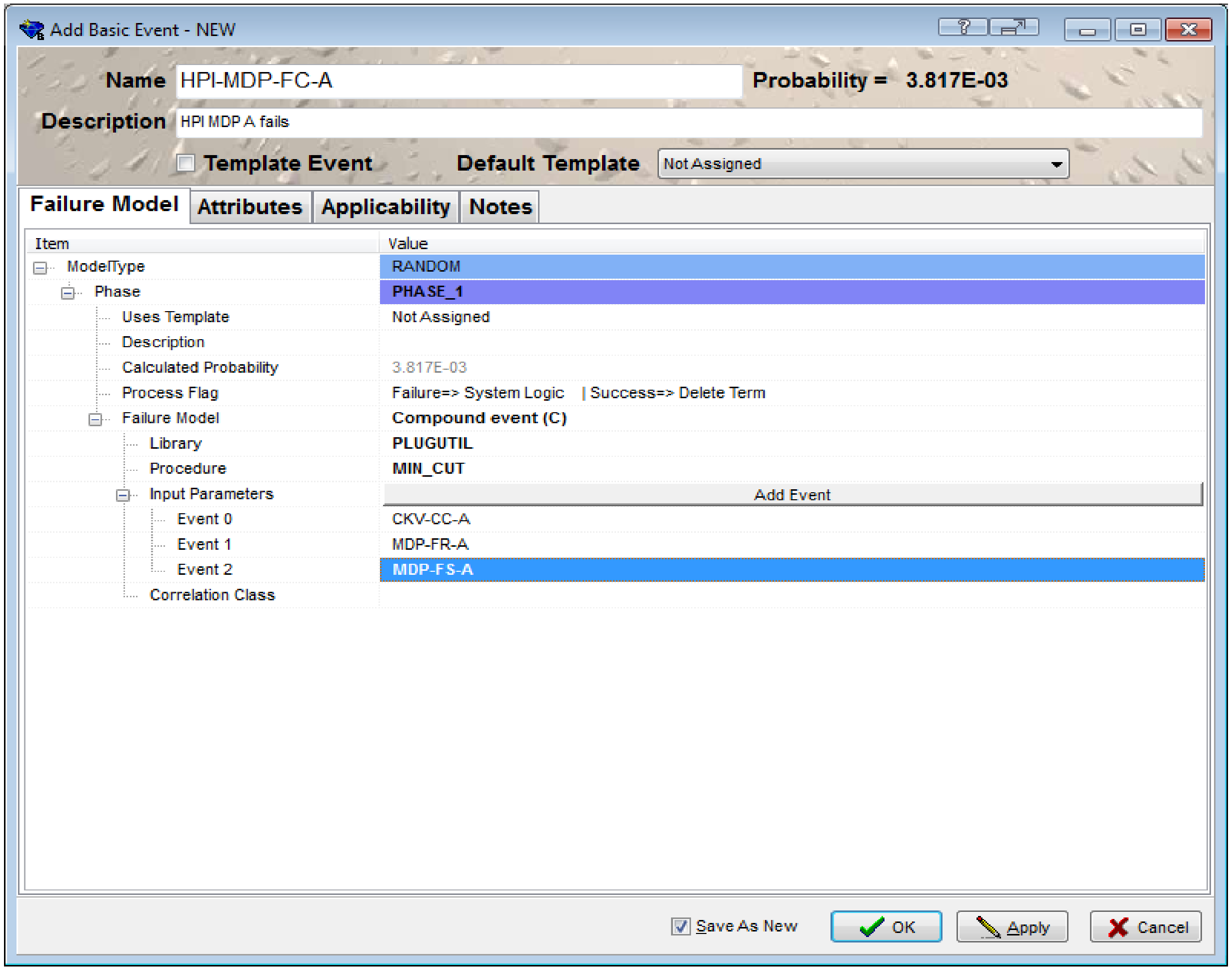


2. The basic events HPI-MDP-FC-A and HPI-MDP-FC-B are compound basic events. Basic event HPI-MDP-FC-A is comprised of MDP-FR-A, MDP-FS-A and CKV-CC-A, while basic event HPI-MDP-FC-B is comprised of MDP-FR-B, MDP-FS-B and CKV-CC-B. These basic events need to be added to the DEMO-ADV project database. (See below for the basic events that need to be added to the project and their specific information.)

3. Event HPI-MDP-FC-A needs to be made a compound event, by highlighting the basic event, right click to invoke the pop-up menu, then **Edit Basic Event**. Alternately, you can double click on the basic event either in the Basic Event list panel or on the basic event in the COMPOUND fault tree graphical representation. Change the **Failure Model** to “**Compound Event (C)”**.

Basic Event	Failure Model	Probability	Failure Rate	Mission Time	Lognormal EF	Correlation Class
MDP-FS-A	Failure Probability (1)	3.0E-3			5	1
MDP-FS-B	Failure Probability (1)	3.0E-3			5	1
MDP-FR-A	Fails to operate (without repair) (3)		3.0E-5	24	10	2
MDP-FR-B	Fails to operate (without repair) (3)		3.0E-5	24	10	2
CKV-CC-A	Failure Probability (1)	1.0E-4			3	3
CKV-CC-B	Failure Probability (1)	1.0E-4			3	3

4. Select "**PLUGUTIL**" from the **Library** list and then select "**MIN_CUT**" from the **Procedure** list. Click the **Add Event** button to input the basic events MDP-FR-A, MDP-FS-A, and CKV-CC-A into the sub event fields Event 0, Event 1, and Event 2. Note the probability will automatically update in the upper right of the Edit Basic Event form. Click the **Apply** or **OK** button to save the changes.

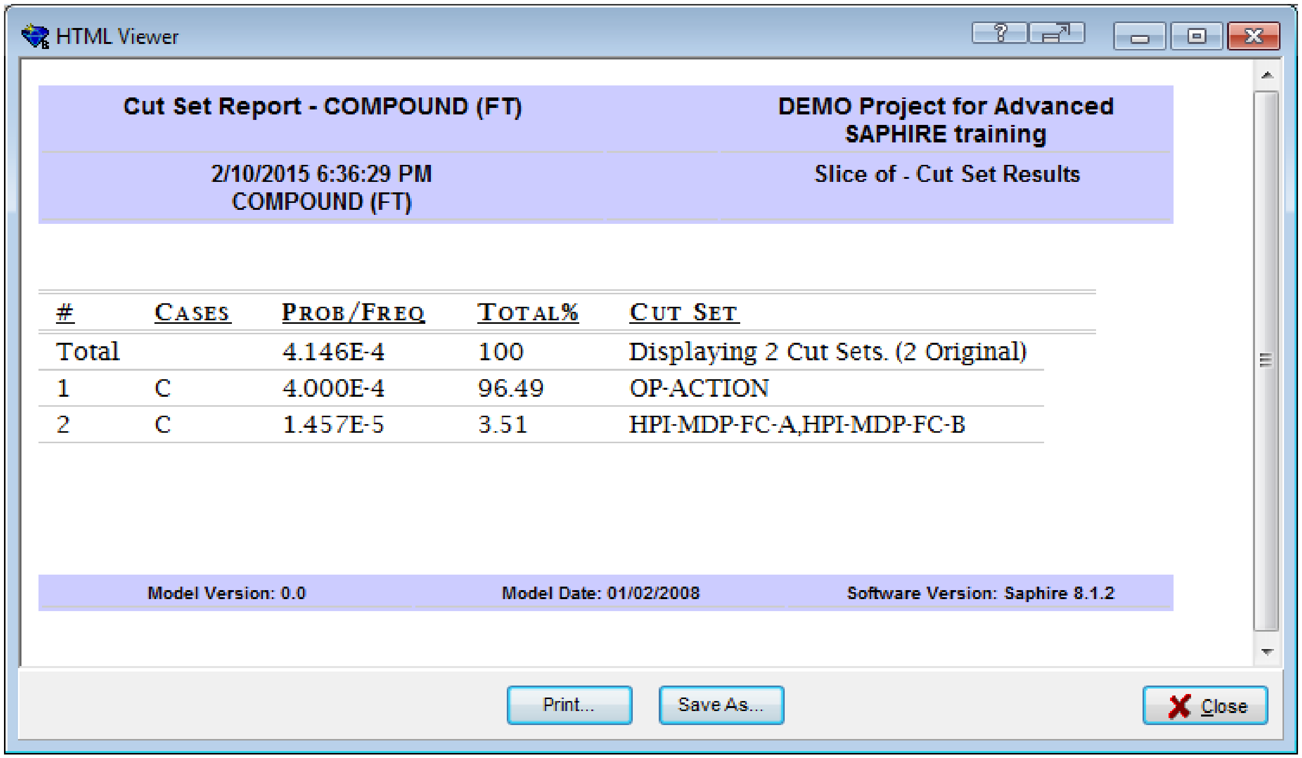


5. Repeat steps 3 and 4 for basic event HPI-MDP-FC-B, except use basic events MDP-FR-B, MDP-FS-B, and CKV-CC-B as the sub events.

6. Adjust the probability for the basic event “OP-ACTION”. Select the “SPAR-H human reliability model (X)” failure model. This operator action is only an action type of operation and no diagnosis is required. Adjust the following PSFs:

Available Time is “Extra time” [100%]
Stress/Stressors is “high” [100%]
Complexity is “moderately complex” [100%]

6. Solve cut sets for the **COMPOUND** fault tree. Two cut sets should exist:


HTML Viewer

Cut Set Report - COMPOUND (FT)	DEMO Project for Advanced SAPHIRE training
2/10/2015 6:36:29 PM COMPOUND (FT)	Slice of - Cut Set Results

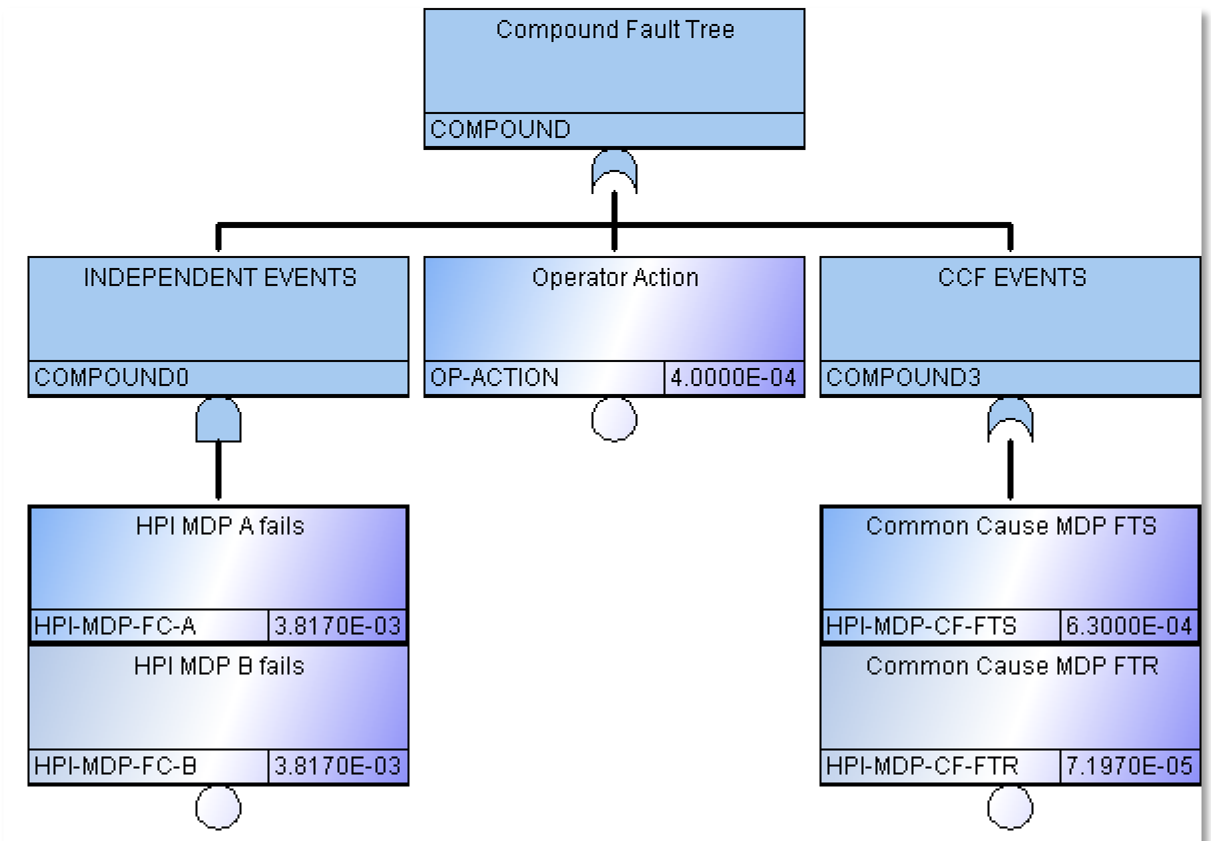
#	Cases	Prob/Freq	Total%	Cut Set
Total		4.146E-4	100	Displaying 2 Cut Sets. (2 Original)
1	C	4.000E-4	96.49	OP-ACTION
2	C	1.457E-5	3.51	HPI-MDP-FC-A,HPI-MDP-FC-B

Model Version: 0.0	Model Date: 01/02/2008	Software Version: Saphire 8.1.2

Print... Save As... Close

Additional Workshop - Using the different common cause failure equations

1. Add two CCF basic events (i.e., HPI-MDP-CF-FTS and HPI-MDP-CF-FTR) to the **COMPOUND** fault tree by changing the top gate to an “OR” and adding a new OR gate as an input into **COMPOUND** that the two CCF basic events will be an input to. The two independent basic events (HPI-MDP-FC-A and HPI-MDP-FC-B) need to be placed under a new “AND” gate (called **COMPOUND0**). The fault tree should look as follows:



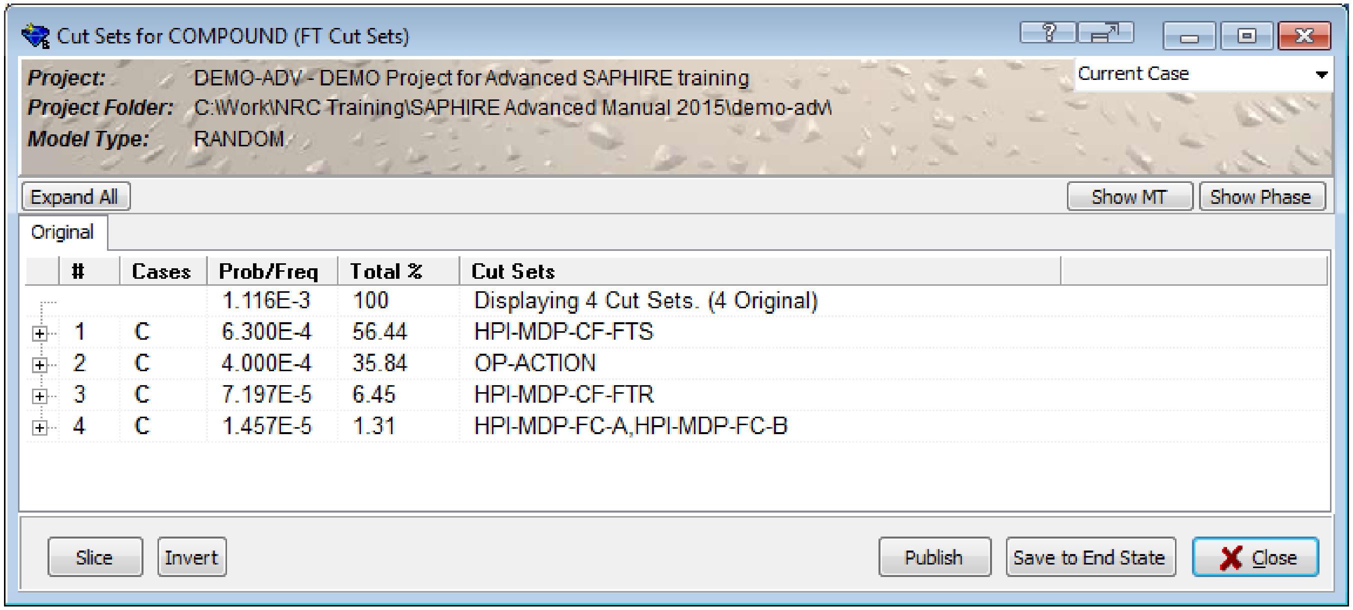
(Optional) For additional practice, add descriptions to the gates and basic events as shown.

2. For the CCF basic event HPI-MDP-CF-FTS, use the MGL equation:
   - Failure Model = **Compound Event (C)**
   - Library = **PLUGCCFMGL**
   - Procedure = **TwoEventGroup**
   - Input Parameters (see paragraph below)
     - Failure Count = **2**
     - CCFEvent1 = **MDP-FS-A**
     - CCFEvent2 = **MDP-FS-B**
     - Beta = **BETA**

   The MDP-FS-A and MDP-FS-B can be used as the independent events for this CCF basic event. A new "**Basic Event**" called **BETA** needs to be added for use in the MGL equation, since the equation requires a β-factor. Set the value for the β-factor (basic event **BETA**) to 0.21. Add the applicable events into their correct fields and note the calculated probability (6.3E-04).

3. For the CCF basic event HPI-MDP-CF-FTR, use the Alpha Factor equation (**PLUGCCFSTAG**). The MDP-FR-A and MDP-FR-B can be used as the independent events for this CCF basic event. New "**Basic events**" called **ALPHA-1** and **ALPHA-2** need to be added for use in the alpha factor equation. The values for these events are alpha-1 = 0.9 and alpha-2 = 0.1. Add the applicable events into their correct fields note the calculated probability (7.197E-05).

4. Solve cut sets for the **COMPOUND** fault tree. Three cut sets should be generated:


Cut Sets for COMPOUND (FT Cut Sets)
Project: DEMO-ADV - DEMO Project for Advanced SAPHIRE training
Project Folder: C:\Work\NRC Training\SAPHIRE Advanced Manual 2015\demo-adv\
Model Type: RANDOM
Current Case
Expand All
Show MT
Show Phase
Original
Cases Prob/Freq Total % Cut Sets
1.116E-3 100 Displaying 4 Cut Sets. (4 Original)
1 C 6.300E-4 56.44 HPI-MDP-CF-FTS
2 C 4.000E-4 35.84 OP-ACTION
3 C 7.197E-5 6.45 HPI-MDP-CF-FTR
4 C 1.457E-5 1.31 HPI-MDP-FC-A,HPI-MDP-FC-B
Slice
Invert
Publish
Save to End State
Close

| 4 | Linking Event Trees

Workshop Objectives

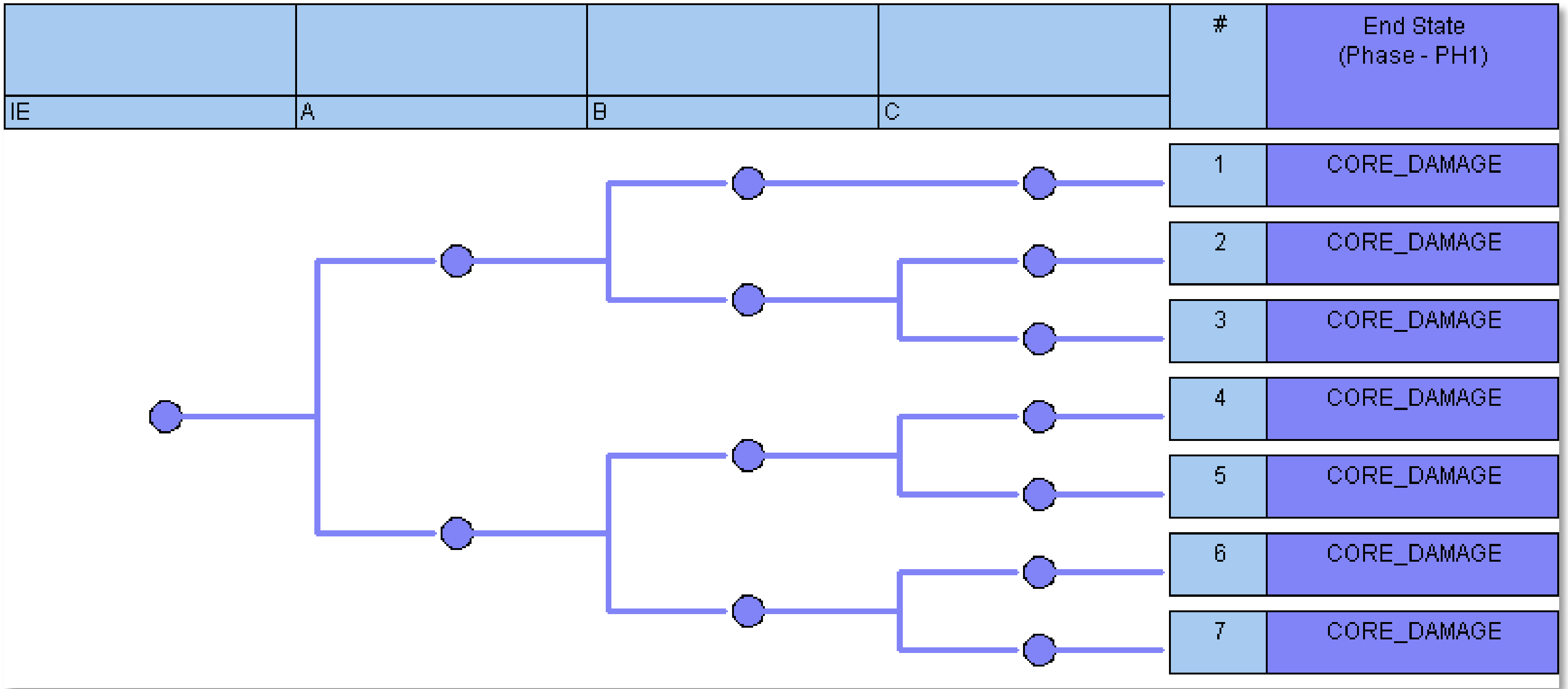
The "Link Event Trees" Rule Editor workshop accompanies Section 4 in the Advanced SAPHIRE training manual. The workshop objectives are to practice using the rule editor and become more familiar with the rule nomenclature.

Workshop Instructions

To practice using the “Link Event Tree” rules, create a *new project* named **RULES** and use this project to edit event tree linking rules.

1. Create a new SAPHIRE project named **RULES** (main menu **File** → **New** → **Project**).

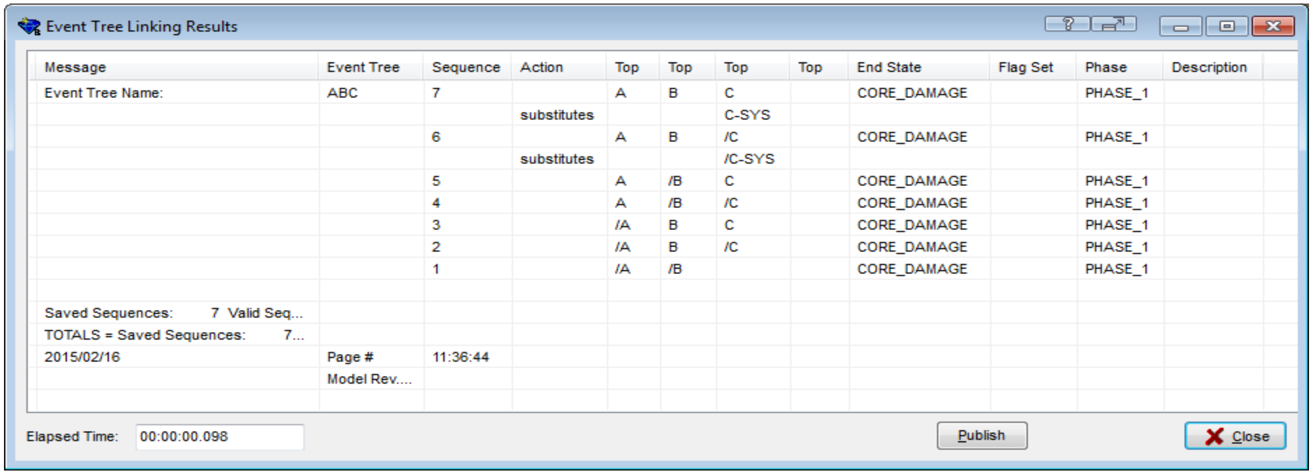
2. Create the event tree shown by selecting the **New Event Tree** in the Event Tree list panel. Name the new event tree **ABC** and specify its initiating event as IE.


IE
A
B
C
#
End State
(Phase - PH1)
1 CORE_DAMAGE
2 CORE_DAMAGE
3 CORE_DAMAGE
4 CORE_DAMAGE
5 CORE_DAMAGE
6 CORE_DAMAGE
7 CORE_DAMAGE

3. Enter the rules shown below for event tree ABC by highlighting the event tree, right clicking the mouse, and selecting **Edit Linkage Rules**. Note that the "system" C-SYS is called by the rule, so this new fault tree called **C-SYS** needs to be created prior to saving the linking rule or it will not compile. **C-SYS** does not require any inputs to its top gate for this workshop.

```
| RULE 1.
| 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
```

4. When the rule has been added and compiled (from the editor, use the compile button, then **File → Save/Exit**). Link the event tree sequences by right clicking on the ABC event tree in the Event Tree list panel to invoke the menu, then choosing using the **Link** option. Sequence logic can be printed to the screen by clicking the checkbox Create Report.



Message	Event Tree	Sequence	Action	Top	Top	Top	Top	End State	Flag Set	Phase	Description
Event Tree Name:	ABC	7		A	B	C		CORE_DAMAGE		PHASE_1	
			substitutes			C-SYS					
		6		A	B	/C		CORE_DAMAGE		PHASE_1	
			substitutes			/C-SYS					
		5		A	/B	C		CORE_DAMAGE		PHASE_1	
		4		A	/B	/C		CORE_DAMAGE		PHASE_1	
		3		/A	B	C		CORE_DAMAGE		PHASE_1	
		2		/A	B	/C		CORE_DAMAGE		PHASE_1	
		1		/A	/B			CORE_DAMAGE		PHASE_1	
Saved Sequences: 7 Valid Seq...											
TOTALS = Saved Sequences: 7...											
2015/02/16	Page #	11:36:44									
	Model Rev....										

5. Experiment by adding the different rules listed below. Remember to delete or "comment out" (by placing a "|" at the beginning of the line) the prior rules. The new fault trees need to be added to the database.

```
| RULE 2.
| 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
```

```
| RULE 3.
| The "if-then-elsif" Structure:
| This rule replaces C with C-AB if A and B are
| failed, & 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
```

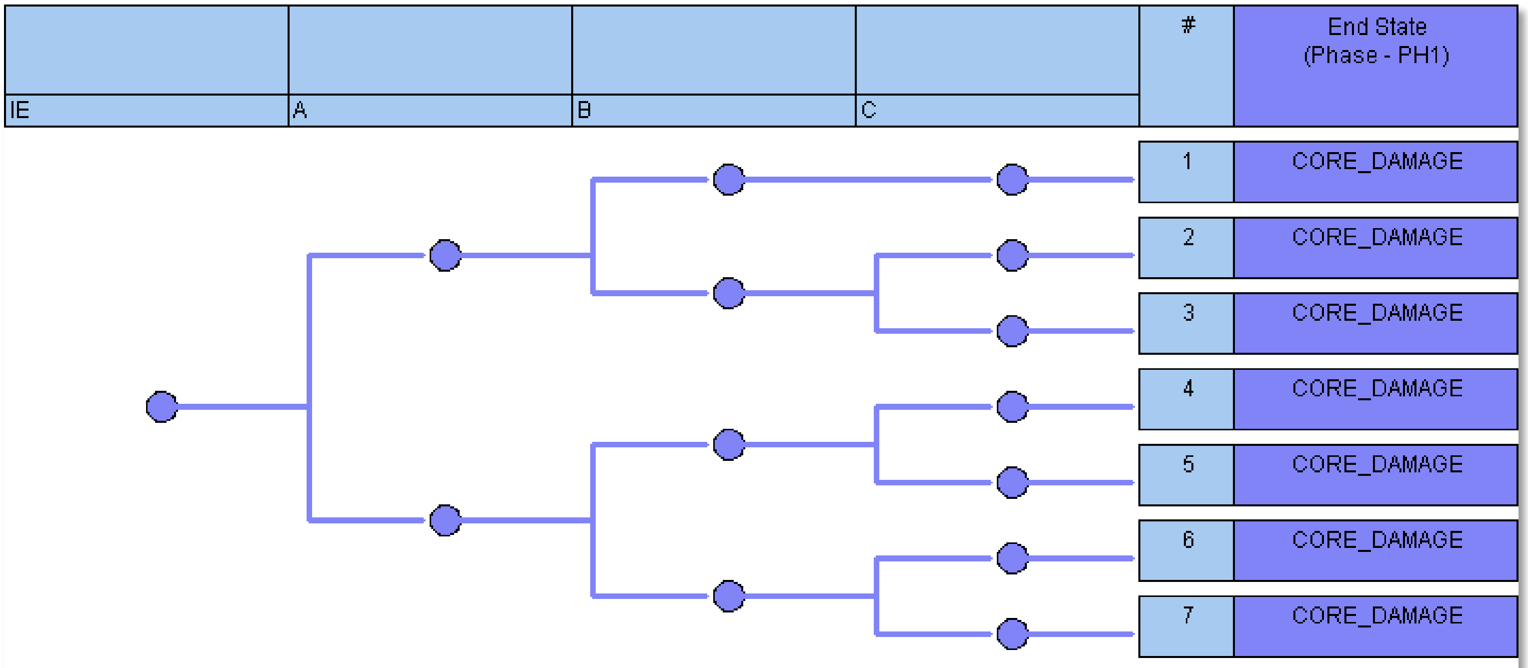
```
| RULE 4.  The "if-then-elsif-else" Structure:
| Rule replaces C with C-NA when A is successful,
| replaces C with C-NB if B is successful, & replaces
| C with C-XX in any other case
if /A  then
  /C = C-NA;
   C = C-NA;
elsif /B then
  /C = C-NB;
   C = C-NB;
else
  /C = C-XX;
   C = C-XX;
endif
```

```
| RULE 5.
| The "Skip" Structure: Rule has the ability to "skip"
| sequences in sequence logic.  The rule also uses a
| MACRO to search on system B.
SEARCH-B = B;
if  SEARCH-B  then
  /C = Skip(C);
  C = Skip(C);
endif
```

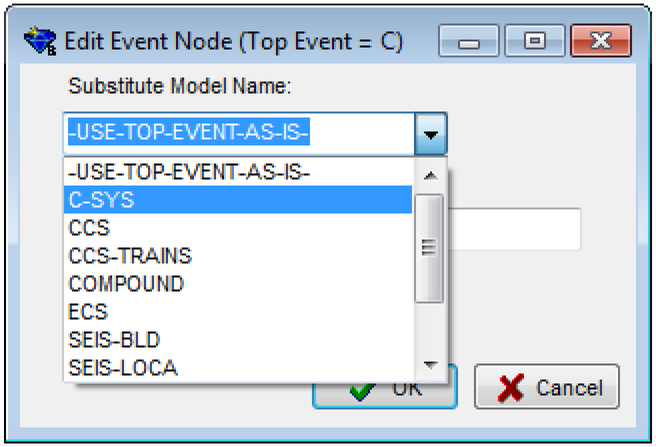
Optional Workshop:

1. Using the ABC event tree, perform a top event substitution in the graphical editor.

2. Highlight the ABC event tree, right click and select the **Edit Logic** (or double left mouse click to open the ABC event tree).

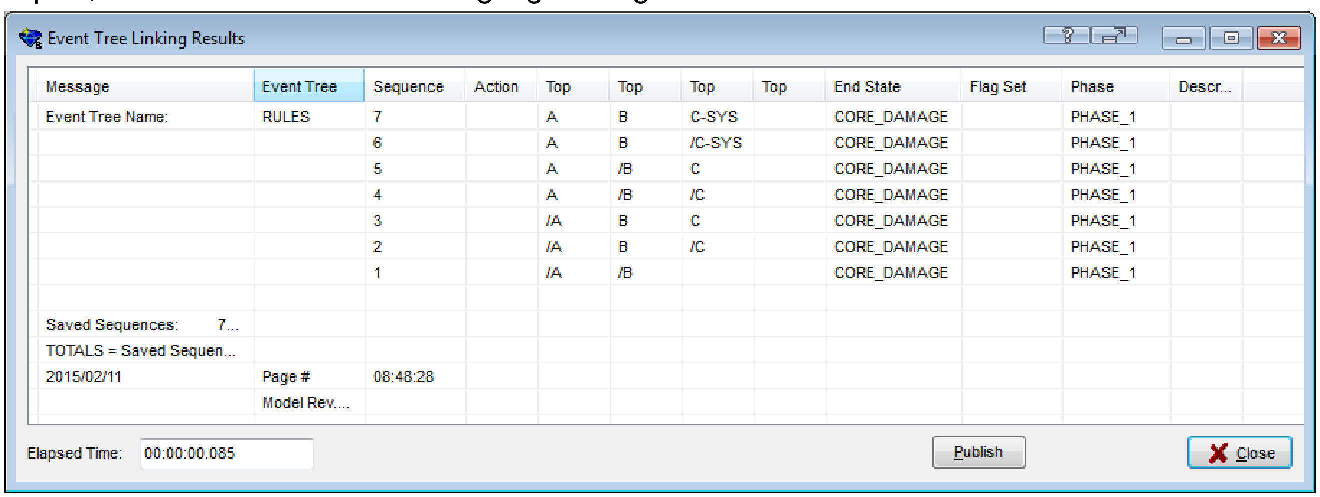


3. Select the node under top event C for sequence 6, right click and select **Edit**, click the drop down box “Substitute Model Name” and change from “–USE-TOP-EVENT-AS-IS-“ to “C-SYS”.



4. Perform this same step for the top event C for sequence 7.

5. Save the event tree. SAPHIRE will automatically link the even tree and update the sequence logic. To get a report, with the ABC event tree highlighted right click and select "Link".



| 5 | Post Processing Rules

Workshop Objectives

The Post Processing Rules workshop accompanies Section 5 in the Advanced SAPHIRE training manual. The workshop objectives are to gain experience using the Post Processing Rules editors, applying Post Processing Rules, and viewing cut sets after modification via the Post Processing Rules.

Workshop Instructions

Post processing rules will be created for the ECS fault tree in the DEMO-ADV project. These post processing rules will then be applied to the ECS fault tree where the nominal case and current case ECS cut sets can be compared to see the effect of the rules.

1. Open the DEMO-ADV project. Prior to creating the post processing rule, make a new basic event called SYS-ECS (via the Basic Events list panel **New basic event…** option and set its probability to 0.5). Enter the post processing rules shown using the appropriate rule editor.

Rule type	Menu	Rule

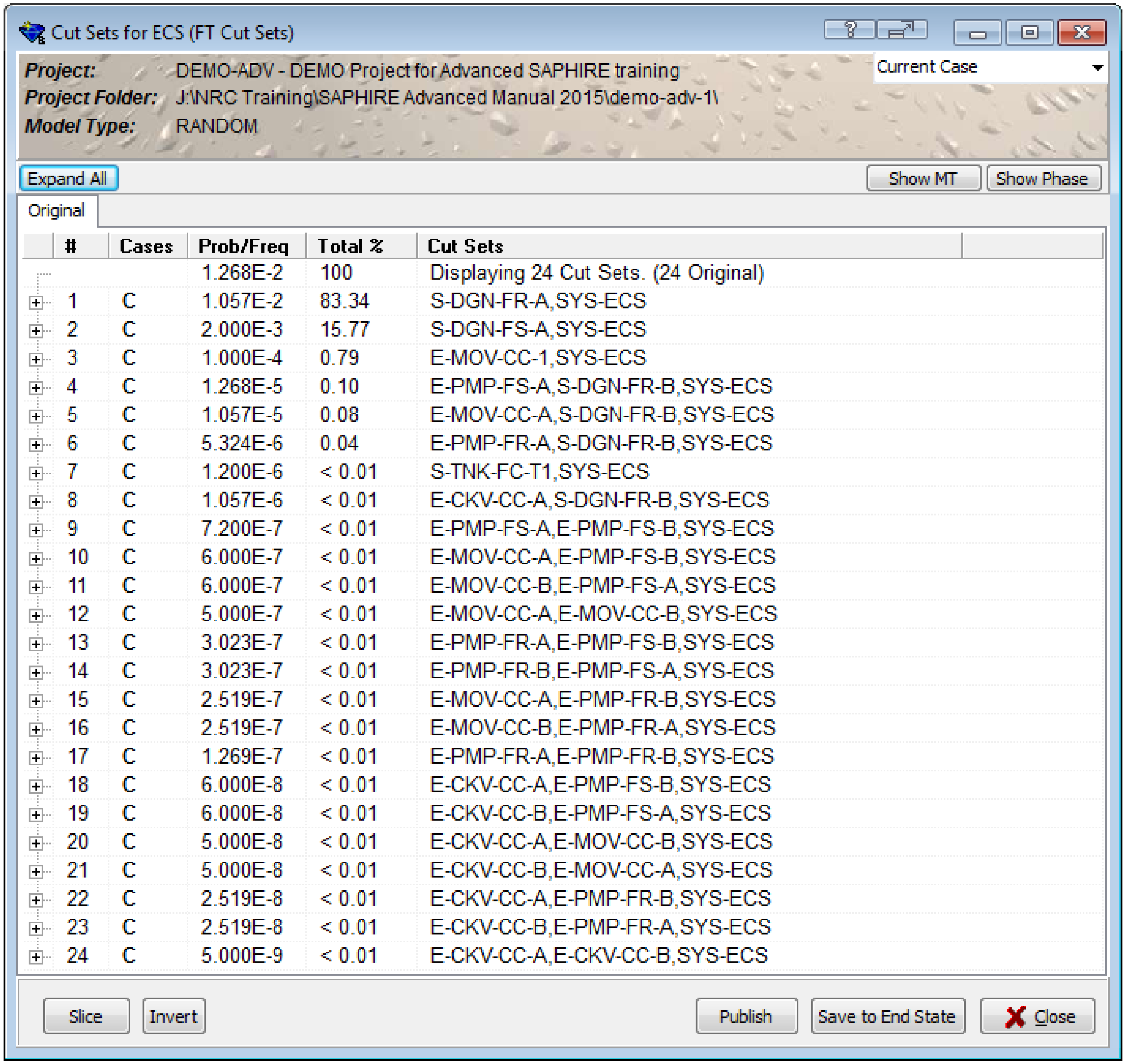
Fault tree	Highlight the ECS fault tree. Right click and select the **Edit Post-processing Rules** option.	`| Add an identifier onto ECS cut sets`
		`if always then`
		`   AddEvent = SYS-ECS;`
		`endif`

Fault tree project	Main menu **Project → Edit Rules → FT (Post-processing)** option.	`| Take out cut sets that have DG-B`
		`if S-DGN-FS-B then`
		`   DeleteRoot;`
		`endif`

2. Unmark any change sets that may have been previously marked.

3. Solve the ECS Fault Tree with a check in the checkbox “**Apply Post-processing Rules**”.

4. View the fault tree cut sets to see what changes have been applied to the cut sets.



#	Cases	Prob/Freq	Total %	Cut Sets
		1.268E-2	100	Displaying 24 Cut Sets. (24 Original)
1	C	1.057E-2	83.34	S-DGN-FR-A,SYS-ECS
2	C	2.000E-3	15.77	S-DGN-FS-A,SYS-ECS
3	C	1.000E-4	0.79	E-MOV-CC-1,SYS-ECS
4	C	1.268E-5	0.10	E-PMP-FS-A,S-DGN-FR-B,SYS-ECS
5	C	1.057E-5	0.08	E-MOV-CC-A,S-DGN-FR-B,SYS-ECS
6	C	5.324E-6	0.04	E-PMP-FR-A,S-DGN-FR-B,SYS-ECS
7	C	1.200E-6	< 0.01	S-TNK-FC-T1,SYS-ECS
8	C	1.057E-6	< 0.01	E-CKV-CC-A,S-DGN-FR-B,SYS-ECS
9	C	7.200E-7	< 0.01	E-PMP-FS-A,E-PMP-FS-B,SYS-ECS
10	C	6.000E-7	< 0.01	E-MOV-CC-A,E-PMP-FS-B,SYS-ECS
11	C	6.000E-7	< 0.01	E-MOV-CC-B,E-PMP-FS-A,SYS-ECS
12	C	5.000E-7	< 0.01	E-MOV-CC-A,E-MOV-CC-B,SYS-ECS
13	C	3.023E-7	< 0.01	E-PMP-FR-A,E-PMP-FS-B,SYS-ECS
14	C	3.023E-7	< 0.01	E-PMP-FR-B,E-PMP-FS-A,SYS-ECS
15	C	2.519E-7	< 0.01	E-MOV-CC-A,E-PMP-FR-B,SYS-ECS
16	C	2.519E-7	< 0.01	E-MOV-CC-B,E-PMP-FR-A,SYS-ECS
17	C	1.269E-7	< 0.01	E-PMP-FR-A,E-PMP-FR-B,SYS-ECS
18	C	6.000E-8	< 0.01	E-CKV-CC-A,E-PMP-FS-B,SYS-ECS
19	C	6.000E-8	< 0.01	E-CKV-CC-B,E-PMP-FS-A,SYS-ECS
20	C	5.000E-8	< 0.01	E-CKV-CC-A,E-MOV-CC-B,SYS-ECS
21	C	5.000E-8	< 0.01	E-CKV-CC-B,E-MOV-CC-A,SYS-ECS
22	C	2.519E-8	< 0.01	E-CKV-CC-A,E-PMP-FR-B,SYS-ECS
23	C	2.519E-8	< 0.01	E-CKV-CC-B,E-PMP-FR-A,SYS-ECS
24	C	5.000E-9	< 0.01	E-CKV-CC-A,E-CKV-CC-B,SYS-ECS

5. Return to the recovery rules and either delete or comment out (using “|”) the recovery rules that were created.

NOTES

| 6 | End State Analysis

Workshop Objectives

The End State Analysis workshop accompanies Section 6 in the Advanced SAPHIRE training manual. The workshop objective is to gain experience using the End State Analysis module.

Workshop Instructions

Using the DEMO-ADV project, a new event tree named SBO will be constructed. The sequences for both the LOSP and SBO event trees need to be solved and then gather the end state cut sets.

1. Open the DEMO-ADV project. Construct the new event tree SBO as shown in Figure 2.

2. Generate the SBO sequences, these are done automatically generated when the event tree is saved or by using the Event Trees list panel menu (right click on SBO event tree) and using the **Link** option. Five new sequences should be created.

3. Access the SBO top events through the Basic Events list panel pull down menu **Developed Event** or **All** selection
   - Modify the top event RDG-A to change its mean failure probability to 0.25 and its Process Flag to “Y”
   - Modify the initiating event SBO to a mean frequency of 4.0E-3/yr.

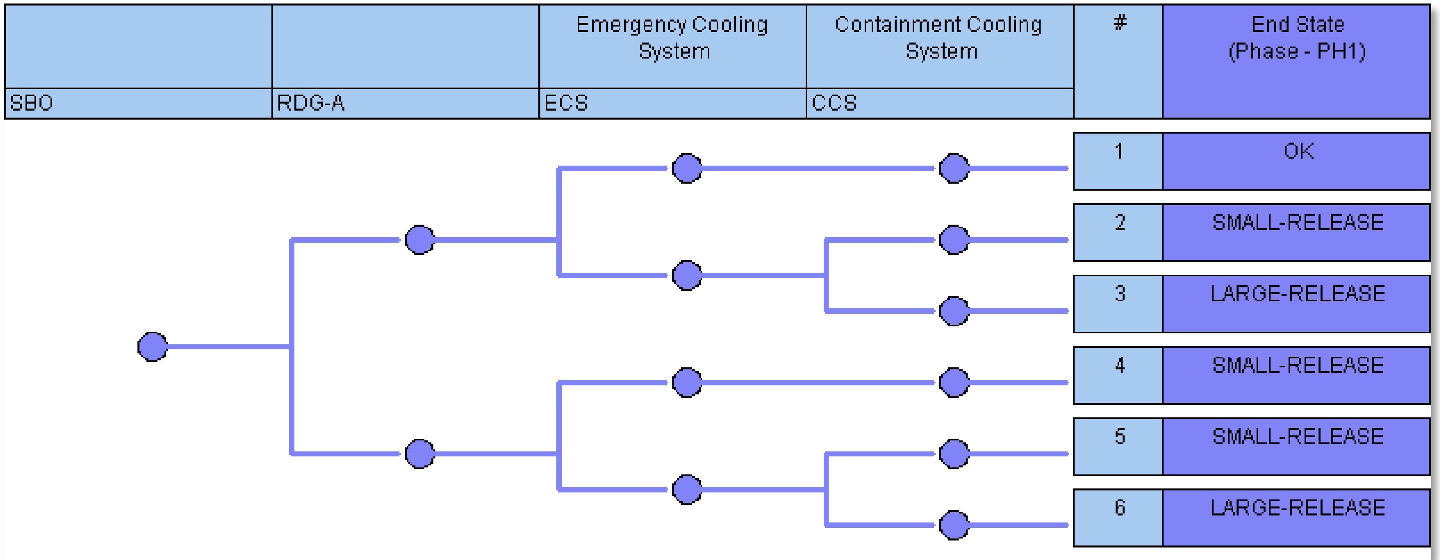


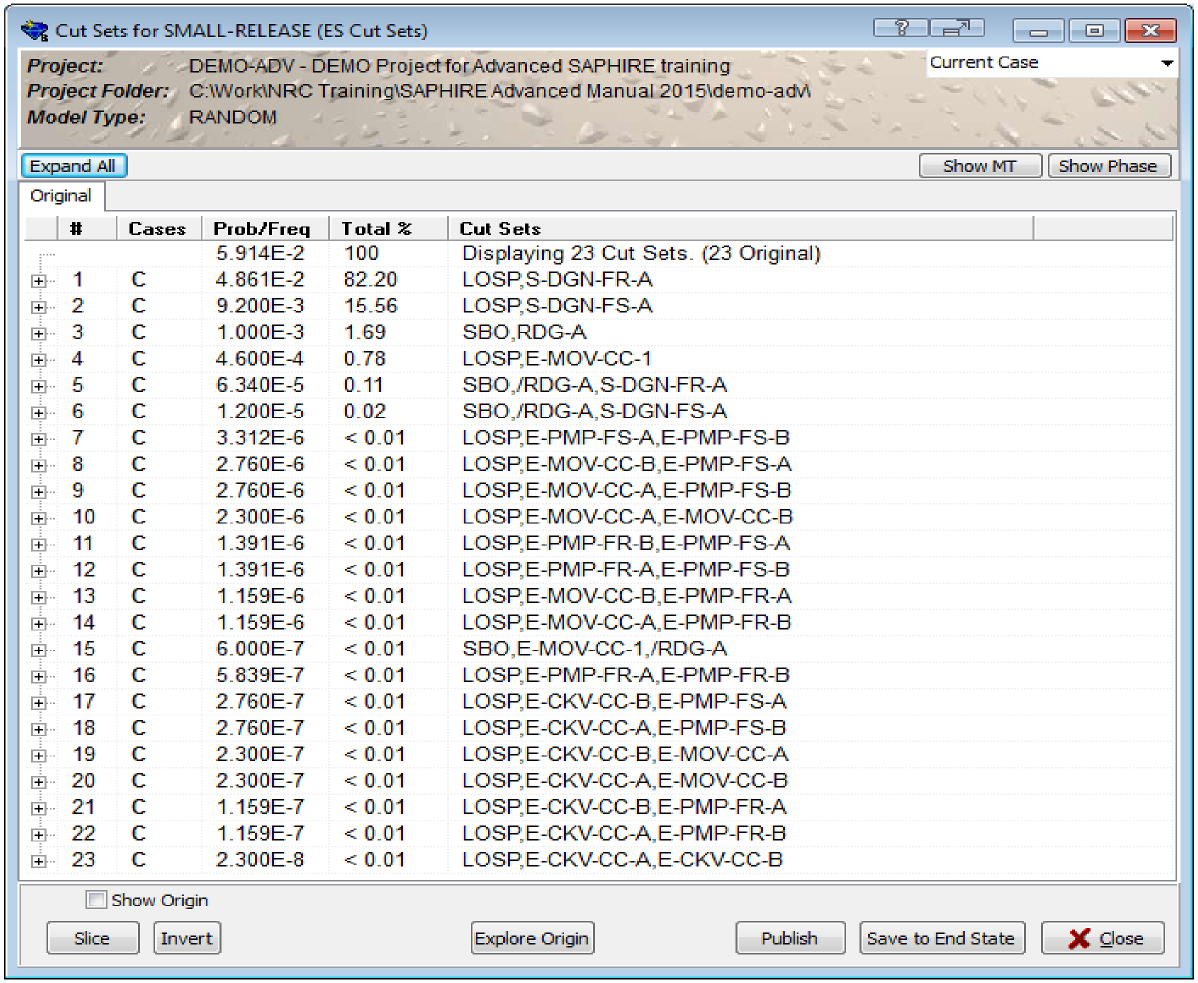
Figure 2**. The SBO event tree required for the end state workshop.**

4. Solve sequence cut sets with no change sets marked for all sequences for both the LOSP and SBO event trees using a frequency truncation of 1.0E-8.

5. View the End States list panel by selecting main menu **View** → **End States**.

6. Gather end state cut sets by highlighting the end states SMALL-RELEASE and LARGE-RELEASE in the **End States** list panel, right click and select Gather using a frequency truncation of 1.0E-8.

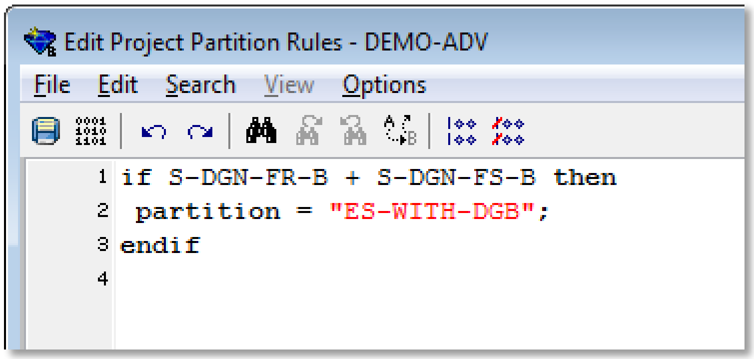
7. View the end state cut sets by highlighting the SMALL-RELEASE end state right click and select **View Cut Sets**. Compare your results with those shown below.



#	Cases	Prob/Freq	Total %	Cut Sets
		5.914E-2	100	Displaying 23 Cut Sets. (23 Original)
1	C	4.861E-2	82.20	LOSP,S-DGN-FR-A
2	C	9.200E-3	15.56	LOSP,S-DGN-FS-A
3	C	1.000E-3	1.69	SBO,RDG-A
4	C	4.600E-4	0.78	LOSP,E-MOV-CC-1
5	C	6.340E-5	0.11	SBO,/RDG-A,S-DGN-FR-A
6	C	1.200E-5	0.02	SBO,/RDG-A,S-DGN-FS-A
7	C	3.312E-6	< 0.01	LOSP,E-PMP-FS-A,E-PMP-FS-B
8	C	2.760E-6	< 0.01	LOSP,E-MOV-CC-B,E-PMP-FS-A
9	C	2.760E-6	< 0.01	LOSP,E-MOV-CC-A,E-PMP-FS-B
10	C	2.300E-6	< 0.01	LOSP,E-MOV-CC-A,E-MOV-CC-B
11	C	1.391E-6	< 0.01	LOSP,E-PMP-FR-B,E-PMP-FS-A
12	C	1.391E-6	< 0.01	LOSP,E-PMP-FR-A,E-PMP-FS-B
13	C	1.159E-6	< 0.01	LOSP,E-MOV-CC-B,E-PMP-FR-A
14	C	1.159E-6	< 0.01	LOSP,E-MOV-CC-A,E-PMP-FR-B
15	C	6.000E-7	< 0.01	SBO,E-MOV-CC-1,/RDG-A
16	C	5.839E-7	< 0.01	LOSP,E-PMP-FR-A,E-PMP-FR-B
17	C	2.760E-7	< 0.01	LOSP,E-CKV-CC-B,E-PMP-FS-A
18	C	2.760E-7	< 0.01	LOSP,E-CKV-CC-A,E-PMP-FS-B
19	C	2.300E-7	< 0.01	LOSP,E-CKV-CC-B,E-MOV-CC-A
20	C	2.300E-7	< 0.01	LOSP,E-CKV-CC-A,E-MOV-CC-B
21	C	1.159E-7	< 0.01	LOSP,E-CKV-CC-B,E-PMP-FR-A
22	C	1.159E-7	< 0.01	LOSP,E-CKV-CC-A,E-PMP-FR-B
23	C	2.300E-8	< 0.01	LOSP,E-CKV-CC-A,E-CKV-CC-B

8. View the LARGE-RELEASE end state cut sets by selecting the **View Cut Sets** menu. You should see a total of 52 cut sets with an overall value of 1.808E-03.

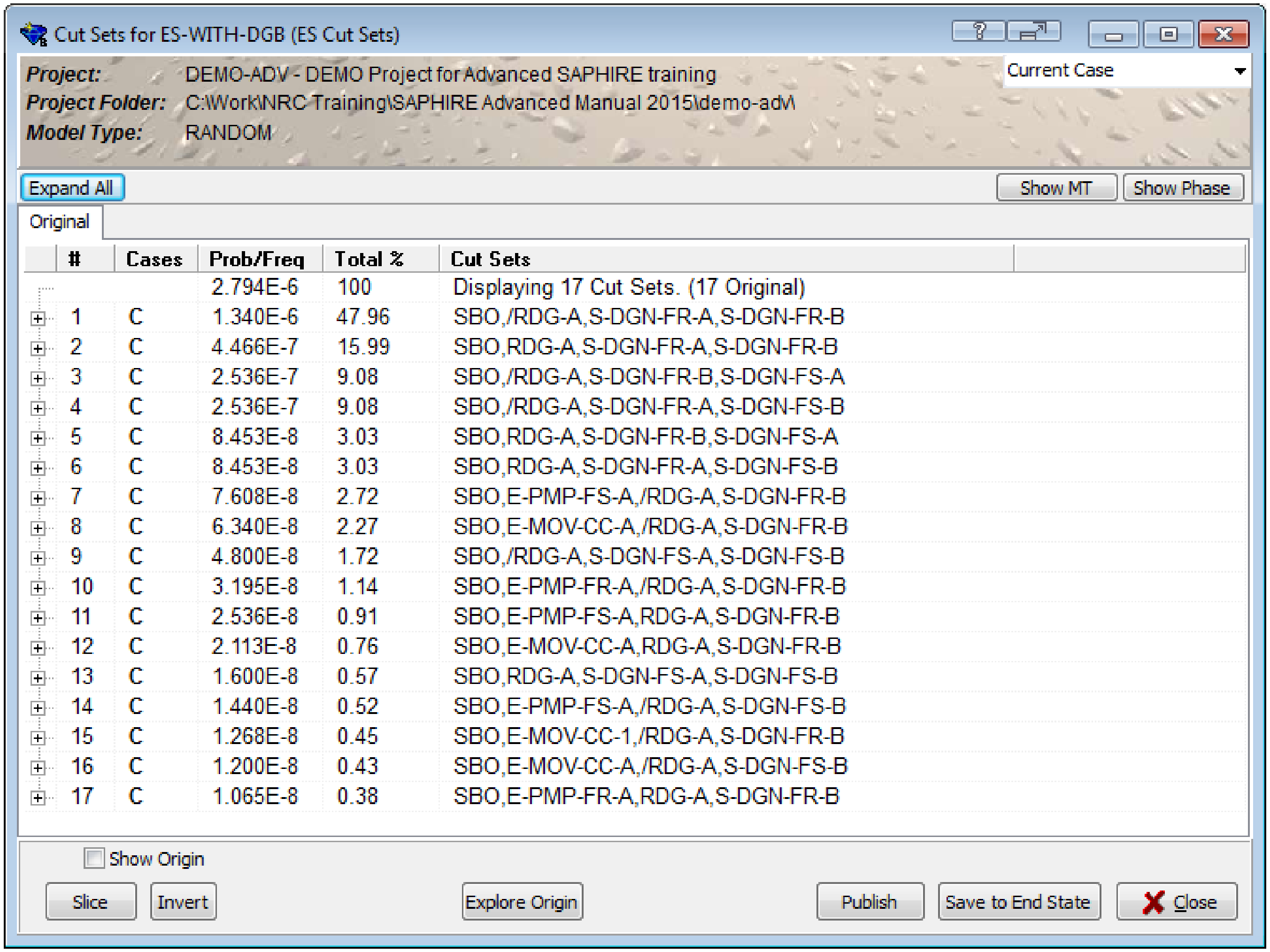
9. Now, add a partition rule that will create a new end state containing cut sets that have S-DGN-FS-B and S-DGN-FR-B in them (similar to example Rule 5 in the Advanced Manual). Enter the rule below into the project partition rules via the **Project → Edit Rules → End State Partition** option.



10. Apply the partition rule:
    - Highlight ***only*** the SBO event tree in the event tree list panel (this selects all sequences in SBO)
    - Solve SBO sequence cut sets with the Apply Partition Rules checkbox checked and truncation set at 1.000E-08

11. Now, the new end state ES-WITH-DGB exists, but the “partitioned” cut sets need to be gathered into this end state. To do this, highlight the listed end state in the **End States** list panel (and gather with **Gather By → Cut Set Partition** radio button selected).

12. View the ES-WITH-DGB end state cut sets by highlighting the end state and selecting **View Cut Sets** option. Compare your results with those shown. Note that each cut set contains S-DGN-FS-B or S-DGN-FR-B.


Cut Sets for ES-WITH-DGB (ES Cut Sets)

Project: DEMO-ADV - DEMO Project for Advanced SAPHIRE training
Project Folder: C:\Work\NRC Training\SAPHIRE Advanced Manual 2015\demo-adv\
Model Type: RANDOM

Current Case

Expand All | Show MT | Show Phase

Original

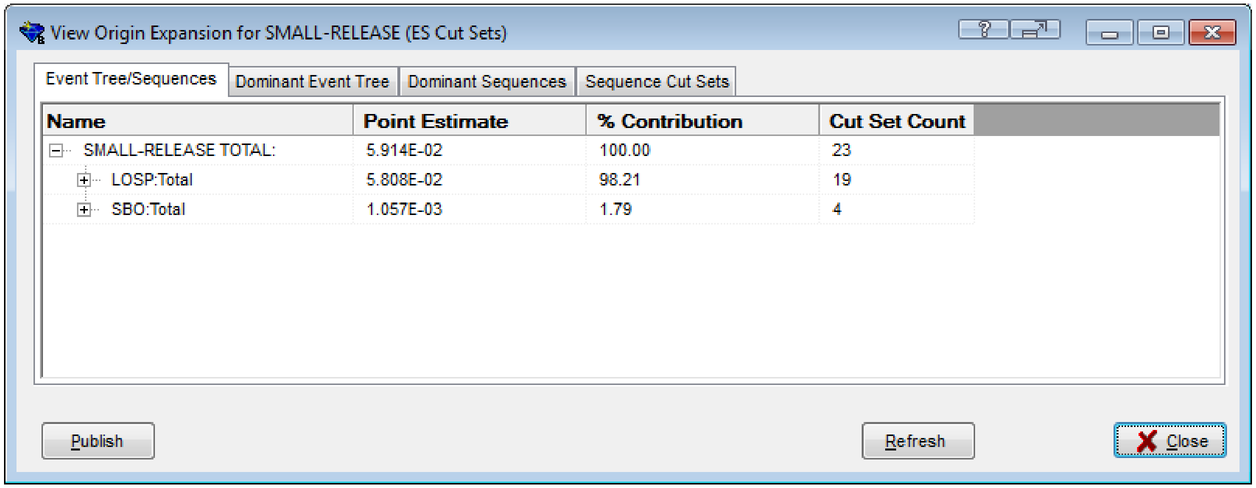
#	Cases	Prob/Freq	Total %	Cut Sets
		2.794E-6	100	Displaying 17 Cut Sets. (17 Original)
1	C	1.340E-6	47.96	SBO,/RDG-A,S-DGN-FR-A,S-DGN-FR-B
2	C	4.466E-7	15.99	SBO,RDG-A,S-DGN-FR-A,S-DGN-FR-B
3	C	2.536E-7	9.08	SBO,/RDG-A,S-DGN-FR-B,S-DGN-FS-A
4	C	2.536E-7	9.08	SBO,/RDG-A,S-DGN-FR-A,S-DGN-FS-B
5	C	8.453E-8	3.03	SBO,RDG-A,S-DGN-FR-B,S-DGN-FS-A
6	C	8.453E-8	3.03	SBO,RDG-A,S-DGN-FR-A,S-DGN-FS-B
7	C	7.608E-8	2.72	SBO,E-PMP-FS-A,/RDG-A,S-DGN-FR-B
8	C	6.340E-8	2.27	SBO,E-MOV-CC-A,/RDG-A,S-DGN-FR-B
9	C	4.800E-8	1.72	SBO,/RDG-A,S-DGN-FS-A,S-DGN-FS-B
10	C	3.195E-8	1.14	SBO,E-PMP-FR-A,/RDG-A,S-DGN-FR-B
11	C	2.536E-8	0.91	SBO,E-PMP-FS-A,RDG-A,S-DGN-FR-B
12	C	2.113E-8	0.76	SBO,E-MOV-CC-A,RDG-A,S-DGN-FR-B
13	C	1.600E-8	0.57	SBO,RDG-A,S-DGN-FS-A,S-DGN-FS-B
14	C	1.440E-8	0.52	SBO,E-PMP-FS-A,/RDG-A,S-DGN-FS-B
15	C	1.268E-8	0.45	SBO,E-MOV-CC-1,/RDG-A,S-DGN-FR-B
16	C	1.200E-8	0.43	SBO,E-MOV-CC-A,/RDG-A,S-DGN-FS-B
17	C	1.065E-8	0.38	SBO,E-PMP-FR-A,RDG-A,S-DGN-FR-B

Show Origin

Slice | Invert | Explore Origin | Publish | Save to End State | Close

13. Given the cut sets that were grouped into the end state SMALL-RELEASE, select the **Explore Origin** 

option.



14. Preview each tab:
    Dominant Event – Lists the event trees sorted on highest frequency
    Dominant Sequences – Lists the event tree accident sequences sorted by highest frequency
    Sequence Cut Sets – Provides a list of the cut sets for each dominant sequence.

| 7 | Solving Fault Tree Cut Sets

Workshop Objectives

The Solving Fault Tree Cut Sets workshop accompanies Section 7 in the Advanced SAPHIRE training manual. The workshop objective is to practice using different truncation options for solving fault tree cut sets.

Workshop Instructions

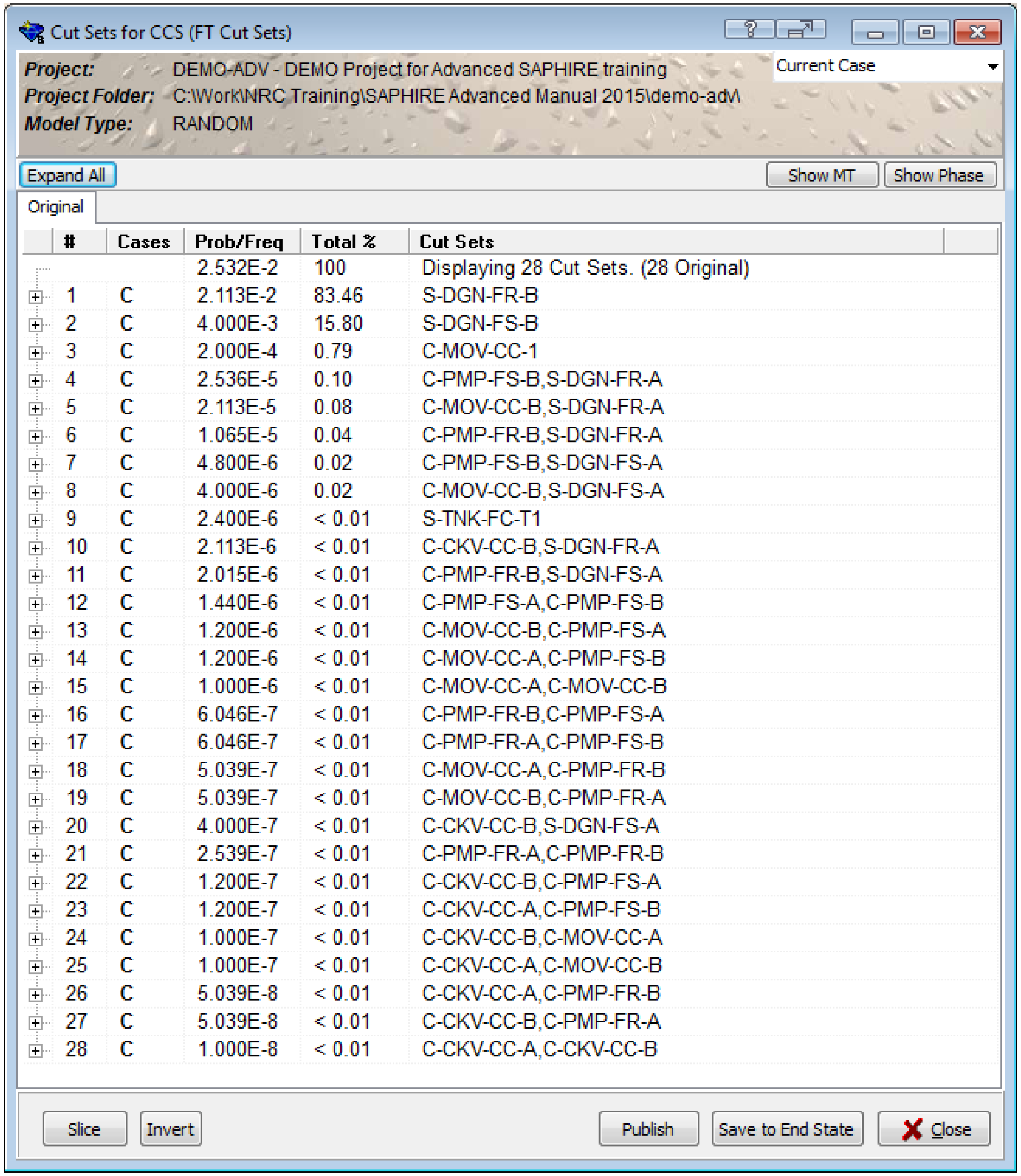
Using the DEMO-ADV project, fault tree cut sets will be solved for the CCS fault tree using a variety of truncation options.

1. Open the DEMO-ADV project and select the CCS fault tree.

2. Solve the CCS fault tree cut sets using the truncation options described below. View the CCS cut set results via one of the **View Cut Sets** options.

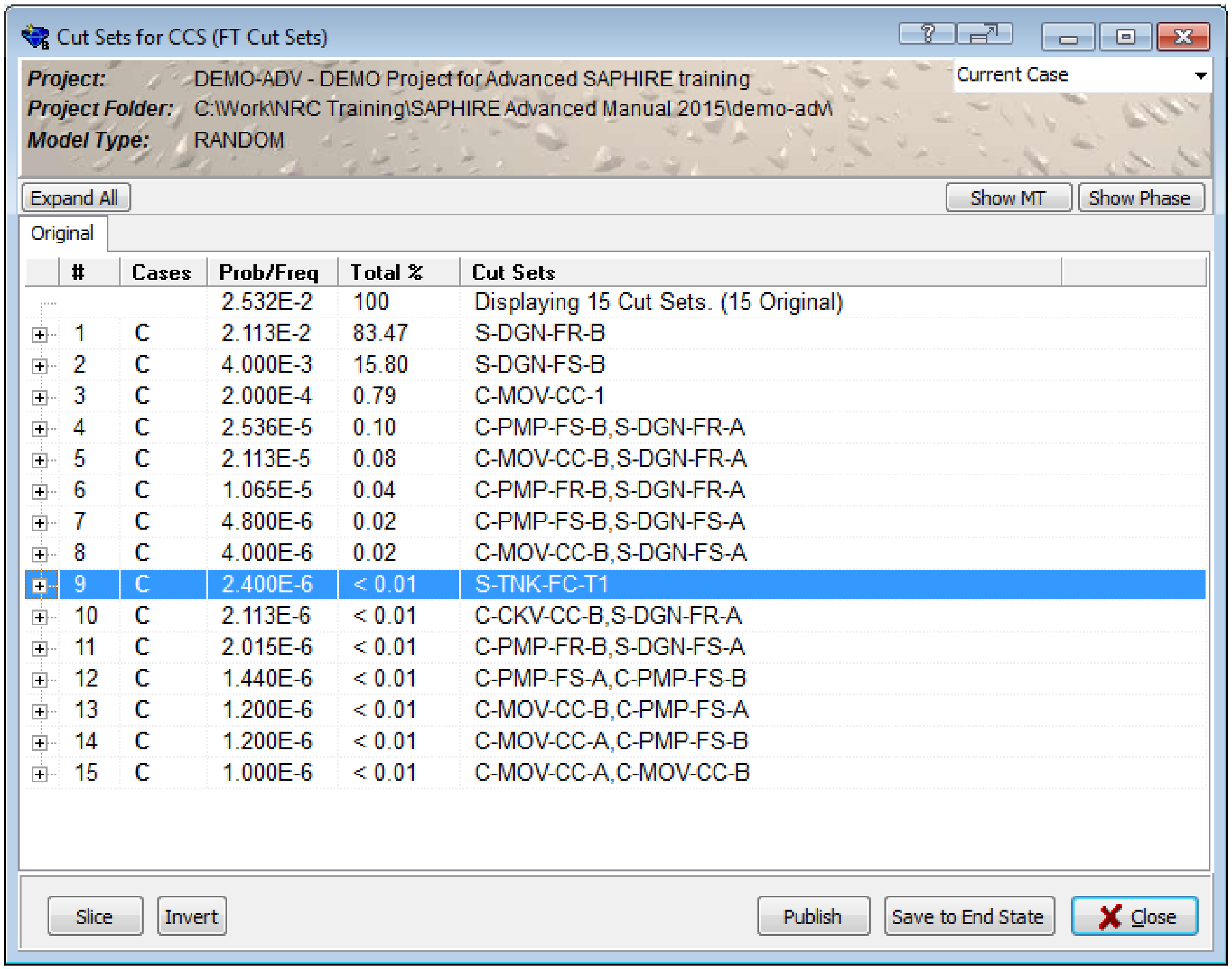
CCS Fault Tree Solve Options
No truncation
Probability truncation of 1E-6
Probability truncation of 1E-8 and Size cutoff of 1
Starting Gate Name of “CCS-TRAINS” and Probability truncation of 1E-8

No Truncation:



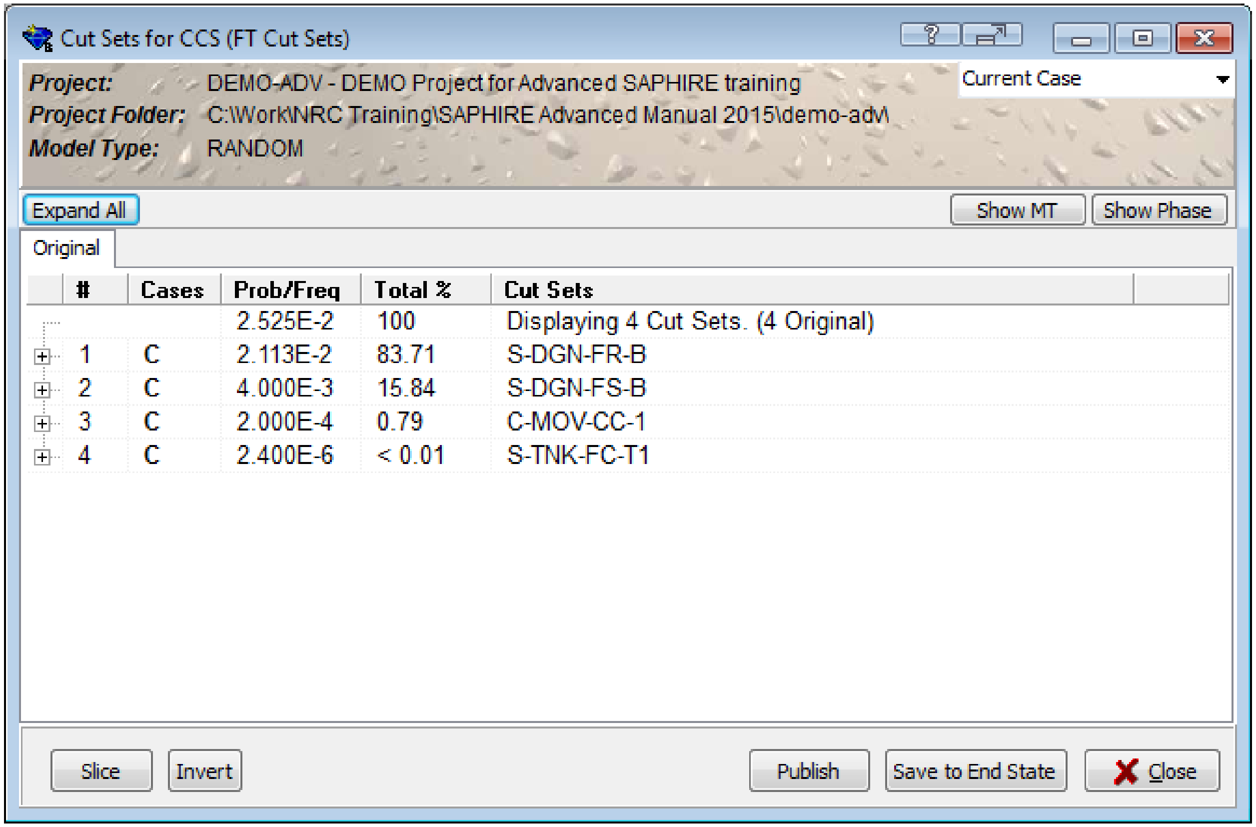
#	Cases	Prob/Freq	Total %	Cut Sets
		2.532E-2	100	Displaying 28 Cut Sets. (28 Original)
1	C	2.113E-2	83.46	S-DGN-FR-B
2	C	4.000E-3	15.80	S-DGN-FS-B
3	C	2.000E-4	0.79	C-MOV-CC-1
4	C	2.536E-5	0.10	C-PMP-FS-B,S-DGN-FR-A
5	C	2.113E-5	0.08	C-MOV-CC-B,S-DGN-FR-A
6	C	1.065E-5	0.04	C-PMP-FR-B,S-DGN-FR-A
7	C	4.800E-6	0.02	C-PMP-FS-B,S-DGN-FS-A
8	C	4.000E-6	0.02	C-MOV-CC-B,S-DGN-FS-A
9	C	2.400E-6	< 0.01	S-TNK-FC-T1
10	C	2.113E-6	< 0.01	C-CKV-CC-B,S-DGN-FR-A
11	C	2.015E-6	< 0.01	C-PMP-FR-B,S-DGN-FS-A
12	C	1.440E-6	< 0.01	C-PMP-FS-A,C-PMP-FS-B
13	C	1.200E-6	< 0.01	C-MOV-CC-B,C-PMP-FS-A
14	C	1.200E-6	< 0.01	C-MOV-CC-A,C-PMP-FS-B
15	C	1.000E-6	< 0.01	C-MOV-CC-A,C-MOV-CC-B
16	C	6.046E-7	< 0.01	C-PMP-FR-B,C-PMP-FS-A
17	C	6.046E-7	< 0.01	C-PMP-FR-A,C-PMP-FS-B
18	C	5.039E-7	< 0.01	C-MOV-CC-A,C-PMP-FR-B
19	C	5.039E-7	< 0.01	C-MOV-CC-B,C-PMP-FR-A
20	C	4.000E-7	< 0.01	C-CKV-CC-B,S-DGN-FS-A
21	C	2.539E-7	< 0.01	C-PMP-FR-A,C-PMP-FR-B
22	C	1.200E-7	< 0.01	C-CKV-CC-B,C-PMP-FS-A
23	C	1.200E-7	< 0.01	C-CKV-CC-A,C-PMP-FS-B
24	C	1.000E-7	< 0.01	C-CKV-CC-B,C-MOV-CC-A
25	C	1.000E-7	< 0.01	C-CKV-CC-A,C-MOV-CC-B
26	C	5.039E-8	< 0.01	C-CKV-CC-A,C-PMP-FR-B
27	C	5.039E-8	< 0.01	C-CKV-CC-B,C-PMP-FR-A
28	C	1.000E-8	< 0.01	C-CKV-CC-A,C-CKV-CC-B

Probability Truncation at 1.0E-06:

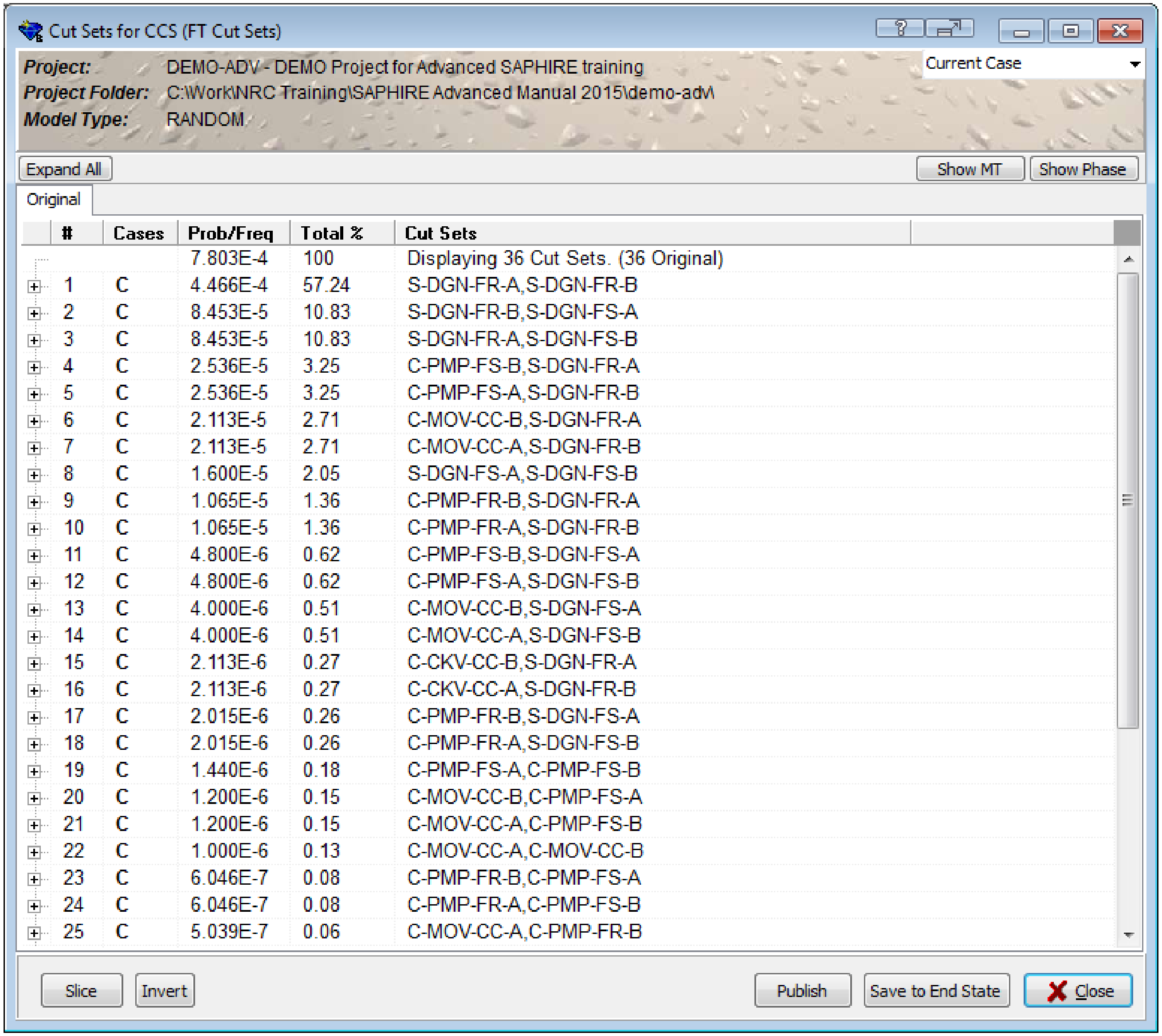


#	Cases	Prob/Freq	Total %	Cut Sets
		2.532E-2	100	Displaying 15 Cut Sets. (15 Original)
1	C	2.113E-2	83.47	S-DGN-FR-B
2	C	4.000E-3	15.80	S-DGN-FS-B
3	C	2.000E-4	0.79	C-MOV-CC-1
4	C	2.536E-5	0.10	C-PMP-FS-B,S-DGN-FR-A
5	C	2.113E-5	0.08	C-MOV-CC-B,S-DGN-FR-A
6	C	1.065E-5	0.04	C-PMP-FR-B,S-DGN-FR-A
7	C	4.800E-6	0.02	C-PMP-FS-B,S-DGN-FS-A
8	C	4.000E-6	0.02	C-MOV-CC-B,S-DGN-FS-A
9	C	2.400E-6	< 0.01	S-TNK-FC-T1
10	C	2.113E-6	< 0.01	C-CKV-CC-B,S-DGN-FR-A
11	C	2.015E-6	< 0.01	C-PMP-FR-B,S-DGN-FS-A
12	C	1.440E-6	< 0.01	C-PMP-FS-A,C-PMP-FS-B
13	C	1.200E-6	< 0.01	C-MOV-CC-B,C-PMP-FS-A
14	C	1.200E-6	< 0.01	C-MOV-CC-A,C-PMP-FS-B
15	C	1.000E-6	< 0.01	C-MOV-CC-A,C-MOV-CC-B

Probability Truncation 1.0E-08 and Cut Set Size of 1:



Starting Gate Name of “CCS-TRAINS” and Probability truncation of 1E-8:

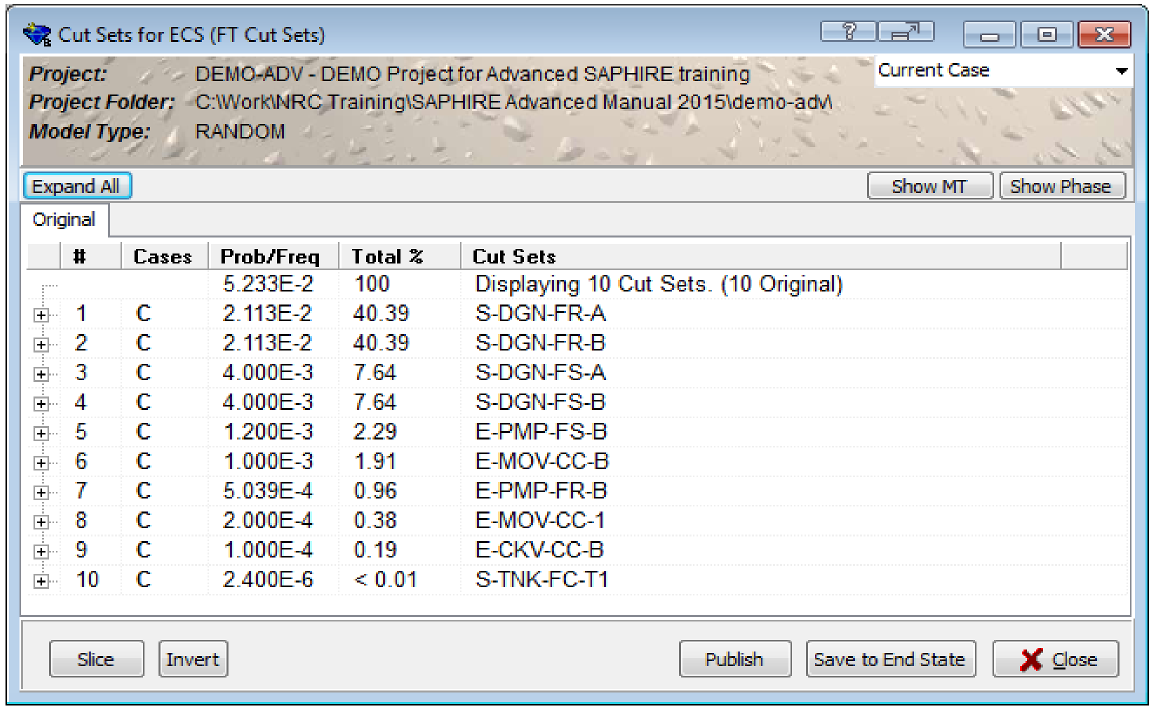


#	Cases	Prob/Freq	Total %	Cut Sets
		7.803E-4	100	Displaying 36 Cut Sets. (36 Original)
1	C	4.466E-4	57.24	S-DGN-FR-A,S-DGN-FR-B
2	C	8.453E-5	10.83	S-DGN-FR-B,S-DGN-FS-A
3	C	8.453E-5	10.83	S-DGN-FR-A,S-DGN-FS-B
4	C	2.536E-5	3.25	C-PMP-FS-B,S-DGN-FR-A
5	C	2.536E-5	3.25	C-PMP-FS-A,S-DGN-FR-B
6	C	2.113E-5	2.71	C-MOV-CC-B,S-DGN-FR-A
7	C	2.113E-5	2.71	C-MOV-CC-A,S-DGN-FR-B
8	C	1.600E-5	2.05	S-DGN-FS-A,S-DGN-FS-B
9	C	1.065E-5	1.36	C-PMP-FR-B,S-DGN-FR-A
10	C	1.065E-5	1.36	C-PMP-FR-A,S-DGN-FR-B
11	C	4.800E-6	0.62	C-PMP-FS-B,S-DGN-FS-A
12	C	4.800E-6	0.62	C-PMP-FS-A,S-DGN-FS-B
13	C	4.000E-6	0.51	C-MOV-CC-B,S-DGN-FS-A
14	C	4.000E-6	0.51	C-MOV-CC-A,S-DGN-FS-B
15	C	2.113E-6	0.27	C-CKV-CC-B,S-DGN-FR-A
16	C	2.113E-6	0.27	C-CKV-CC-A,S-DGN-FR-B
17	C	2.015E-6	0.26	C-PMP-FR-B,S-DGN-FS-A
18	C	2.015E-6	0.26	C-PMP-FR-A,S-DGN-FS-B
19	C	1.440E-6	0.18	C-PMP-FS-A,C-PMP-FS-B
20	C	1.200E-6	0.15	C-MOV-CC-B,C-PMP-FS-A
21	C	1.200E-6	0.15	C-MOV-CC-A,C-PMP-FS-B
22	C	1.000E-6	0.13	C-MOV-CC-A,C-MOV-CC-B
23	C	6.046E-7	0.08	C-PMP-FR-B,C-PMP-FS-A
24	C	6.046E-7	0.08	C-PMP-FR-A,C-PMP-FS-B
25	C	5.039E-7	0.06	C-MOV-CC-A,C-PMP-FR-B

3. Now we will make a flag set (refer to workshop 2 as needed).
   - Make a flag set called ECS-PUMP-FLAG.
   - Add the event E-PMP-FS-A to the flag set and set the Failure Model to a TRUE house event

4. Assign the flag set to fault tree ECS by opening ECS Fault Tree editor, then select **Edit → Properties** from the fault tree main menu. Use the **Default Flag Set** pull-down menu to set the flag set assigned to ECS as **ECS-PUMP-FLAG** or highlight the ECS fault tree, right click, and select **Edit Properties**. Click OK to save and also save the changes when exiting the fault tree editor.

5. Solve cut sets for the ECS fault tree (turn off all truncation). Compare your results to those shown here.



#	Cases	Prob/Freq	Total %	Cut Sets
		5.233E-2	100	Displaying 10 Cut Sets. (10 Original)
1	C	2.113E-2	40.39	S-DGN-FR-A
2	C	2.113E-2	40.39	S-DGN-FR-B
3	C	4.000E-3	7.64	S-DGN-FS-A
4	C	4.000E-3	7.64	S-DGN-FS-B
5	C	1.200E-3	2.29	E-PMP-FS-B
6	C	1.000E-3	1.91	E-MOV-CC-B
7	C	5.039E-4	0.96	E-PMP-FR-B
8	C	2.000E-4	0.38	E-MOV-CC-1
9	C	1.000E-4	0.19	E-CKV-CC-B
10	C	2.400E-6	< 0.01	S-TNK-FC-T1

6. Restore ECS by removing the flag set by repeating step 4 and setting the Default Flag Set to set to the blank selection on the top of the pull-down menu.

| 8 | Quantifying Fault Tree Cut Sets

Workshop Objectives

The Quantifying Fault Tree Cut Sets workshop accompanies Section 8 in the Advanced SAPHIRE training manual. The workshop objective is to gain experience using the Min-Max analysis option.

Workshop Instructions

Using the DEMO-ADV project, all motor-operated valve basic event probabilities will be increased and then the ECS fault tree will be reevaluated using the Min-Max option. The minimal cut set upperbound results will be compared to the Min-Max results.

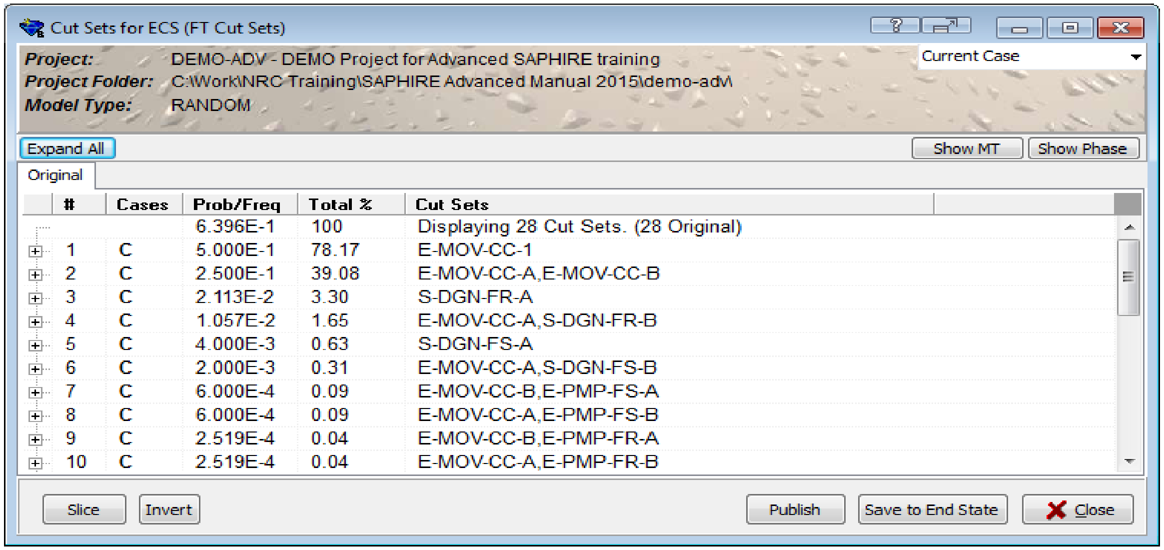
1. Make a change set called MOD-MOVS by selecting the **View → Change Sets** or **Project → Change Sets** option. Double click New Change Set, name the change set MOD-MOVs and provide a description, click the **Class** Tab, check the Name Mask and enter the search criteria as:

   Event Attribute → Name = "*-MOV*" Probability/Frequency = 0.5

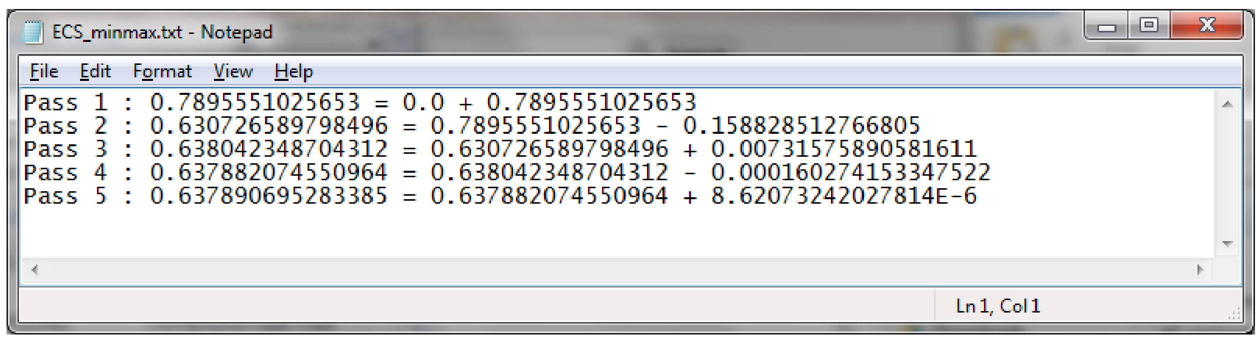
2. Mark *just* the MOD-MOVS change set (by placing a check in its checkbox). If you have more than one change set, this change set should be the only one with a number to the left of its name.

3. Solve the ECS fault tree (turn off all truncation).

4. The cut sets, with minimal cut set upper-bound value, are shown here (we are interested in the top Prob/Freq sum):



5. Now, perform the Min-Max calculation by solving ECS with the **Quantification Method** set to Min/Max and the pass count set to 5.



The result for the Min-Max quantification is 6.379E-1. Note that (1) the minimal cut set upper-bound is higher than the exact quantification and (2) the min-cut approximation is close to the exact even though the overall system failure probability is quite high.

| 9 | Solving Event Tree Cut Sets

Workshop Objectives

The Solving Event Tree Cut Sets workshop accompanies Section 9 in the Advanced SAPHIRE training manual. The workshop objective is to generate event tree sequence cut sets using both Process Flags and Flag Sets.

Workshop Instructions

Using the DEMO-ADV project, top event process flags will be modified and applied to the LOSP sequences. The LOSP event tree sequences will then be solved to generate the new sequence cut sets.

Part I – Change Sets

1. In the DEMO-ADV project, double click on **New change set…** under the **Change Sets** list panel. Enter the Change Set name and description below.

Name	*Description*
SET-CCS-TO-I	Top event CCS has Process Flag set to I

2. To make a data change in a Change Set, view **All** basic events in the Basic Events list panel. Drag and drop the event **CCS** to the **Single** pane of the Change Set Editor.

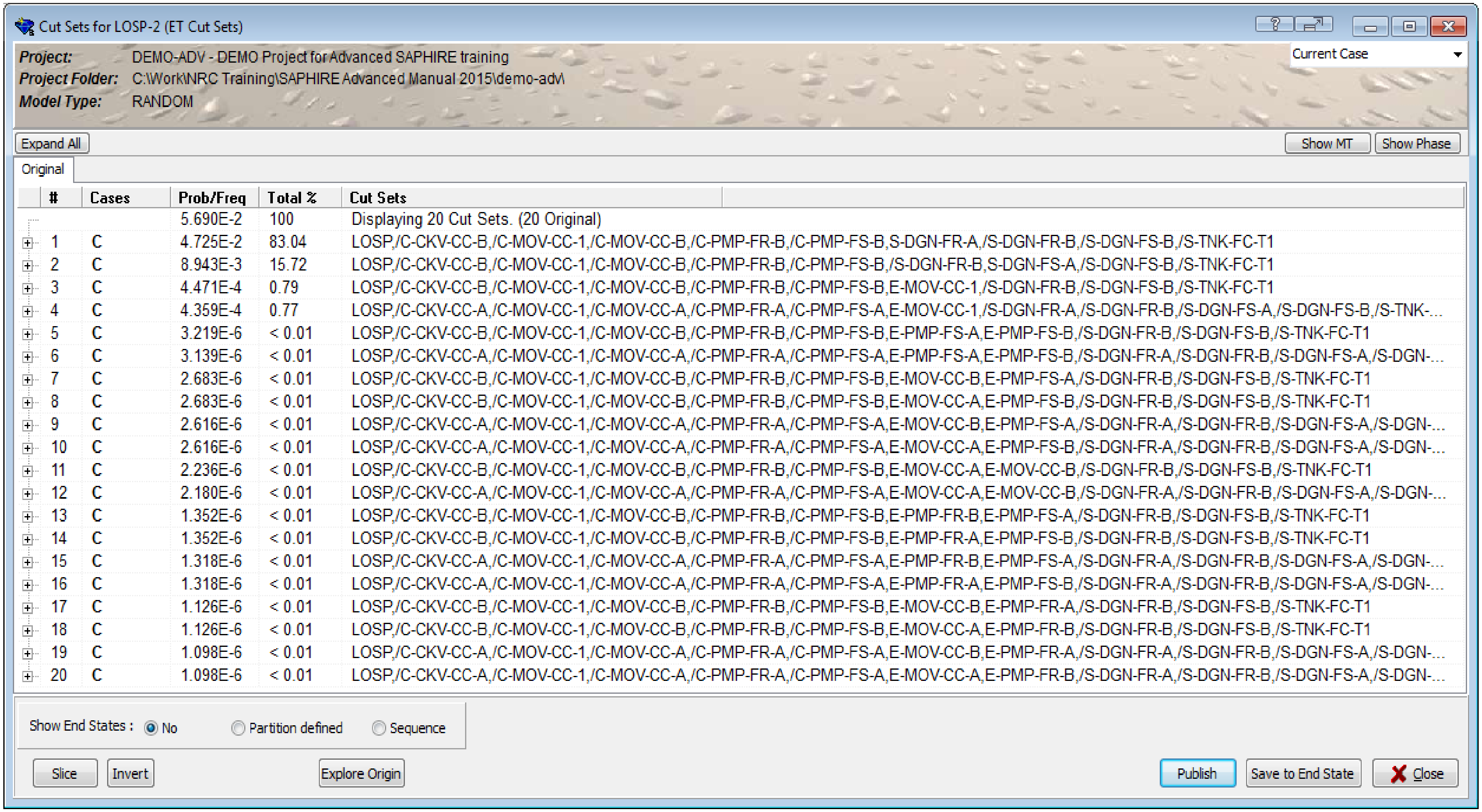
3. For the SET-CCS-TO-I Change Set that was just added, make the change indicated below.

Event to be changed	*Changes*
CCS	Process Flag set to "I"

4. "Unmark" any change sets that may be marked. Then, mark the change set named SET-CCS-TO-I.

5. Now, solve sequence cut sets for Sequence 2 of the LOSP event tree (with truncation of 1E-6).
   - Cut sets for Sequence 3 will not change since both systems (CCS and ECS) fail in this sequence.

6. View the sequence cut sets and compare to the following:



#	Cases	Prob/Freq	Total %	Cut Sets
		5.690E-2	100	Displaying 20 Cut Sets. (20 Original)
1	C	4.725E-2	83.04	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
2	C	8.943E-3	15.72	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,/S-DGN-FR-B,S-DGN-FS-A,/S-DGN-FS-B,/S-TNK-FC-T1
3	C	4.471E-4	0.79	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-1,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
4	C	4.359E-4	0.77	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-1,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-FS-B,/S-TNK-...
5	C	3.219E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-PMP-FS-A,E-PMP-FS-B,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
6	C	3.139E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-PMP-FS-A,E-PMP-FS-B,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
7	C	2.683E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-B,E-PMP-FS-A,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
8	C	2.683E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-A,E-PMP-FS-B,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
9	C	2.616E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-B,E-PMP-FS-A,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
10	C	2.616E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-A,E-PMP-FS-B,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
11	C	2.236E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-A,E-MOV-CC-B,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
12	C	2.180E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-A,E-MOV-CC-B,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
13	C	1.352E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-PMP-FR-B,E-PMP-FS-A,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
14	C	1.352E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-PMP-FR-A,E-PMP-FS-B,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
15	C	1.318E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-PMP-FR-B,E-PMP-FS-A,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
16	C	1.318E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-PMP-FR-A,E-PMP-FS-B,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
17	C	1.126E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-B,E-PMP-FR-A,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
18	C	1.126E-6	< 0.01	LOSP,/C-CKV-CC-B,/C-MOV-CC-1,/C-MOV-CC-B,/C-PMP-FR-B,/C-PMP-FS-B,E-MOV-CC-A,E-PMP-FR-B,/S-DGN-FR-B,/S-DGN-FS-B,/S-TNK-FC-T1
19	C	1.098E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-B,E-PMP-FR-A,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...
20	C	1.098E-6	< 0.01	LOSP,/C-CKV-CC-A,/C-MOV-CC-1,/C-MOV-CC-A,/C-PMP-FR-A,/C-PMP-FS-A,E-MOV-CC-A,E-PMP-FR-B,/S-DGN-FR-A,/S-DGN-FR-B,/S-DGN-FS-A,/S-DGN-...

Part II – Flag Sets

Using the DEMO-ADV project, the Flag Sets created in the Section 2 workshop (FLAG-SET-SBO1 and FLAG-SET-SBO2) will be assigned to the SBO event tree sequences. Then, sequence cut sets for SBO will be solved.

1. The two Flag Sets developed in the Section 2 workshop will be used in this workshop exercise. These two Flag Sets and the settings are shown below. (If these two Flag Sets do not exist, refer back to the Section 2 workshop and add the two Flag Sets to the database.)

Name	Events to be Changed	Changes
FLAG-SET-SBO1	S-DGN-FS-A S-DGN-FS-B	Calculation type set to “T”
FLAG-SET-SBO2	S-DGN-FS-B	Calculation type set to “T”

2. Assign a Flag Set to a sequence via the **Sequence Properties** menu:
   - Expand the sequences under the event tree in the event tree list panel
   - Highlight the desired event tree sequence
   - Right click on a highlighted sequence and select **Edit Properties**
   - The Flag Set name is entered in the field labeled **Default Flag Set** pull-down menu for the selected sequence
   - OR write a “Link Event Tree” rule to assign the flag sets and then link the event tree.

3. Assign the Flag Sets to the SBO sequences as shown:

Event Tree	*Sequence*	*Flag Set Name*
SBO	2	FLAG-SET-SBO2
SBO	3	FLAG-SET-SBO2
SBO	4	FLAG-SET-SBO1
SBO	5	FLAG-SET-SBO1
SBO	6	FLAG-SET-SBO1

4. Solve all of the SBO event tree sequences by highlighting the sequences (or just SBO), right clicking to invoke the menu and selecting Solve (with truncation turned off).

5. View the cut sets and compare them against the results below generated through main menu **Publish → Event Tree Report→ Cut Sets (by Sequence)** Rich Text Format (RTF):

Note: Cut sets that contribute >= 1.0% are reportedSequence/CS#	**Prob./Freq.**	**Total %**	**Cut Sets**
SBO - 2	0.000E+0	100%	Displaying 0 of 1 cut sets
SBO - 3	8.405E-5	100%	Displaying 5 of 8 cut sets
1	6.340E-5	75.43%	SBO, /RDG-A, S-DGN-FR-A
2	1.200E-5	14.28%	SBO, /RDG-A, S-DGN-FS-A
3	3.600E-6	4.28%	SBO, E-PMP-FS-A, /RDG-A
4	3.000E-6	3.57%	SBO, E-MOV-CC-A, /RDG-A
5	1.512E-6	1.80%	SBO, E-PMP-FR-A, /RDG-A
SBO - 4	0.000E+0	100%	Displaying 0 of 1 cut sets
SBO - 5	0.000E+0	100%	Displaying 0 of 1 cut sets
SBO - 6	1.000E-3	100%	Displaying 1 of 1 cut sets
1	1.000E-3	100%	SBO, RDG-A

NOTES

| 10 | Editing Cut Sets

Workshop Objectives

The Editing Cut Sets workshop accompanies Section 10 in the Advanced SAPHIRE training manual. The workshop objective is to perform cut set editing by using the “edit cut set” option

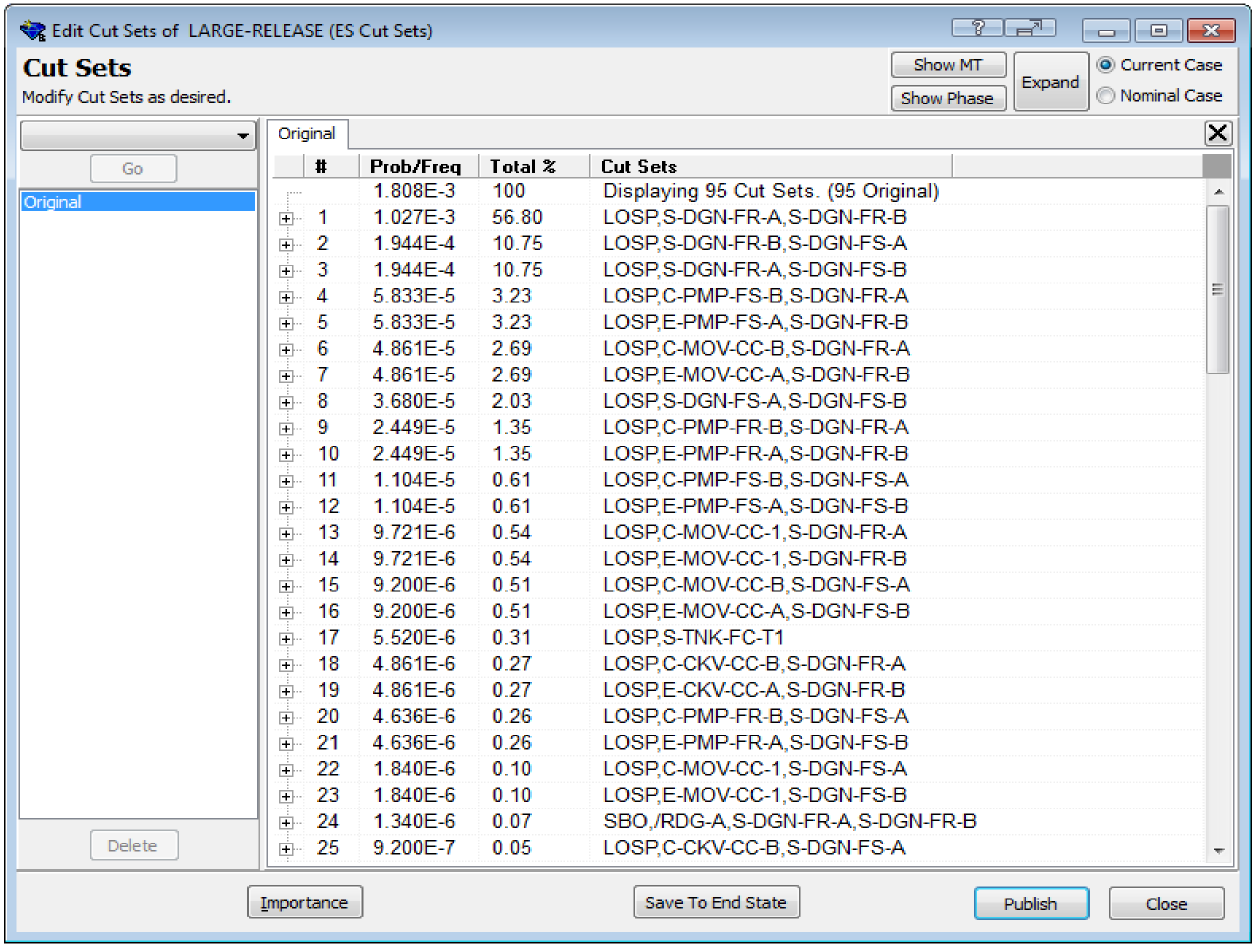
Workshop Instructions

The DEMO-ADV project will be used to illustrate the use of the “Edit Cut Set” option.

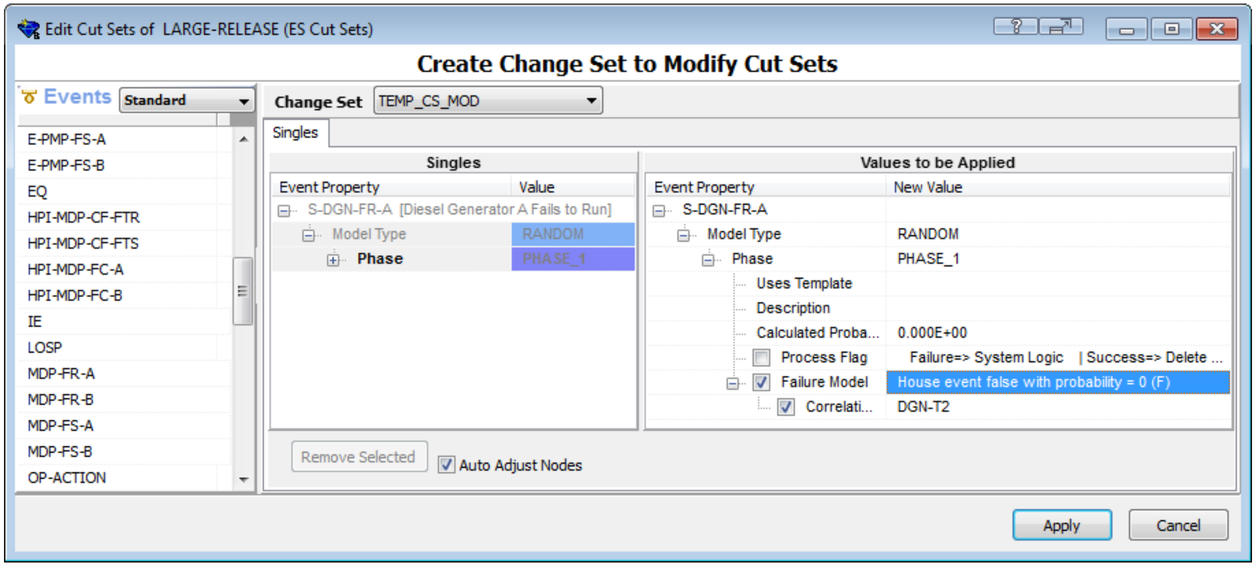
1. **Solve** for sequence cut sets, both the LOSP and SBO event tree. Make sure that no recovery rules or flag sets are assigned to these event trees. Solve the sequence cut sets with a 1E-10 truncation.

2. **Gather** the LARGE-RELEASE end state with no truncation. There should be 95 cut sets and a frequency of 1.808E-03/yr.

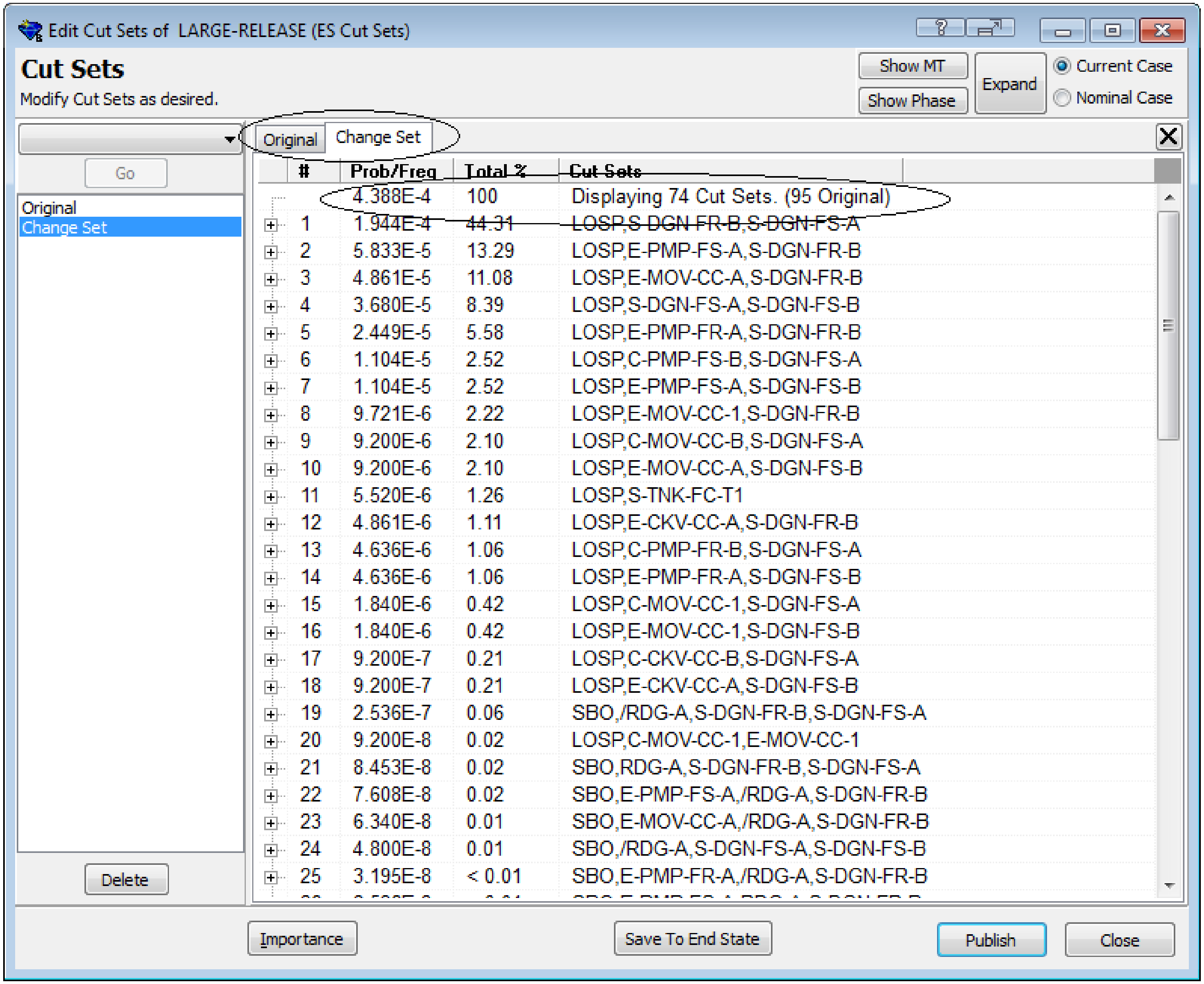
3. With the LARGE-RELEASE end state highlighted, right mouse click and select **Edit Cut Sets**.



3. Apply a Change Set to these cut sets. Under the Modify Cut Sets as desired drop down option, select Apply Change Set and set S-DGN-FR-A to “FALSE” and apply this change set.



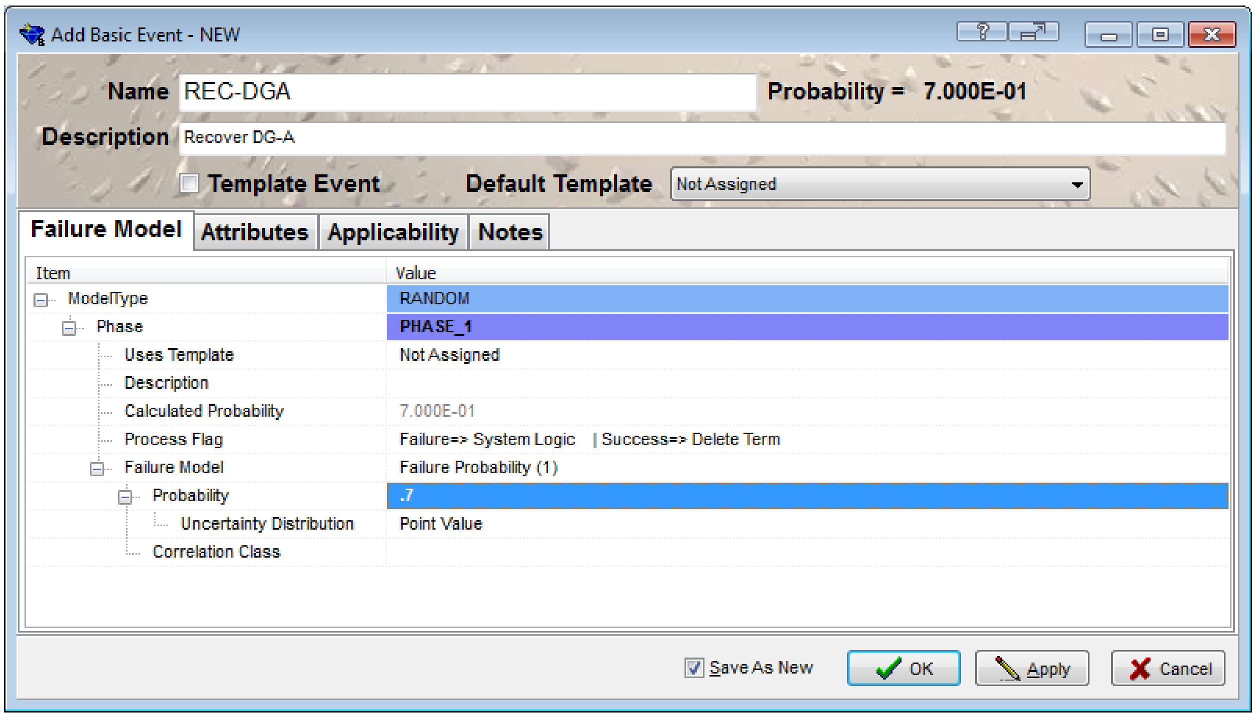
4. Select Apply. SAPHIRE will apply this change set to the cut sets and then re-calculate the new frequency.

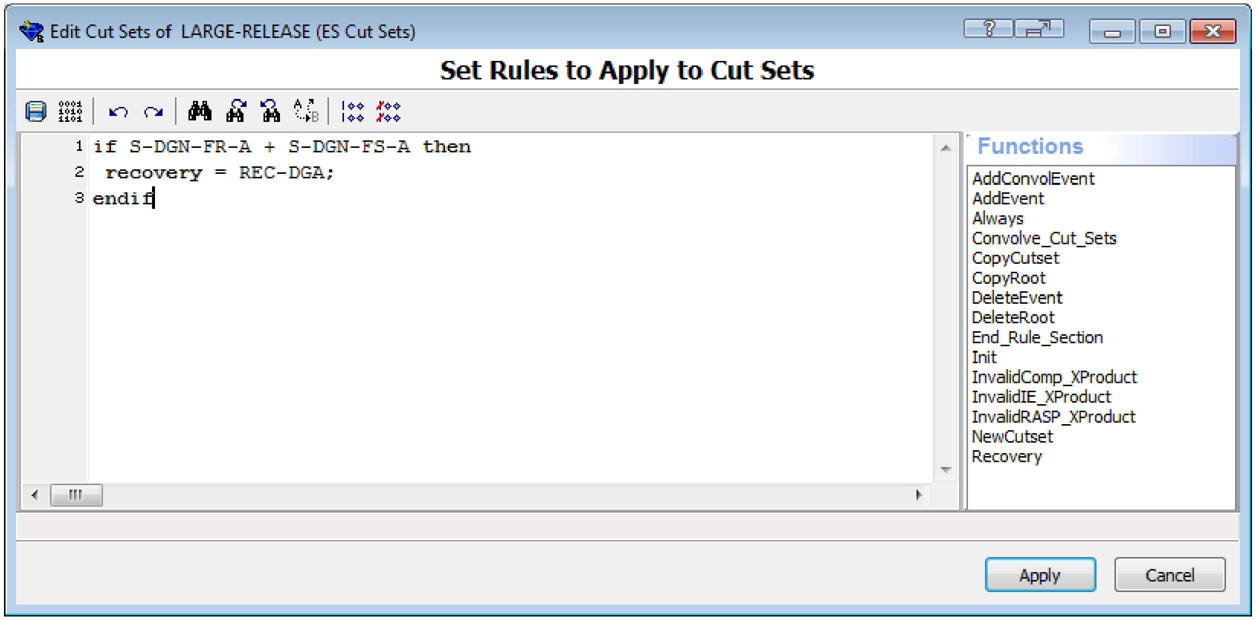


Edit Cut Sets of LARGE-RELEASE (ES Cut Sets)
Cut Sets
Modify Cut Sets as desired.
Show MT
Show Phase
Expand
Current Case
Nominal Case
Go
Original
Change Set
Original
Change Set
Prob/Freq Total % Cut Sets
4.388E-4 100 Displaying 74 Cut Sets. (95 Original)
~~1 1.944E-4 44.31 LOSP,S-DGN-FR-B,S-DGN-FS-A~~
2 5.833E-5 13.29 LOSP,E-PMP-FS-A,S-DGN-FR-B
3 4.861E-5 11.08 LOSP,E-MOV-CC-A,S-DGN-FR-B
4 3.680E-5 8.39 LOSP,S-DGN-FS-A,S-DGN-FS-B
5 2.449E-5 5.58 LOSP,E-PMP-FR-A,S-DGN-FR-B
6 1.104E-5 2.52 LOSP,C-PMP-FS-B,S-DGN-FS-A
7 1.104E-5 2.52 LOSP,E-PMP-FS-A,S-DGN-FS-B
8 9.721E-6 2.22 LOSP,E-MOV-CC-1,S-DGN-FR-B
9 9.200E-6 2.10 LOSP,C-MOV-CC-B,S-DGN-FS-A
10 9.200E-6 2.10 LOSP,E-MOV-CC-A,S-DGN-FS-B
11 5.520E-6 1.26 LOSP,S-TNK-FC-T1
12 4.861E-6 1.11 LOSP,E-CKV-CC-A,S-DGN-FR-B
13 4.636E-6 1.06 LOSP,C-PMP-FR-B,S-DGN-FS-A
14 4.636E-6 1.06 LOSP,E-PMP-FR-A,S-DGN-FS-B
15 1.840E-6 0.42 LOSP,C-MOV-CC-1,S-DGN-FS-A
16 1.840E-6 0.42 LOSP,E-MOV-CC-1,S-DGN-FS-B
17 9.200E-7 0.21 LOSP,C-CKV-CC-B,S-DGN-FS-A
18 9.200E-7 0.21 LOSP,E-CKV-CC-A,S-DGN-FS-B
19 2.536E-7 0.06 SBO,/RDG-A,S-DGN-FR-B,S-DGN-FS-A
20 9.200E-8 0.02 LOSP,C-MOV-CC-1,E-MOV-CC-1
21 8.453E-8 0.02 SBO,RDG-A,S-DGN-FR-B,S-DGN-FS-A
22 7.608E-8 0.02 SBO,E-PMP-FS-A,/RDG-A,S-DGN-FR-B
23 6.340E-8 0.01 SBO,E-MOV-CC-A,/RDG-A,S-DGN-FR-B
24 4.800E-8 0.01 SBO,/RDG-A,S-DGN-FS-A,S-DGN-FS-B
25 3.195E-8 < 0.01 SBO,E-PMP-FR-A,/RDG-A,S-DGN-FR-B
Delete
Importance
Save To End State
Publish
Close

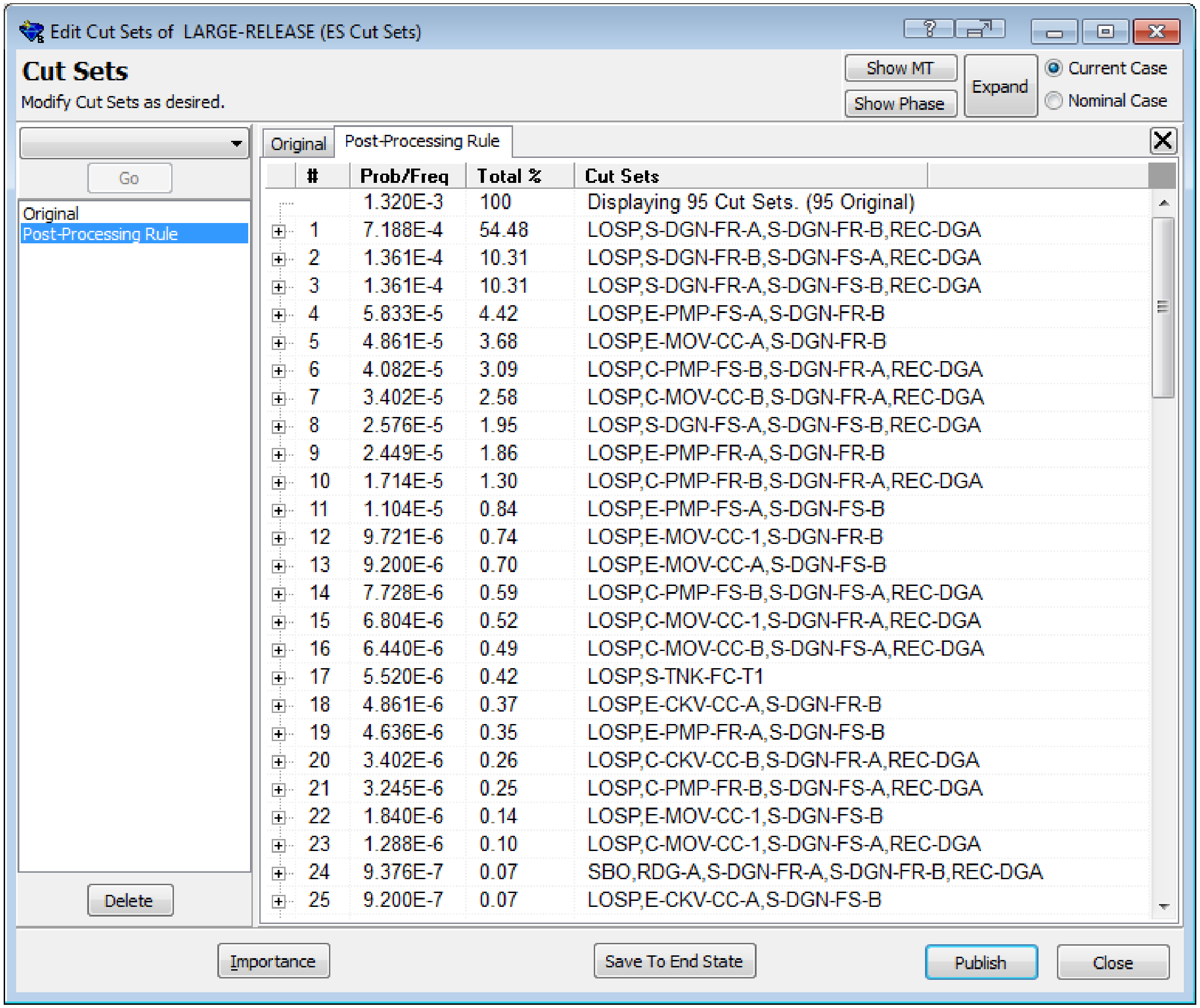
5. Click the  option at the end of the Tab to erase the change set Tab. This way a new Edit can be performed on the original cut sets.

6. Click the drop down option and select Post-Processing Rules. The option will open up the form to allow the analyst to create a rule that will be applied to the cut sets that are displayed. Create the following rule (the recovery basic event will have to be added to the project):





7. Apply the post-processing rule to the cut sets.



#	Prob/Freq	Total %	Cut Sets
	1.320E-3	100	Displaying 95 Cut Sets. (95 Original)
1	7.188E-4	54.48	LOSP,S-DGN-FR-A,S-DGN-FR-B,REC-DGA
2	1.361E-4	10.31	LOSP,S-DGN-FR-B,S-DGN-FS-A,REC-DGA
3	1.361E-4	10.31	LOSP,S-DGN-FR-A,S-DGN-FS-B,REC-DGA
4	5.833E-5	4.42	LOSP,E-PMP-FS-A,S-DGN-FR-B
5	4.861E-5	3.68	LOSP,E-MOV-CC-A,S-DGN-FR-B
6	4.082E-5	3.09	LOSP,C-PMP-FS-B,S-DGN-FR-A,REC-DGA
7	3.402E-5	2.58	LOSP,C-MOV-CC-B,S-DGN-FR-A,REC-DGA
8	2.576E-5	1.95	LOSP,S-DGN-FS-A,S-DGN-FS-B,REC-DGA
9	2.449E-5	1.86	LOSP,E-PMP-FR-A,S-DGN-FR-B
10	1.714E-5	1.30	LOSP,C-PMP-FR-B,S-DGN-FR-A,REC-DGA
11	1.104E-5	0.84	LOSP,E-PMP-FS-A,S-DGN-FS-B
12	9.721E-6	0.74	LOSP,E-MOV-CC-1,S-DGN-FR-B
13	9.200E-6	0.70	LOSP,E-MOV-CC-A,S-DGN-FS-B
14	7.728E-6	0.59	LOSP,C-PMP-FS-B,S-DGN-FS-A,REC-DGA
15	6.804E-6	0.52	LOSP,C-MOV-CC-1,S-DGN-FR-A,REC-DGA
16	6.440E-6	0.49	LOSP,C-MOV-CC-B,S-DGN-FS-A,REC-DGA
17	5.520E-6	0.42	LOSP,S-TNK-FC-T1
18	4.861E-6	0.37	LOSP,E-CKV-CC-A,S-DGN-FR-B
19	4.636E-6	0.35	LOSP,E-PMP-FR-A,S-DGN-FS-B
20	3.402E-6	0.26	LOSP,C-CKV-CC-B,S-DGN-FR-A,REC-DGA
21	3.245E-6	0.25	LOSP,C-PMP-FR-B,S-DGN-FS-A,REC-DGA
22	1.840E-6	0.14	LOSP,E-MOV-CC-1,S-DGN-FS-B
23	1.288E-6	0.10	LOSP,C-MOV-CC-1,S-DGN-FS-A,REC-DGA
24	9.376E-7	0.07	SBO,RDG-A,S-DGN-FR-A,S-DGN-FR-B,REC-DGA
25	9.200E-7	0.07	LOSP,E-CKV-CC-A,S-DGN-FS-B

NOTES

| 11 | General Analysis for Sensitivity

Workshop Objectives

The General Analysis workshop accompanies Section 11 in the Advanced SAPHIRE training manual. The workshop objectives are to gain experience using General Analysis Workspace to perform sensitivity evaluations to a project.

Workshop Instructions

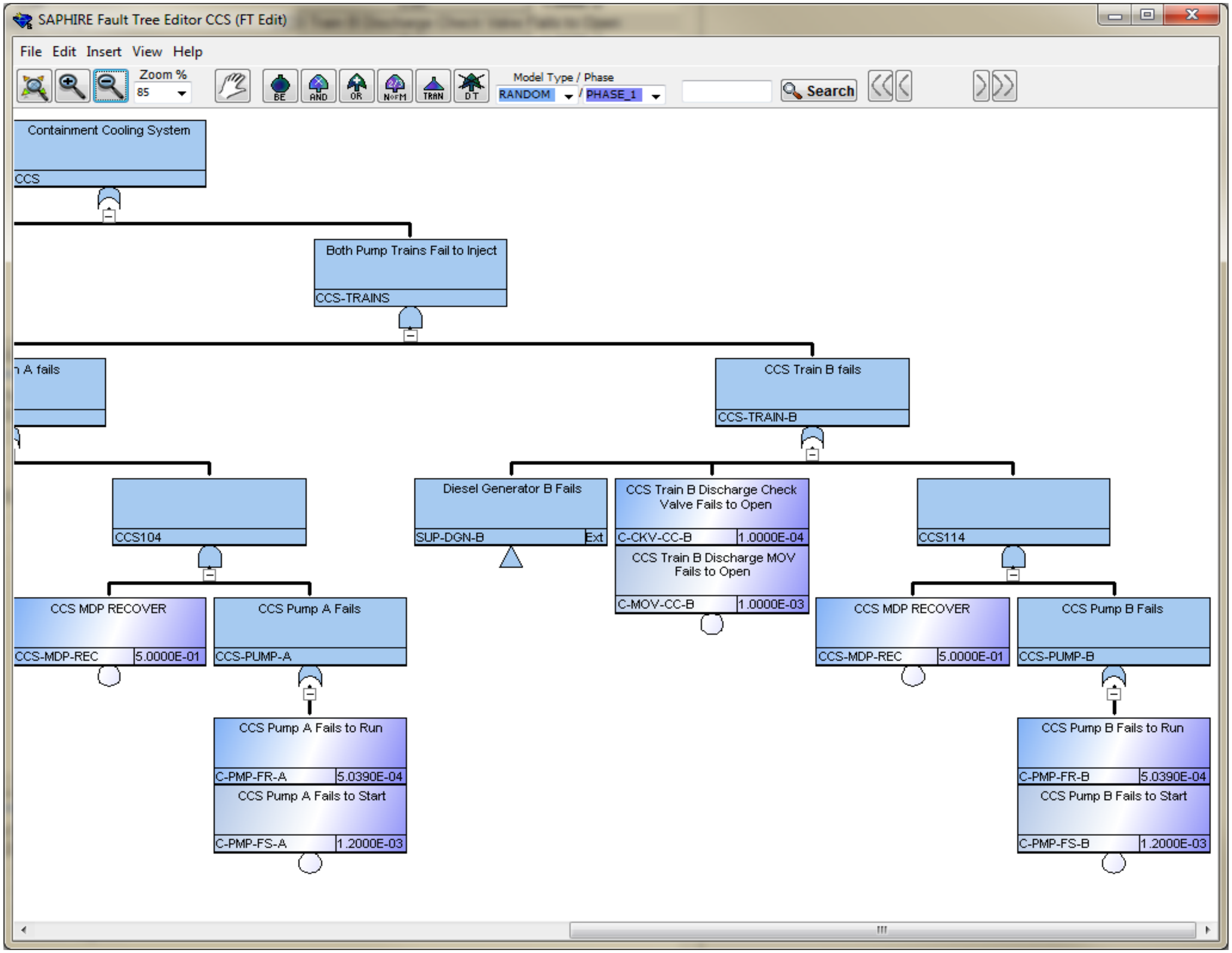
For this workshop the General Analysis Workspace will be used to modify the DEMO-ADV project by adding an operator recovery action to the CCS fault tree, modify the LOSP event tree and adjust the ECS check valves failure probabilities.

1. Using the DEMO-ADV project, double click the New Analysis… option under the General Analysis option in the Workspace list panel. The whole project is copied into a temporary directory for modification and evaluation.

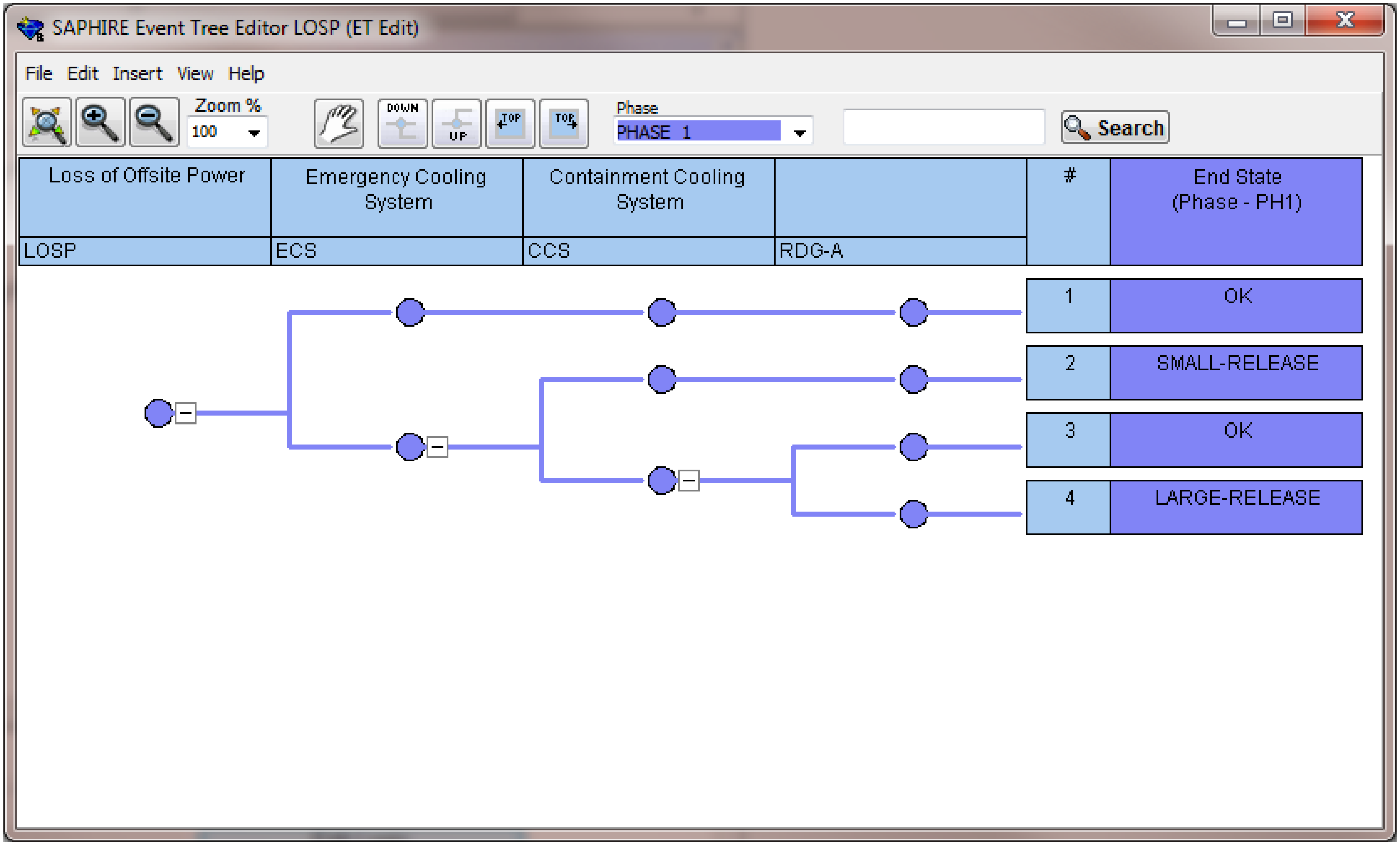
2. Select the logic models (CCS and ECS fault trees and LOSP event tree) and basic events (E-CKV-CC-A and E-CKV-CC-B) that will be modified for this sensitivity evaluation and then select **Next**.

3. Click the Edit button under the E-CKV-CC-A basic event and change its nominal probability to 1E-02, do the same for E-CKV-CC-B basic event .

4. Modify the CCS fault tree by adding a new basic event that is ANDed to the failure of the motor-driven pumps (both fails to start and run). This basic event is called CCS-MDP-REC and has a probability of 0.5.

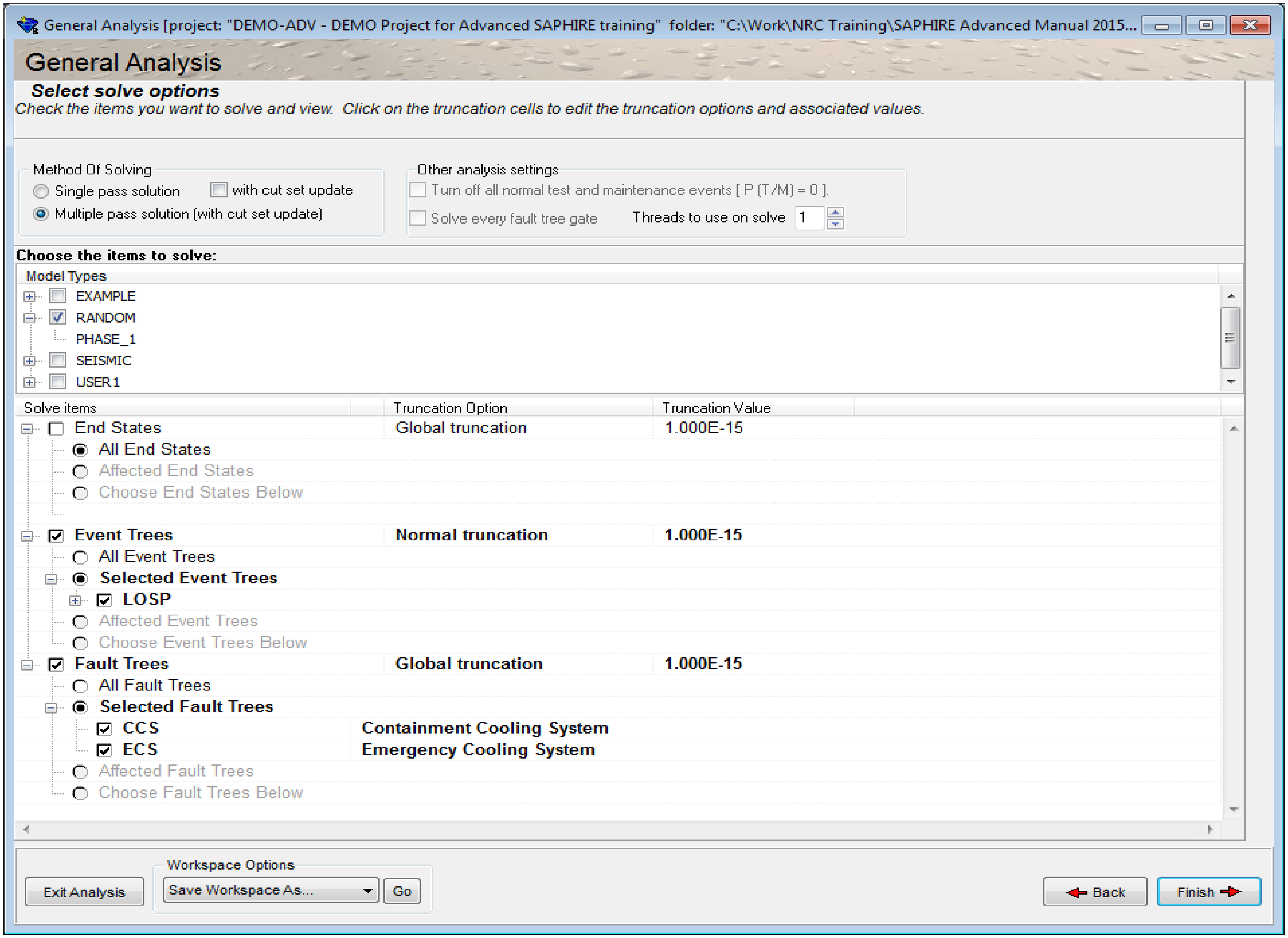


5. Modify the LOSP event tree by adding the RDG-A top event. This top event is questioned given failure of both ECS and CCS.

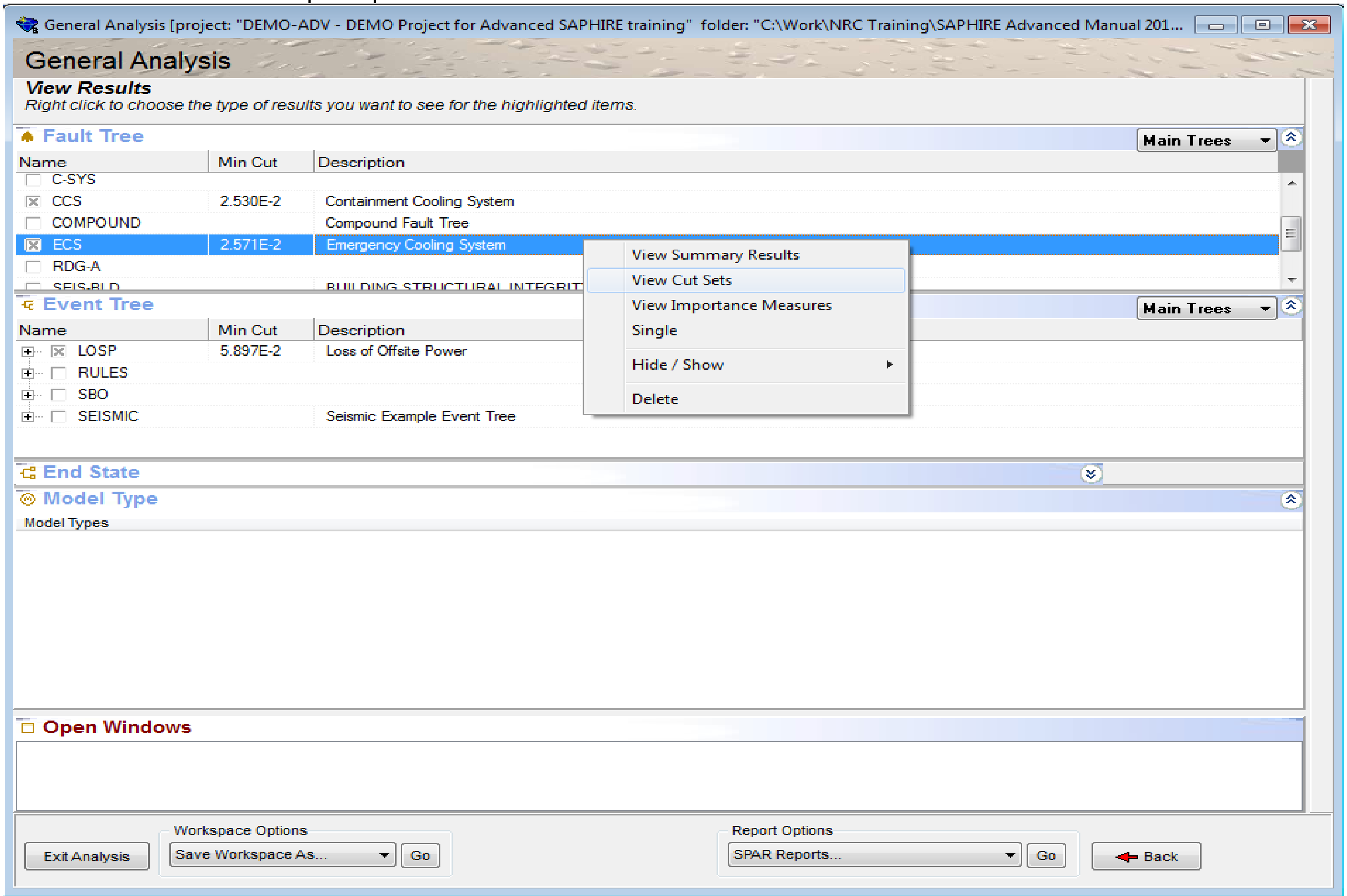


6. Select **Next**, once the LOSP event tree has been saved (along with the previous modifications).

7. Solve the fault trees (ECS and CCS) and LOSP event tree accident sequences by selecting the "Selected Event Trees" and "Selected Fault Trees" radio buttons, since only the modified logic models need to be re-solved and then click the **Finish** button.



8. The results from the analyses can be viewed by selecting event tree or fault tree of interest, right-click the mouse and then select the report option.



NOTES

| 12 | Mutually Exclusive Events

Workshop Objectives

The Mutually Exclusive Events workshop accompanies Section 12 in the Advanced SAPHIRE training manual. The workshop objectives are to gain experience using Post-processing Rules to remove mutually exclusive events and to practice using the Post Processing Rules editor.

Workshop Instructions

It is assumed that any combination of two (or more) motor operated valves appearing in a single cut set constitutes a mutually exclusive event. The DEMO-ADV project is used to create Post-processing Rules that will remove mutually exclusive events from all sequence cut sets.

1. Using the DEMO-ADV project, unmark any previously marked change sets and solve the ECS and CCS fault trees and then the LOSP Event Tree at 1.000E-15 truncation.

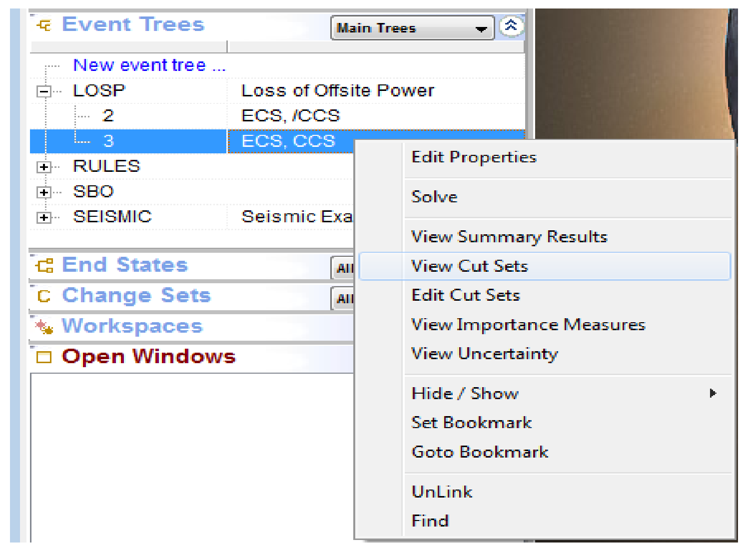
2. Edit the event tree accident sequence post-processing rules via the **Project → Edit Rules → ET (Post processing)** option.

3. Enter the post-processing rule as shown:

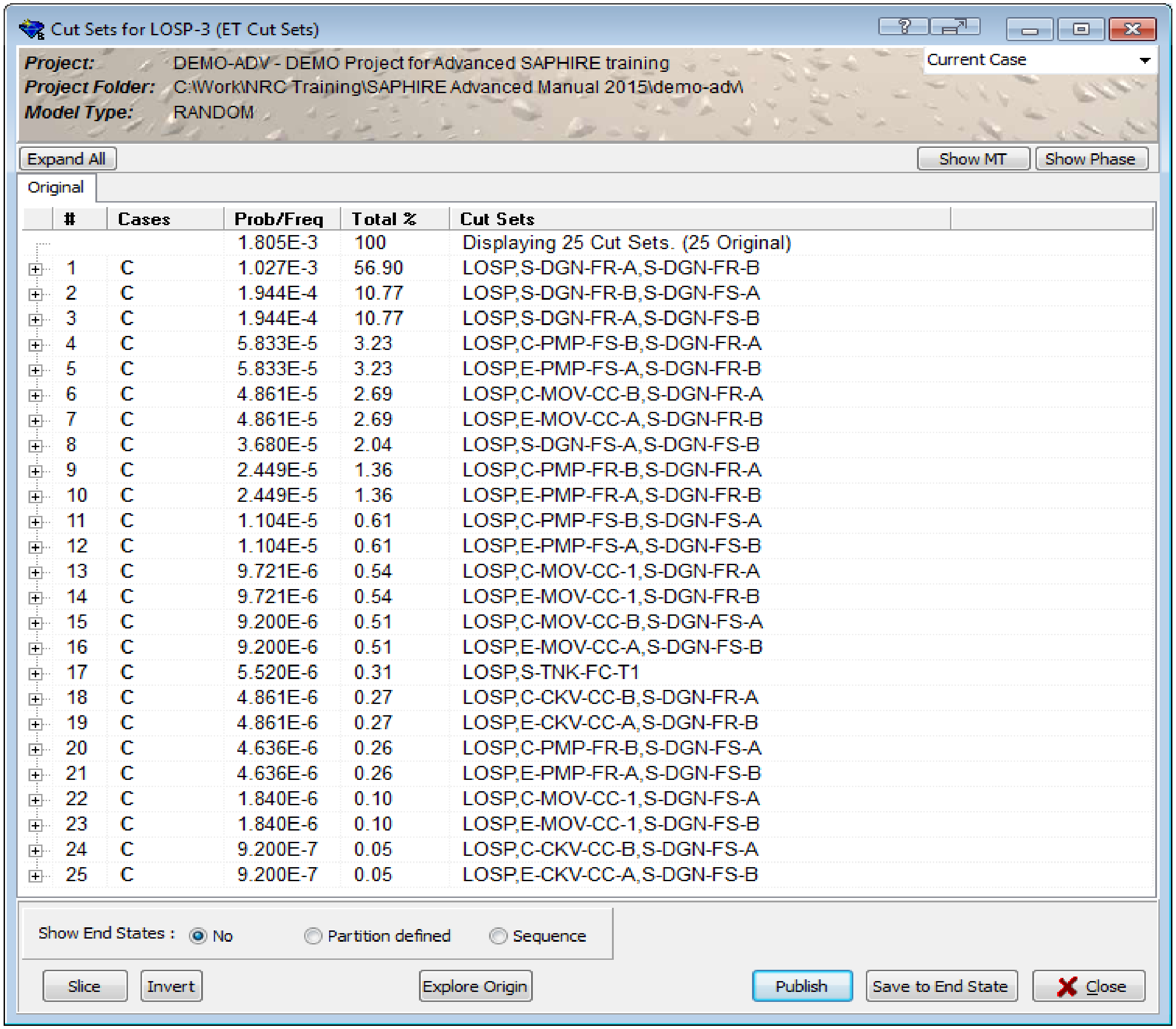
```
| This rule searches for any combination of two motor operated valves (MOVs) in a single cut
| set and then removes the cut set.  First define a macro for each MOV
E1 = E-MOV-CC-1;
EA = E-MOV-CC-A;
EB = E-MOV-CC-B;
C1 = C-MOV-CC-1;
CA = C-MOV-CC-A;
CB = C-MOV-CC-B;
| Second define macros identifying combinations of five of the six MOVs
MOV1 = (E1 + EA + EB + C1 + CA);
MOV2 = (E1 + EA + EB + C1 + CB);
MOV3 = (E1 + EA + EB + CA + CB);
MOV4 = (E1 + EA + C1 + CA + CB);
MOV5 = (E1 + EB + C1 + CA + CB);
MOV6 = (EA + EB + C1 + CA + CB);
| Now define a macro that removes combinations of MOVs
RM-MOV = (CB * MOV1) + (CA * MOV2) + (C1 * MOV3) + (EB * MOV4) + (EA * MOV5) + (E1 * MOV6);
| Now set up the if...then structure of the rule
if  RM-MOV  then
   DeleteRoot;
endif
```

4. Apply the sequence Post-processing Rules by solving the LOSP event tree with a check in the checkbox "**Apply Post-processing Rules**" and a truncation of 1.000E-08.

5. Display the cut sets by right clicking on the LOSP-3 sequence under LOSP in the Event Tree list panel and selecting the **View Cut Sets** option.



6. Review the LOSP sequence 3 cut sets to see the changes to the cut sets. Note that no cut sets with two (or more) motor operated valve failures appear:



#	Cases	Prob/Freq	Total %	Cut Sets
		1.805E-3	100	Displaying 25 Cut Sets. (25 Original)
1	C	1.027E-3	56.90	LOSP,S-DGN-FR-A,S-DGN-FR-B
2	C	1.944E-4	10.77	LOSP,S-DGN-FR-B,S-DGN-FS-A
3	C	1.944E-4	10.77	LOSP,S-DGN-FR-A,S-DGN-FS-B
4	C	5.833E-5	3.23	LOSP,C-PMP-FS-B,S-DGN-FR-A
5	C	5.833E-5	3.23	LOSP,E-PMP-FS-A,S-DGN-FR-B
6	C	4.861E-5	2.69	LOSP,C-MOV-CC-B,S-DGN-FR-A
7	C	4.861E-5	2.69	LOSP,E-MOV-CC-A,S-DGN-FR-B
8	C	3.680E-5	2.04	LOSP,S-DGN-FS-A,S-DGN-FS-B
9	C	2.449E-5	1.36	LOSP,C-PMP-FR-B,S-DGN-FR-A
10	C	2.449E-5	1.36	LOSP,E-PMP-FR-A,S-DGN-FR-B
11	C	1.104E-5	0.61	LOSP,C-PMP-FS-B,S-DGN-FS-A
12	C	1.104E-5	0.61	LOSP,E-PMP-FS-A,S-DGN-FS-B
13	C	9.721E-6	0.54	LOSP,C-MOV-CC-1,S-DGN-FR-A
14	C	9.721E-6	0.54	LOSP,E-MOV-CC-1,S-DGN-FR-B
15	C	9.200E-6	0.51	LOSP,C-MOV-CC-B,S-DGN-FS-A
16	C	9.200E-6	0.51	LOSP,E-MOV-CC-A,S-DGN-FS-B
17	C	5.520E-6	0.31	LOSP,S-TNK-FC-T1
18	C	4.861E-6	0.27	LOSP,C-CKV-CC-B,S-DGN-FR-A
19	C	4.861E-6	0.27	LOSP,E-CKV-CC-A,S-DGN-FR-B
20	C	4.636E-6	0.26	LOSP,C-PMP-FR-B,S-DGN-FS-A
21	C	4.636E-6	0.26	LOSP,E-PMP-FR-A,S-DGN-FS-B
22	C	1.840E-6	0.10	LOSP,C-MOV-CC-1,S-DGN-FS-A
23	C	1.840E-6	0.10	LOSP,E-MOV-CC-1,S-DGN-FS-B
24	C	9.200E-7	0.05	LOSP,C-CKV-CC-B,S-DGN-FS-A
25	C	9.200E-7	0.05	LOSP,E-CKV-CC-A,S-DGN-FS-B

Optional

Modify the recovery rule to remove instances of two diesel generator events appearing in the same cut set. For example, the line:

```
if  RM-MOV  then
```

could be changed to:

```
if  RM-MOV + (S-DGN-FS-A * S-DGN-FS-B) then
```

Resolve Sequence 3 cut sets (**Sequences → Solve**) using the "Auto Apply Post-Processing Rules" check box checked to verify that the rule functions as intended.

| 13 | User-Defined Model Types

Workshop Objectives

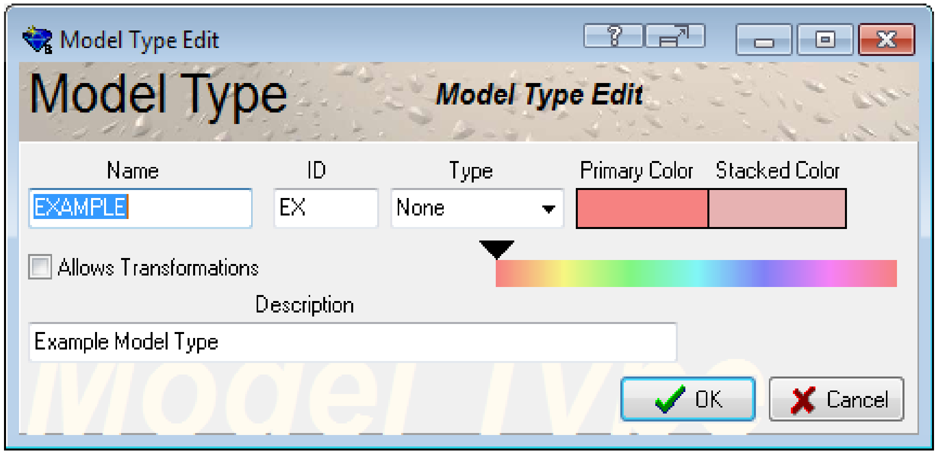
This Model Type workshop accompanies Section 13 in the Advanced SAPHIRE training manual. The workshop objectives are to investigate the user-defined Model Type features of SAPHIRE 8.

Workshop Instructions

Using the DEMO-ADV project, a new model type is created and its use in event trees and fault trees is investigated.

1. Use the DEMO-ADV project. Go to **Project** → **Model Types** and select **Add Model Type** from the pull-down menu on the Model Type dialogue and select **Go**.

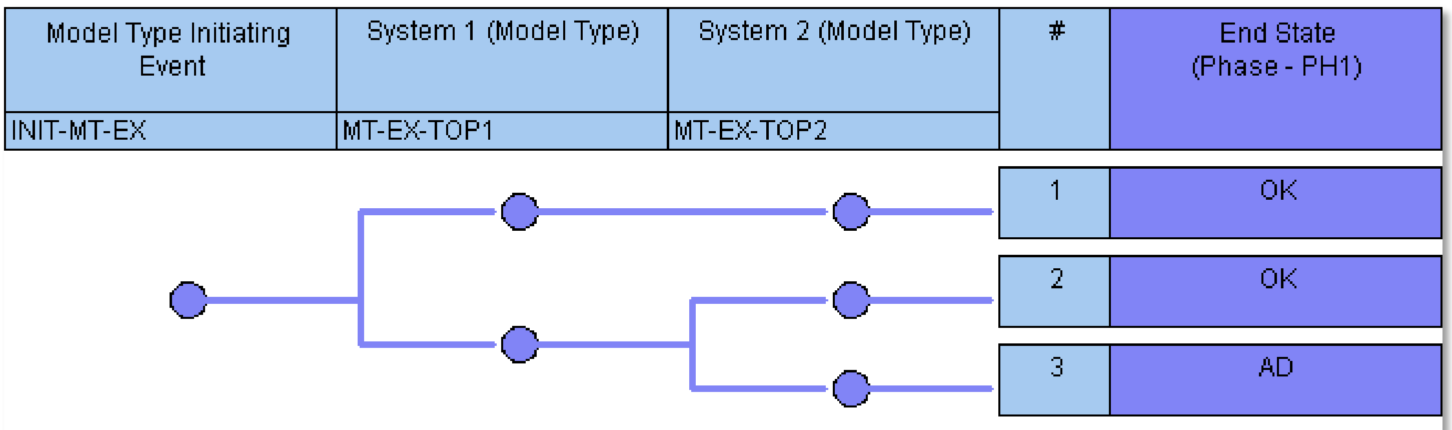
2. Use the Model Type Edit dialogue to name the model type EXAMPLE, select a color that is different than the RANDOM model type and give it a Description of **Example Model Type** and an ID of **EX**. Select **OK** and you're ready to use this new model type.



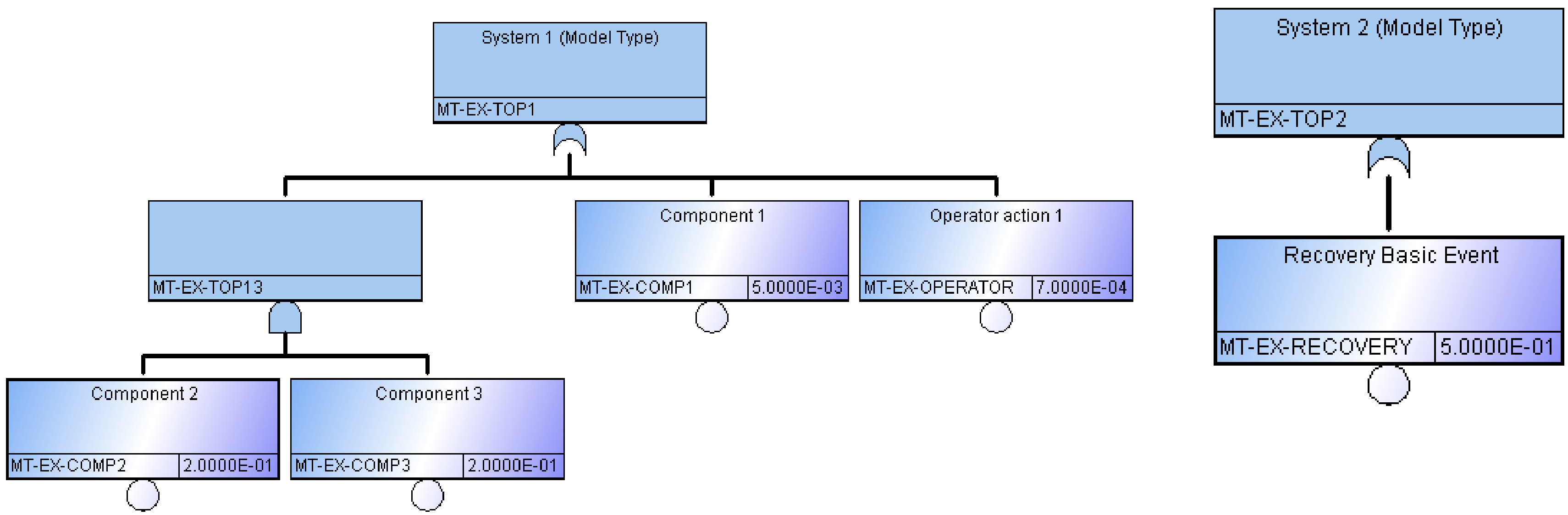
3. Build an event tree with the following tops:

Top	Name	Description
Initiating Event	INIT-MT-EX	Model Type Initiating Event
Top #1	MT-EX-TOP1	System 1 (Model Type)
Top #2	MT-EX-TOP2	System 2 (Model Type)

Add the branches and end states as below:

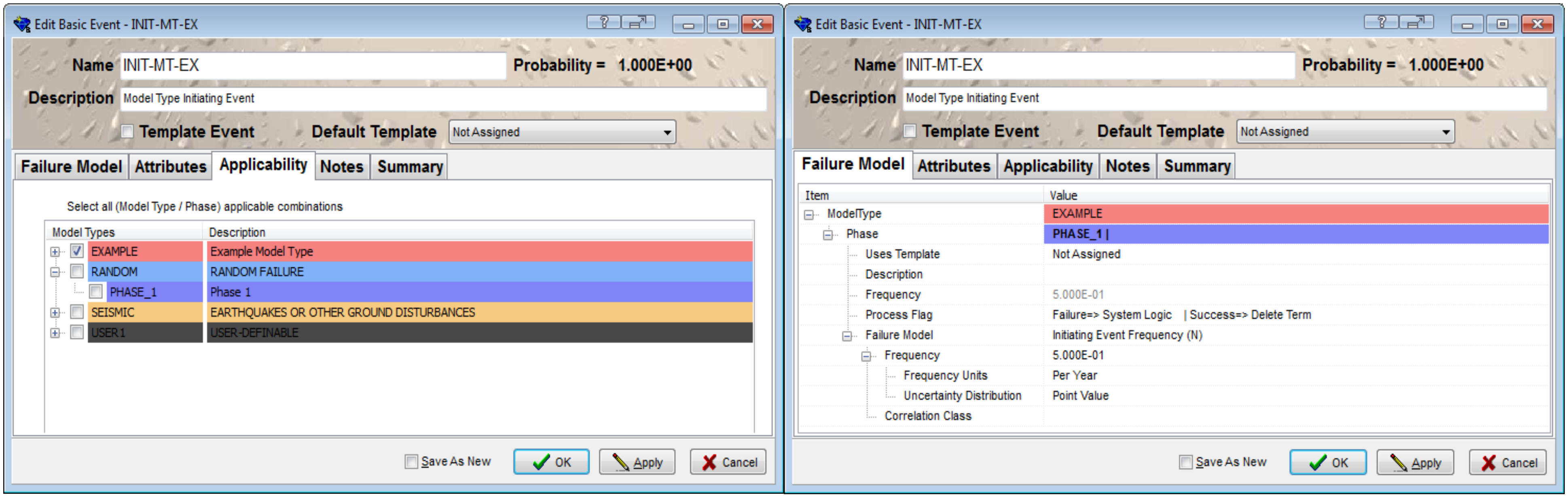


4. Build the two fault trees in the event tree with the following components using the RANDOM (default) model type and leave the basic event probabilities as default for now.



5. Open up the initiator Basic Event “INIT-MT-EX” for editing and select the **Applicability** tab. Change the Model Type to **EXAMPLE**. Then select the Failure Model tab and change the **Frequency** to 0.5 and click **Apply** or **OK**.

   The value field for ModelType in the Failure Model tab for INIT-MT-EX should now show the color of the model type selected.



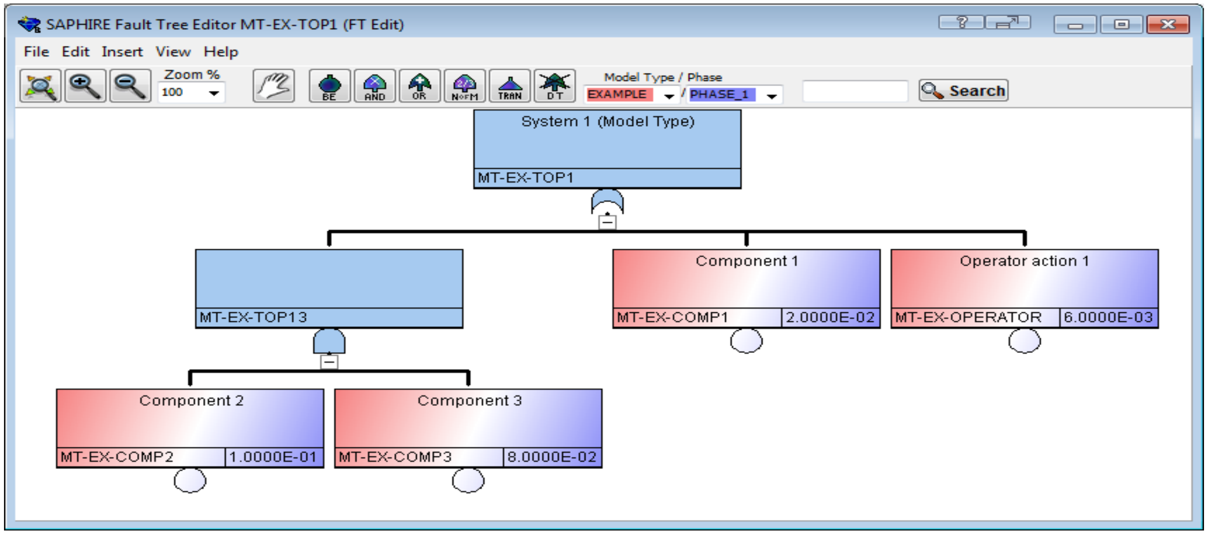
- Note that SAPHIRE will only allow one model type to be used for an initiating event and the current form allows you to select more than one model type. If more than one model type is selected SAPHIRE will use the one that is first alphabetically, so be sure to de-select all model types other than the one you want to use for initiating events.

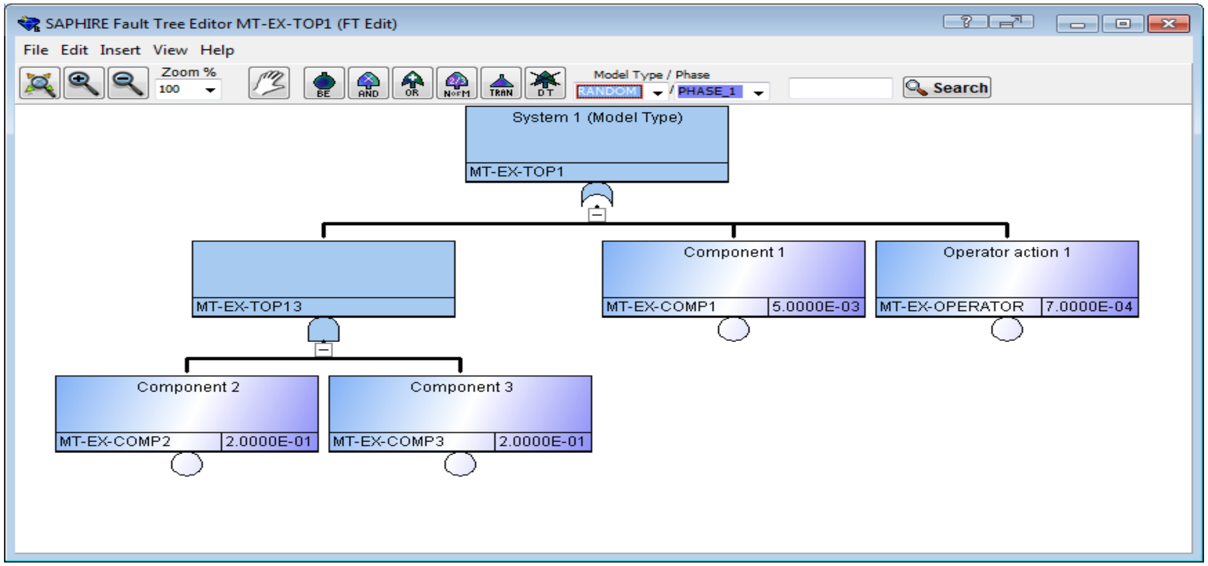
6. Modify the basic events with the following values:

- All events will have Failure Model = Failure Probability
- All events will be applicable to both EXAMPLE and RANDOM Model Types

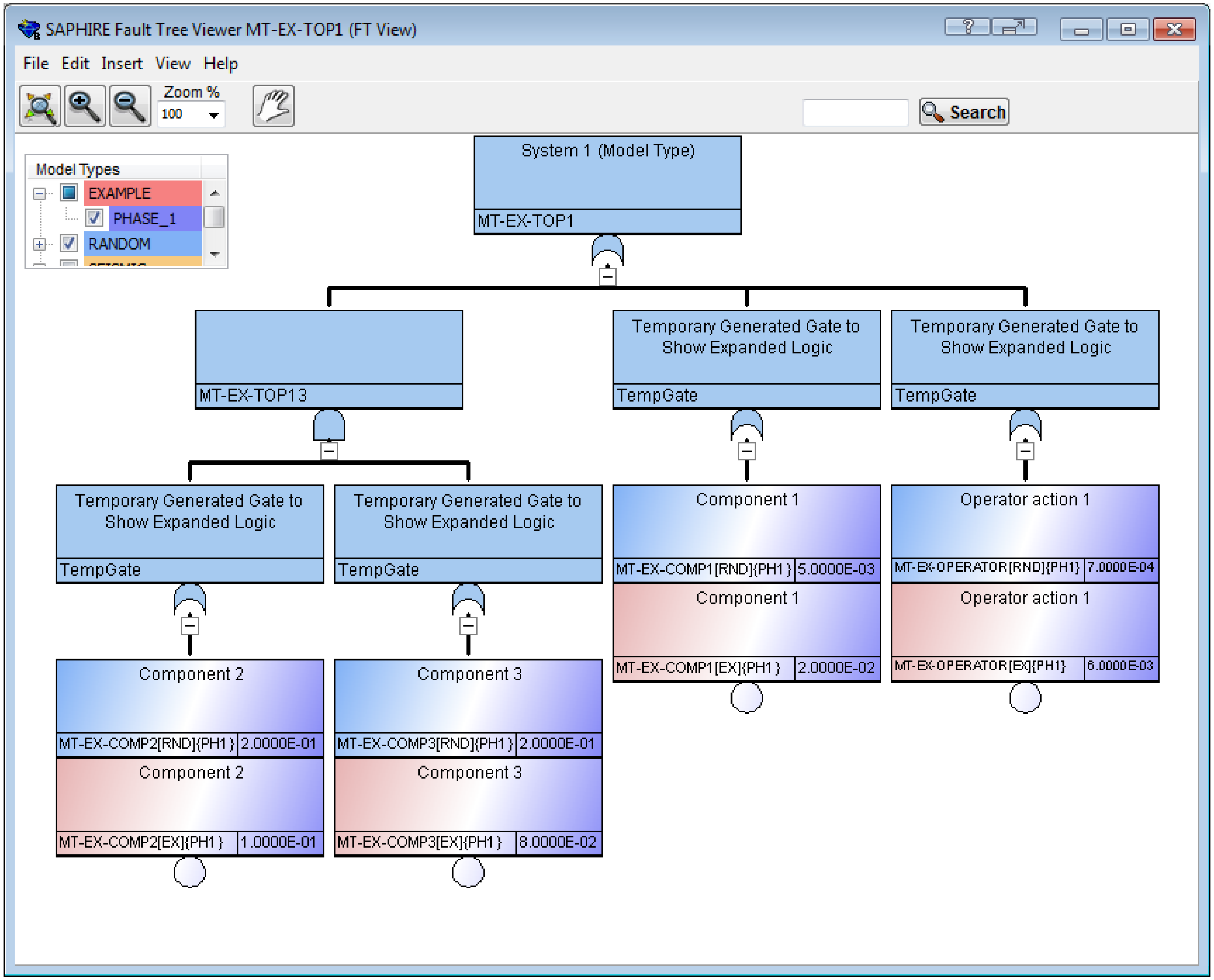
Basic Event	Model Type	Probability	Uncertainty
MT-EX-COMP1	EXAMPLE	2.000E-02	Beta, b = 200
	RANDOM	5.000E-03	Log Normal, EF = 5
MT-EX-COMP2	EXAMPLE	1.000E-01	Point Value
	RANDOM	2.000E-01	Point Value
MT-EX-COMP3	EXAMPLE	8.000E-02	Point Value
	RANDOM	2.000E-01	Point Value
MT-EX-OPERATOR	EXAMPLE	6.000E-03	Point Value
	RANDOM	7.000E-04	Point Value
MT-EX-RECOVERY	EXAMPLE	5.000E-01	Point Value
	RANDOM	5.000E-01	Point Value

7. To view the Model Type, the analyst will have to solve the Fault Tree using that Model Type and then SAPHIRE will default to this Model Type (the drop down option does not allow to change from one Model Type to another unless it has been invoked):

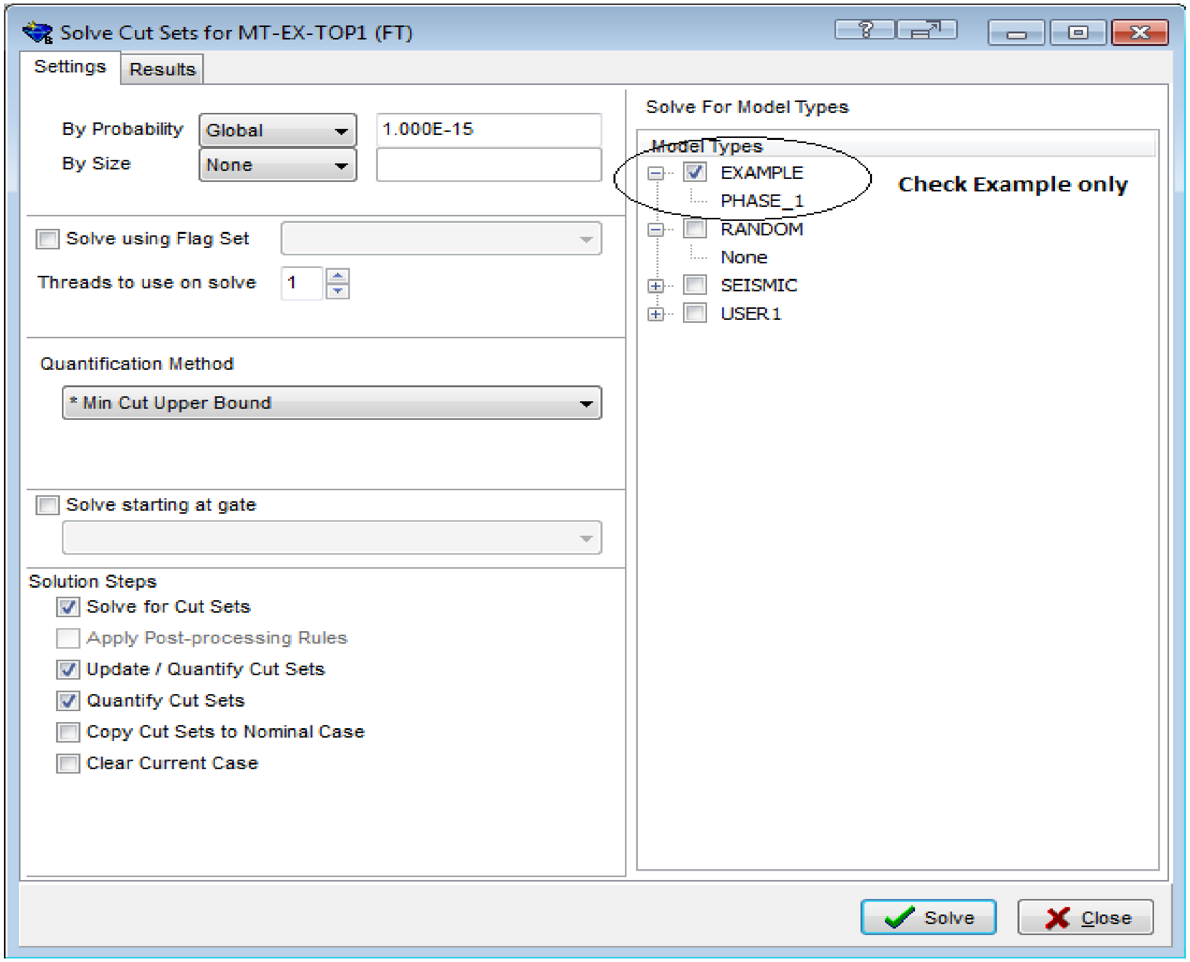




8. Use **File → View Expanded Model Types** to see all Model Types used in the fault tree:

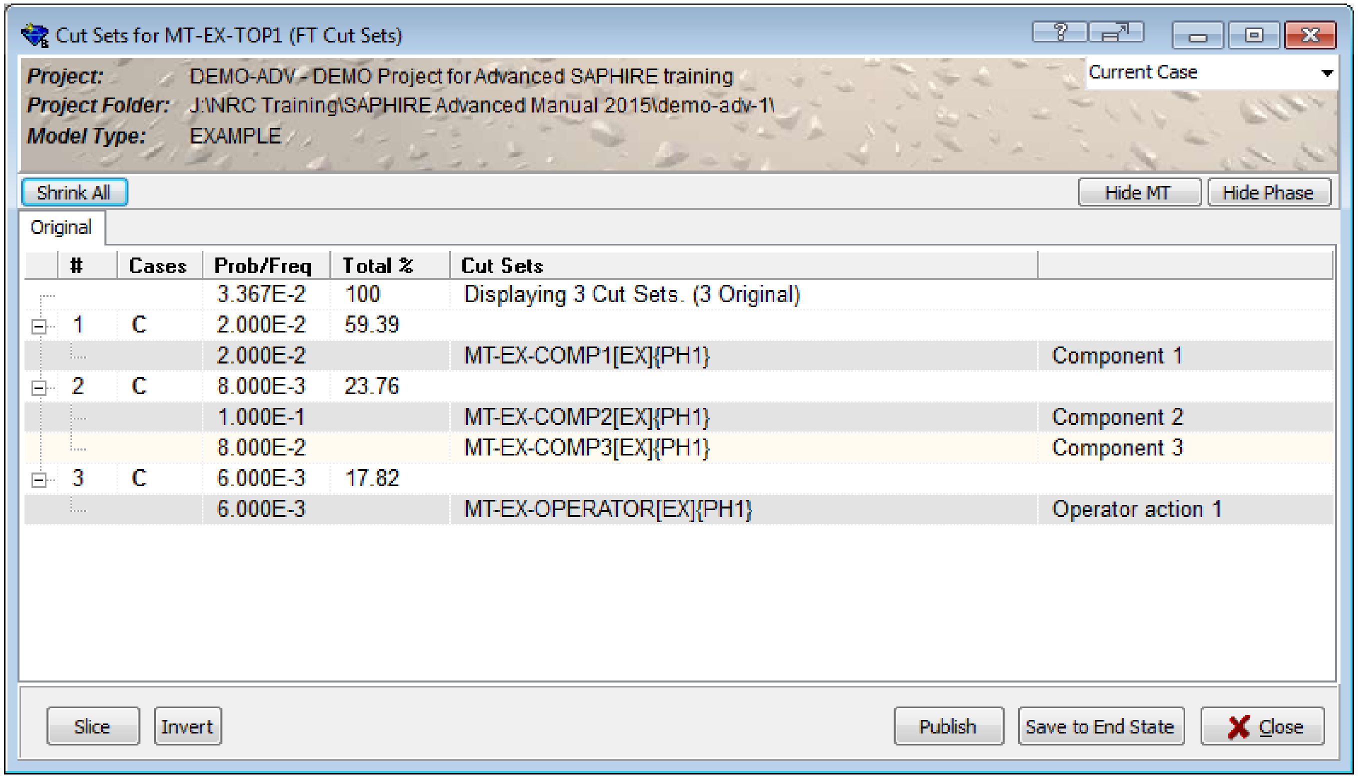


9. Solve the MX-EX-TOP1 fault tree for the EXAMPLE Model Type by selecting *ONLY* **EXAMPLE** in the solve cut sets dialogue.



- NOTE that there is an option to select multiple model types by checking their boxes in the solve screen. If EXAMPLE and RANDOM were selected, SAPHIRE would solve the tree shown previously in the Expanded Model Types. Typically we are concerned with just one model type at a time. Practice solving for the two different model types.

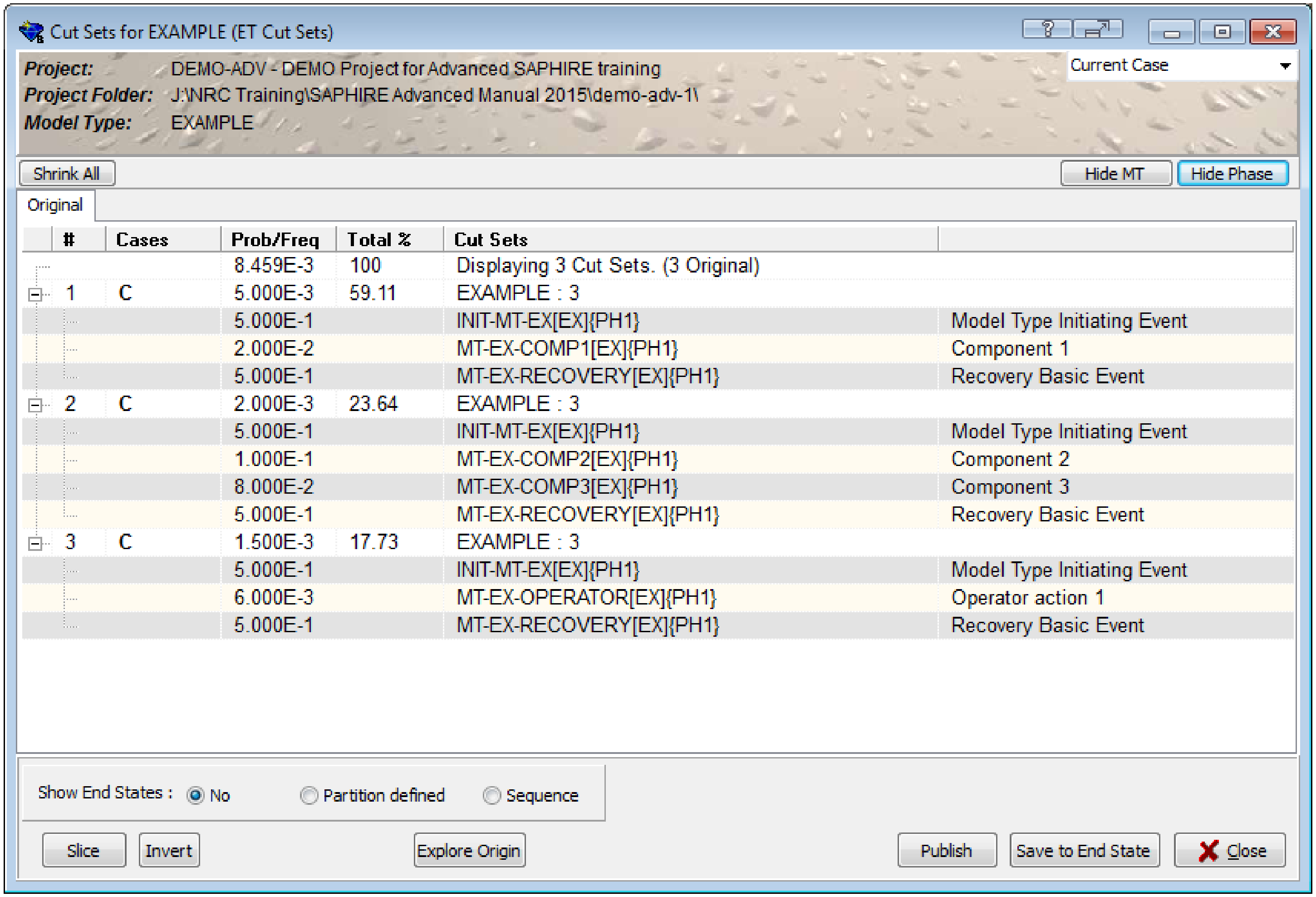
- Here are the expanded cut sets for MT-EX-TOP1 solved for the EXAMPLE model type:



- Notice that "{EX}" is appended on the end of each basic event to denote that the basic event is susceptible to the EXAMPLE model type (Select the Show MT option).

10. Solve the MT-EXAMPLE event tree for the EXAMPLE Model Type by selecting *ONLY* **EXAMPLE** in the solve cut sets dialogue.

- Here are the sequences for MT-EXAMPLE solved for the EXAMPLE model type:



- These sequences are only available in the EXAMPLE model type

| 14 | The Large Event Tree Methodology

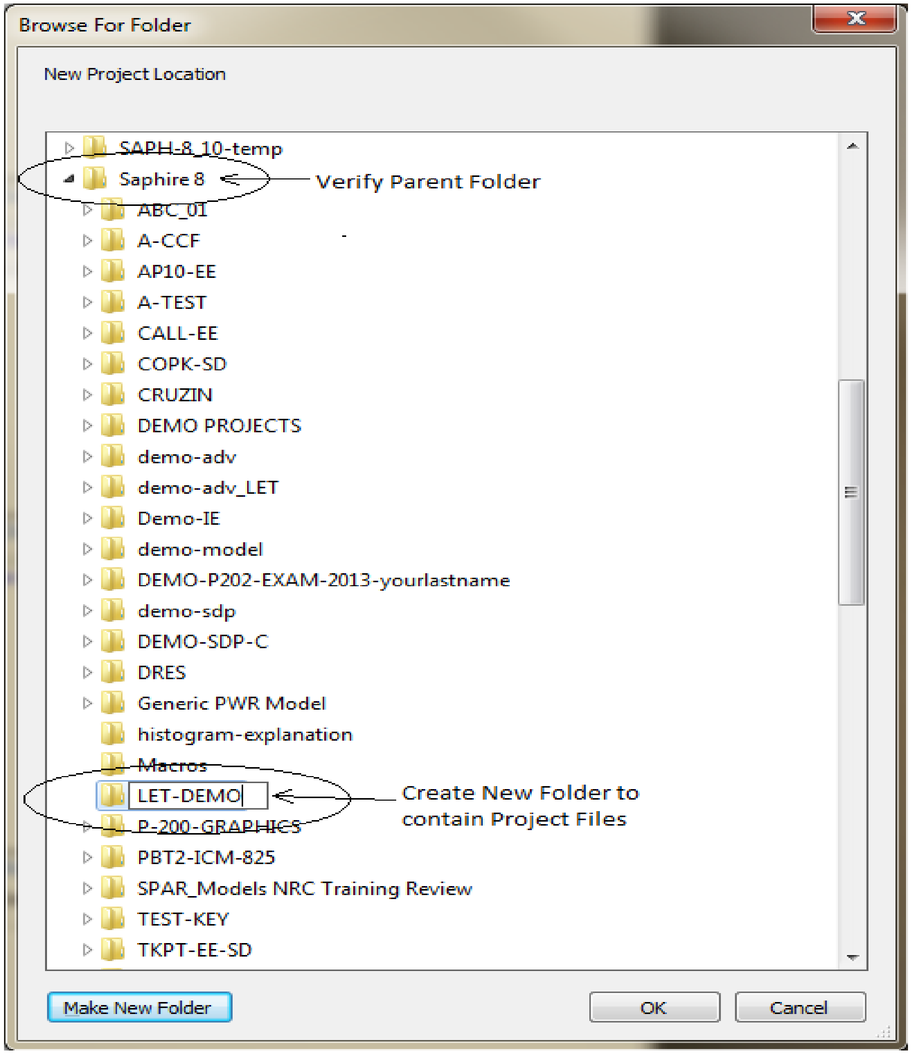
Workshop Objectives

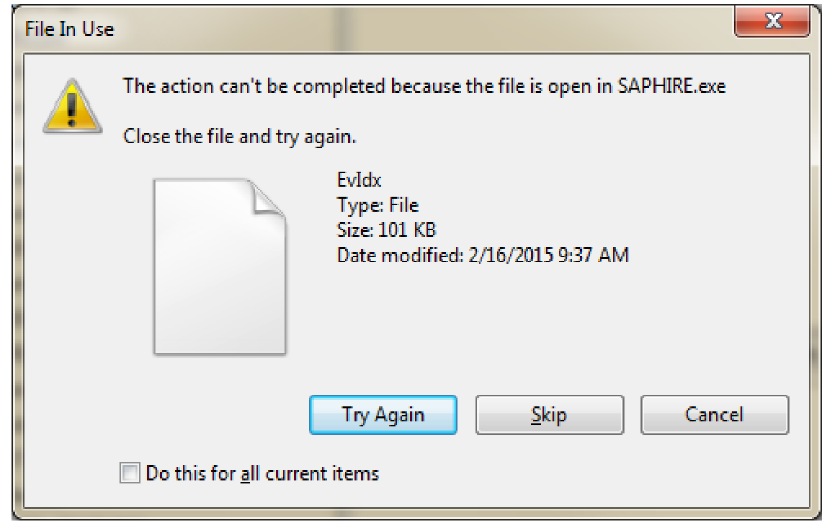
The Large Event Tree Methodology workshop accompanies Section 14 in the Advanced SAPHIRE training manual. The workshop objectives are to gain experience using the large event tree methodology by creating and analyzing an example case in SAPHIRE.

Workshop Instructions

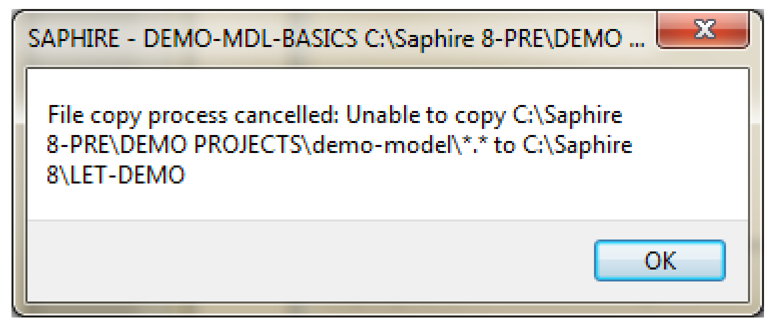
A new project will be created using the project name LET-DEMO. Within this project, create three event trees, add event tree rules, and develop fault tree flag sets. Then, generate the sequence logic, evaluate the sequence "cut sets," and view the results.

1. Open up the DEMO-ADV project in SAPHIRE 8 and use the main menu **File → Save As…** option to create a new folder called LET-DEMO and save the DEMO project there.



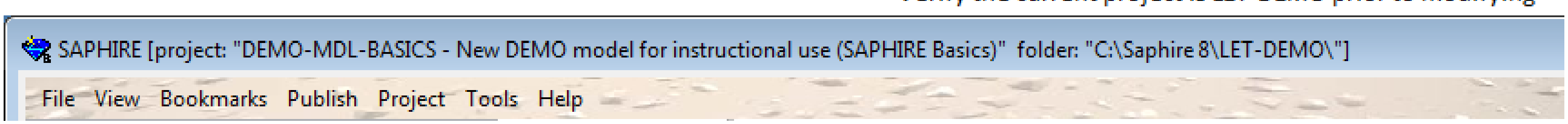


Click Skip



Click OK

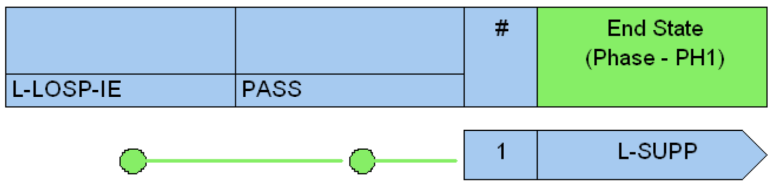
Verify the current project is LET-DEMO prior to modifying



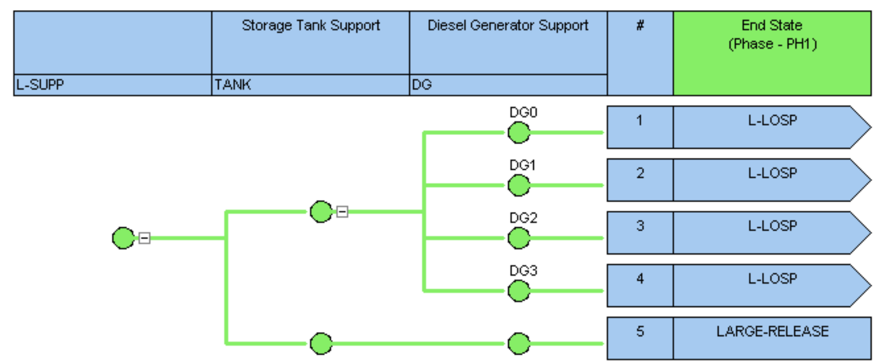
2. Change the project name (**Project → Modify**) to LET-DEMO for the project in the LET-DEMO folder and archive the project in the current LET-DEMO folder.

3. Create the three event trees shown below. Save each event tree using their respective event tree name shown. Remember to include the end states and transfers in the event trees.

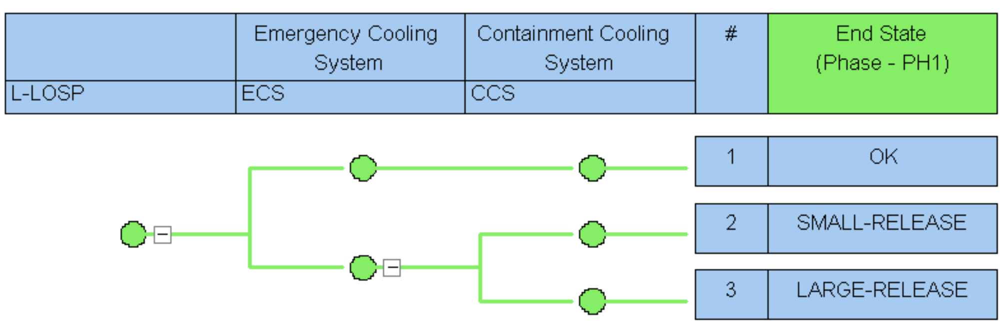
The "Initiating Event" Tree “named L-LOSPIE”



The "Support System" Tree “named L-SUPP”



The "Plant Response" Tree “named L-LOSP”



4. Create the following ECS fault trees. To create these fault trees, go into **Fault Trees** list panel and highlight the ECS fault tree, right click, and select **Edit Logic**. Then save the fault tree using **File → Save As** and type in the new fault tree name (i.e., **ECS-0**) (There is a check box to have SAPHIRE rename gates, if unchecked SAPHIRE will leave gate names the same and descriptions will stay. Unchecking this option is OK, since the fault tree will not be linked to the event tree logic). (Perform this step to create the following fault trees.)
   ECS-0 ECS-A ECS-B ECS-AB

5. Perform the same process as step 4 to create the following CCS fault trees.
   CCS-0 CCS-A CCS-B CCS-AB

6. Create the following diesel generator fault trees and TANK fault tree using the logic specified in the Advanced SAPHIRE training manual.
   DG0 DG1 DG2 DG3 TANK

7. Create the following fault tree flag sets to be used to trim the fault tree logic in order to generate the appropriate split fraction probability for the event tree.

Flag Set Name	Basic Events	House Event Identifier
FT-FLAG-0	S-DGN-FR-A S-DGN-FS-A S-DGN-FR-B S-DGN-FS-B S-TNK-FC-T1	FALSE FALSE FALSE FALSE FALSE
FT-FLAG-A	S-DGN-FR-A S-DGN-FS-A S-DGN-FR-B S-DGN-FS-B S-TNK-FC-T1	TRUE TRUE FALSE FALSE FALSE

Flag Set Name	Basic Events	House Event Identifier
FT-FLAG-B	S-DGN-FR-A S-DGN-FS-A S-DGN-FR-B S-DGN-FS-B S-TNK-FC-T1	FALSE FALSE TRUE TRUE FALSE
FT-FLAG-AB	S-DGN-FR-A S-DGN-FS-A S-DGN-FR-B S-DGN-FS-B S-TNK-FC-T1	TRUE TRUE TRUE TRUE FALSE

8. Assign the fault tree flag sets to the appropriate fault tree for the calculation process. This is performed by highlighting the fault tree in the Fault Trees list panel, right clicking and selecting Edit Properties. Click the drop down box under Default Flag Set and selecting the appropriate flag set. Perform this step for each fault tree.

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

9. Generate fault tree cut sets for all of the new fault trees (Fault Trees list panel, highlight the fault trees, right click and select **Solve** with no truncation.

10. Modify the Failure Model for the new DG, ECS and CCS fault trees by changing the **Basic Events** filter to show A**ll** and then highlight the fault tree top event and select **Edit Basic Event** (or double click the fault tree top). Change the Failure Model from Failure Probability (1) to **Use mincut upperbound of fault tree with same name (S)**. Also, change the process flag for these top events to **Y Failure => Developed Event | Success => /Developed Event**.

11. Set the L-LOSP-IE initiating event frequency to 2.3/yr.

12. From the **Event Tree** list panel, highlight the L-SUPP event tree (change the event tree filter to show **All**), right click and select **Edit Linkage Rules**, enter the support system event tree rules for the L-SUPP event tree.

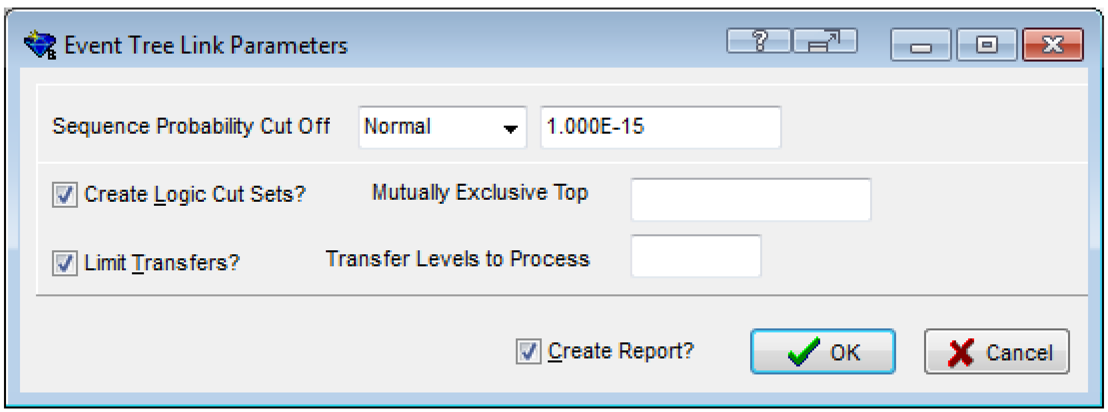
```
| Support system rules
if always then
 /DG   = DG0;
 DG[1] = DG1;
 DG[2] = DG2;
 DG[3] = DG3;
Endif
```

13. From the **Event Tree** list panel, highlight the L-LOSP event tree, right click and select **Edit Linkage Rules**, enter the plant response event tree rules for the L-LOSP event tree.

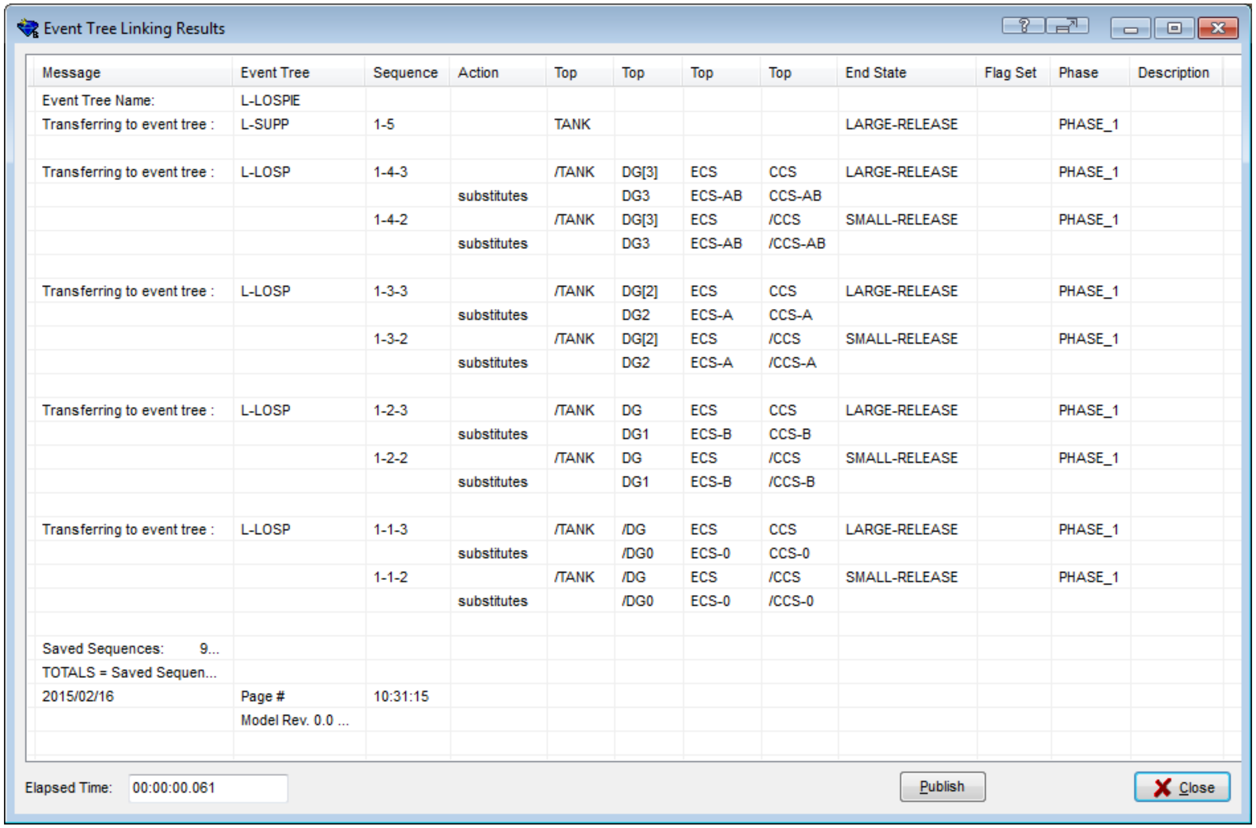
```
| Plant response system rules
|     THE RULES CAN BE WRITTEN IN TERMS OF BRANCH IDENTIFIERS OR SUBSTITUTED NAMES
if /DG then
 /ECS = ECS-0;
  ECS = ECS-0;
 /CCS = CCS-0;
  CCS = CCS-0;
elsif DG[1] then
 /ECS = ECS-B;
  ECS = ECS-B;
 /CCS = CCS-B;
  CCS = CCS-B;
```

```
elsif DG2 then
 /ECS = ECS-A;
  ECS = ECS-A;
 /CCS = CCS-A;
  CCS = CCS-A;
else
 /ECS = ECS-AB;
  ECS = ECS-AB;
 /CCS = CCS-AB;
  CCS = CCS-AB;
endif
```

14. Link the L-LOSPIE event tree using the Large Event Tree Methodology options by highlighting the L-LOSPIE event tree, right click and select **Link**.



The report should look like that shown below.



Event Tree Linking Results

Message	Event Tree	Sequence	Action	Top	Top	Top	Top	End State	Flag Set	Phase	Description
Event Tree Name:	L-LOSPIE										
Transferring to event tree :	L-SUPP	1-5		TANK				LARGE-RELEASE		PHASE_1	
Transferring to event tree :	L-LOSP	1-4-3		/TANK	DG[3]	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG3	ECS-AB	CCS-AB				
		1-4-2		/TANK	DG[3]	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		DG3	ECS-AB	/CCS-AB				
Transferring to event tree :	L-LOSP	1-3-3		/TANK	DG[2]	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG2	ECS-A	CCS-A				
		1-3-2		/TANK	DG[2]	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		DG2	ECS-A	/CCS-A				
Transferring to event tree :	L-LOSP	1-2-3		/TANK	DG	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		DG1	ECS-B	CCS-B				
		1-2-2		/TANK	DG	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		DG1	ECS-B	/CCS-B				
Transferring to event tree :	L-LOSP	1-1-3		/TANK	/DG	ECS	CCS	LARGE-RELEASE		PHASE_1	
			substitutes		/DG0	ECS-0	CCS-0				
		1-1-2		/TANK	/DG	ECS	/CCS	SMALL-RELEASE		PHASE_1	
			substitutes		/DG0	ECS-0	/CCS-0				
Saved Sequences: 9...											
TOTALS = Saved Sequen...											
2015/02/16	Page #	10:31:15									
	Model Rev. 0.0 ...										

Elapsed Time: 00:00:00.061 Publish Close

15. The sequence “cut sets” may be quantified (but not solved) since they exist following the linking process. To quantify these sequences, highlight the L-LOSPIE event tree, right click and select **Solve**. Uncheck the “Solve for Cut Sets” box and make sure the “Q**uantify Cut Sets**“ box is checked.

16. To view the cut sets, highlight the event tree, right click, and select the **View Cut Sets** option. (The report shown below is from the **Publish → Event Tree Reports → Detailed Cut Sets (by Sequence)**.)

HTML Viewer

Detailed Cut Sets (by Sequence) **New DEMO model for Large Event Tree**

2/16/2015 11:24:32 AM

Note: Cut sets that contribute >= 1.00% are reported

SEQUENCE/CS#	PROB./FREQ.	TOTAL %	BASIC EVENT	DESCRIPTION	PROBABILITY
L-LOSPIE - 1-1-2	4.543E-4	100%		Displaying 1 of 1 cut sets	
1	4.543E-4	100%	L-LOSP-IE		2.300E+0
			/CCS-0	Containment Cooling System	9.998E-1
			/DG0	Success of Diesel Generator A and B	9.505E-1
			ECS-0	Emergency Cooling System	2.079E-4
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-1-3	9.446E-8	100%		Displaying 1 of 1 cut sets	
1	9.446E-8	100%	L-LOSP-IE		2.300E+0
			CCS-0	Containment Cooling System	2.079E-4
			/DG0	Success of Diesel Generator A and B	9.505E-1
			ECS-0	Emergency Cooling System	2.079E-4
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-2-3	1.729E-4	100%		Displaying 1 of 1 cut sets	
1	1.729E-4	100%	L-LOSP-IE		2.300E+0
			CCS-B	Containment Cooling System	1.000E+0
			DG1	Diesel Generator B Failures	2.505E-2
			ECS-B	Emergency Cooling System	3.001E-3
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-3-2	5.744E-2	100%		Displaying 1 of 1 cut sets	
1	5.744E-2	100%	L-LOSP-IE		2.300E+0
			/CCS-A	Containment Cooling System	9.970E-1
			DG2	Diesel Generator A Failures	2.505E-2
			ECS-A	Emergency Cooling System	1.000E+0
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-3-3	1.729E-4	100%		Displaying 1 of 1 cut sets	
1	1.729E-4	100%	L-LOSP-IE		2.300E+0
			CCS-A	Containment Cooling System	3.001E-3
			DG2	Diesel Generator A Failures	2.505E-2
			ECS-A	Emergency Cooling System	1.000E+0
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-4-3	1.453E-3	100%		Displaying 1 of 1 cut sets	
1	1.453E-3	100%	L-LOSP-IE		2.300E+0
			CCS-AB	Containment Cooling System	1.000E+0
			DG3	Diesel Generator A and B Failures	6.316E-4
			ECS-AB	Emergency Cooling System	1.000E+0
			/TANK	Storage Tank Support	1.000E+0
L-LOSPIE - 1-5	5.520E-6	100%		Displaying 1 of 1 cut sets	
1	5.520E-6	100%	L-LOSP-IE		2.300E+0
			TANK	Storage Tank Support	2.400E-6

Print... Save As... Close