INL/EXT-05-00643
Rev. 1

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Data Loading Manual

K. J. Kvarfordt
S. T. Wood
C. L. Smith

August 2008

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-05-00643
Rev. 1

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Data Loading Manual

K. J. Kvarfordt
S. T. Wood
C. L. Smith

August 2008

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington D.C. 20555
Job Code N6203

AVAILABILITY NOTICE

Availability of Reference Materials Cited in NRC Publications

Most documents cited in NRC publications will be available from one of the following sources:

1. The NRC Public Document Room, 11555 Rockville Pike, Rockville, MD 20852 (pdr@nrc.gov)
2. The Superintendent of Documents, U. S. Government Printing Office (GPO), Mail Stop SSOP, Washington, DC 20402-9328
3. The National Technical Information Service, Springfield, VA 22161

Although the listing that follows represents the majority of documents cited in NRC publications, it is not intended to be exhaustive.

Referenced documents available for inspection and copying for a fee from the NRC Public Document Room include NRC correspondence and internal NRC memoranda; NRC bulletins, circulars, information notices, inspection and investigative notices; licensee event reports; vendor reports and correspondence; Commission papers; and applicant and licensee documents and correspondence.

The following documents in the NUREG series are available for purchase from the GPO Sales Program: formal NRC staff and contractor reports, NRC-sponsored conference proceedings, international agreement reports, grant publications, and NRC booklets and brochures. Also available are regulatory guides, NRC regulations in the *Code of Federal Regulations*, *and Nuclear Regulatory Commission Issuances*.

Documents available from the National Technical Information Service include NUREG-series reports and technical reports prepared by other Federal agencies and reports prepared by the Atomic Energy Commission, forerunner agency to the Nuclear Regulatory Commission.

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions. *Federal Register* notices, Federal and State legislation, and congressional reports can usually be obtained from these libraries.

Documents such as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings are available for purchase from the organization sponsoring the publication cited.

Single copies of NRC draft reports are available free, to the extent of supply, upon written request to the Office of Administration, Distribution and Mail Services Section U. S. Nuclear Regulatory Commission, Washington, DC 20555-0001.

The public maintains copies of industry codes and standards used in a substantive manner in the NRC regulatory process at the NRC Library, Two White Flint North, 11545 Rockville Pike, Rockville, MD, 20852, for use. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from the American National Standards Institute, 1430 Broadway, New York, NY 10018.

DISCLAIMER NOTICE

PREVIOUS REPORTS

Smith, C. L., et al., *Testing, Verifying, and Validating SAPHIRE Versions 6.0 and 7.0*, NUREG/CR-6688, October 2000.

K. D. Russell, et al. *Systems Analysis Programs for Hands-on Reliability Evaluations (SAPHIRE) Version 6.0 - System Overview Manual*, NUREG/CR-6532, May 1999.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 5.0, Volume 2 - Reference Manual*, NUREG/CR-6116, EGG-2716, July 1994.

K. D. Russell et al., *Verification and Validation (V&V), Volume 9 – Reference Manual*, NUREG/CR-6116, EGG-2716, July 1994.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 4.0, Volume 1 - Reference Manual*, NUREG/CR-5813, EGG-2664, January 1992.

K. D. Russell et al., *Integrated Reliability and Risk Analysis System (IRRAS) Version 2.5 Reference Manual*, NUREG/CR-5300, EGG-2613, March 1991.

K. D. Russell, M. B. Sattison, D. M. Rasmuson, *Integrated Reliability and Risk Analysis System (IRRAS) - Version 2.0 User's Guide*, NUREG/CR-5111, EGG-2535, manuscript completed March 1989, published June 1990.

K. D. Russell, D. M. Snider, M. B. Sattison, H. D. Stewart, S.D. Matthews, K. L. Wagner, *Integrated Reliability and Risk Analysis System (IRRAS) User's Guide - Version 1.0 (DRAFT)*, NUREG/CR-4844, EGG-2495, June 1987.

ABSTRACT

The Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) is a software application developed for performing a complete probabilistic risk assessment (PRA) using a personal computer. SAPHIRE is primarily funded by the U.S. Nuclear Regulatory Commission (NRC) and developed by the Idaho National Laboratory. This report is intended to assist the user to enter PRA data into the SAPHIRE program using the built-in MAR-D ASCII-text file data transfer process. Towards this end, a small sample database is constructed and utilized for demonstration. Where applicable, the discussion includes how the data processes for loading the sample database relate to the actual processes used to load a larger PRA models. The procedures described herein were developed for use with SAPHIRE Version 6.0 and Version 7.0. In general, the data transfer procedures for version 6 and 7 are the same, but where deviations exist, the differences are noted. The guidance specified in this document will allow a user to have sufficient knowledge to both understand the data format used by SAPHIRE and to carry out the transfer of data between different PRA projects.

FOREWORD

The U.S. Nuclear Regulatory Commission has developed the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) software used to perform probabilistic risk assessments (PRAs) on a personal computer. SAPHIRE enables users to supply basic event data, create and solve fault and event trees, perform uncertainty analyses, and generate reports. In that way, analysts can perform PRAs for any complex system, facility, or process.

SAPHIRE can be used to model a plant's response to initiating events, quantify core damage frequencies, and identify important contributors to core damage (Level 1 PRA). The program can also be used to evaluate containment failure and release models for severe accident conditions, given that core damage has occurred (Level 2 PRA). In so doing, the analyst could build the PRA model assuming that the reactor is initially at full power, low power, or shutdown. In addition, SAPHIRE can be used to analyze both internal and external events, and it includes special features for transforming models built for internal event analysis to models for external event analysis. It can also be used in a limited manner to quantify the frequency of release consequences (Level 3 PRA). Because this software is a very detailed technical tool, users should be familiar with PRA concepts and methods used to perform such analyses.

SAPHIRE has evolved with advances in computer technology. The versions currently in use (6 and 7) run in the Microsoft Windows® environment. A user-friendly interface, Graphical Evaluation Module (GEM), streamlines and automates selected SAPHIRE inputs and processes for performing event assessments.

SAPHIRE has also evolved with users' needs, and Versions 6 and 7 include new features and capabilities for developing and using larger, more complex models. For example, Version 7 can solve up to 2 million sequences and includes enhancements for cut set slicing, event tree rule linkage, and reporting options.

This NUREG-series report comprises seven volumes, which address SAPHIRE/GEM Versions 6 and 7. Volume 1, "Overview/Summary," gives an overview of the functions available in SAPHIRE and presents general instructions for using the software. Volume 2, "Technical Reference," discusses the theoretical background behind the SAPHIRE functions. Volume 3, "SAPHIRE Users' Manual," provides installation instructions and a step-by-step approach to using the program's features. Volume 4, "SAPHIRE Tutorial Manual," provides an example of the overall process of constructing a PRA database. Volume 5, "GEM/GEMDATA Reference Manual," discusses the use of GEM. Volume 6, "SAPHIRE Quality Assurance (QA) Manual," discusses QA methods and tests. Lastly, Volume 7, "SAPHIRE Data Loading Manual," assists the user in entering PRA data into SAPHIRE using the built-in MAR-D ASCII-text file data transfer process.

Christiana H. Lui, Director
Division of Risk Analysis
Office of Nuclear Regulatory Research

CONTENTS

EXECUTIVE SUMMARY

The Data Loading Manual contains an overview of functions for creating event trees and fault trees, defining accident sequences and basic event failure data, solving system fault trees and accident sequence event trees, quantifying cut sets, performing sensitivity and uncertainty analyses, documenting the results, and generating reports. The process of creating a SAPHIRE project is described in terms of the ASCII-formatted data structures available via the MAR-D option. MAR-D is a mechanism in SAPHIRE to import or export probabilistic risk assessment data – via a nonproprietary text format – for use, modification, or storage outside of SAPHIRE.

In order to understand the data import/export functionality, one must understand the parts of a SAPHIRE project. A project is any grouping of fault trees and event trees with their associated basic events, cut sets, reliability data, and descriptions. Inside a project, SAPHIRE reserves storage areas for the various types of information. For example, all basic event data is automatically placed in the base case part of the database (the "current case" part of the database is used only when performing an analysis). Note that basic fault tree and event tree logic remains the same for both current and base cases.

The tutorial in this document leads the student through (a) the basic construction of event tree and fault trees, (b) entering basic event data, and (c) generation and quantification of both fault tree and sequence cut sets. Once the project is complete, the data structures related to the fault trees, event trees, and basic events are discussed. The example that is used is one of modeling upset conditions related to going to work. Consequently, a "going to work" event tree and associated fault trees are used.

One application of the data files that are available from SAPHIRE is for use in quality assurance practices. These text-formatted files may be exported, reviewed by an independent party, and stored for later retrieval. Toward that end, the format for all information that may be entered into SAPHIRE and later exported is defined. For example, one section describes how to load fault trees and associated data in order to verify their accuracy.

The types of data that is defined and discussed in this document includes:

Project name and descriptions
Project attributes
Project text
Project event tree recovery rules
Project fault tree recovery rules
Project end state partition rules

Basic event names and descriptions
Basic event failure rates Basic event attributes Basic event transformations and compound events
Basic event compound information
Basic event notes

Fault tree graphics
Fault tree names and descriptions
Fault tree text
Fault tree attributes
Fault tree logic
Fault tree cut sets

Fault tree recovery rules

Event tree graphics
Event tree names and descriptions
Event tree text
Event tree attributes
Event tree logic
Event tree rules
Event tree recovery rules
Event tree end state partition rules

End state names and descriptions
End state text
End state cut sets

Sequence names and descriptions
Sequence cut sets
Sequence attributes
Sequence text
Sequence logic
Sequence recovery rules
Sequence end state partition rules

Gate description
Gate attributes

Histogram attributes
Histogram descriptions
Histogram information

ACRONYMS

EMF	enhanced metafile
FEP	Fault Tree, Event Tree, and Piping and Instrumentation Diagram Editors
INEEL	Idaho National Engineering and Environmental Laboratory
INL	Idaho National Laboratory
IPE	individual plant examination
IRRAS	Integrated Reliability and Risk Analysis System
MAR-D	Models and Results Database
NRC	Nuclear Regulatory Commission
PC	personal computer
PRA	probabilistic risk analysis
RTF	rich text format
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SARA	System Analysis and Risk Assessment
SETS	Set Equation Transformation System
WMF	Windows metafile

Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE)

Vol. 7 Data Loading Manual

1. INTRODUCTION

1.1 Background

The U.S. Nuclear Regulatory Commission (NRC) has developed a powerful personal computer (PC) software application for performing probabilistic risk assessments (PRAs), called Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE).

Using SAPHIRE on a PC, an analyst can perform a PRA for any complex system, facility, or process. Regarding nuclear power plants, SAPHIRE can be used to model a plant's response to initiating events, quantify associated core damage frequencies and identify important contributors to core damage (Level 1 PRA). It can also be used to evaluate containment failure and release models for severe accident conditions, given that core damage has occurred (Level 2 PRA). It can be used for a PRA assuming that the reactor is at full power, at low power, or at shutdown conditions. Furthermore, it can be used to analyze both internal and external initiating events, and it has special features for transforming models built for internal event analysis to models for external event analysis. It can also be used in a limited manner to quantify risk for release consequences to both the public and the environment (Level 3 PRA). For all of these models, SAPHIRE can evaluate the uncertainty inherent in the probabilistic models.

SAPHIRE development and maintenance has been undertaken by the Idaho National Laboratory (INL). The INL began development of a PRA software application on a PC in the mid 1980s when the enormous potential of PC applications started being recognized. The initial version, *Integrated Risk and Reliability Analysis System* (IRRAS), was released by the Idaho National Engineering Laboratory (now Idaho National Laboratory) in February 1987. IRRAS was an immediate success, because it clearly demonstrated the feasibility of performing reliability and risk assessments on a PC and because of its tremendous need (Russell 1987). Development of IRRAS continued over the following years. However, limitations to the state of the-art during those initial stages led to the development of several independent modules to complement IRRAS capabilities (Russell 1990; 1991; 1992; 1994). These modules were known as Models and Results Database (MAR-D), System Analysis and Risk Assessment (SARA), and Fault Tree, Event Tree, and Piping and Instrumentation Diagram (FEP).

IRRAS was developed primarily for performing a Level 1 PRA. It contained functions for creating event trees and fault trees, defining accident sequences and basic event failure data, solving system fault trees and accident sequence event trees, quantifying cut sets, performing sensitivity and uncertainty analyses, documenting the results, and generating reports.

MAR-D provided the means for loading and unloading PRA data from the IRRAS relational database. MAR-D used a simple ASCII data format. This format allowed interchange of data between PRAs performed with different types of software; data of PRAs performed by different codes could be converted into the data format appropriate for IRRAS, and vice-versa.

SARA provided the capability to access PRA data and results (descriptive facility information, failure data, event trees, fault trees, plant system model diagrams, and dominant accident sequences) stored in MAR-D. With SARA, a user could review and compare results of existing PRAs. It also provided the capability for performing limited sensitivity analyses. SARA was intended to provide easier access to PRA results to users that did not have the level of sophistication required to use IRRAS.

FEP provided common access to the suite of graphical editors. The fault tree and event tree editors were accessible through FEP as well as through IRRAS, whereas the piping and instrumentation diagram (P&ID) editor was only accessible through FEP. With these editors an analyst could construct from scratch as well as modify fault tree, event tree, and plant drawing graphical representations needed in a PRA.

Previous versions of SAPHIRE consisted of the suite of these modules. Taking advantage of the Windows 95 (or Windows NT) environment, all of these modules were integrated into SAPHIRE Version 6; more features were added; and the user interface was simplified.

With the release of SAPHIRE versions 5 and 6, INL included a separate module called the Graphical Evaluation Module (GEM). GEM provides a highly specialized user interface with SAPHIRE, automating SAPHIRE process steps for evaluating operational events at commercial nuclear power plants. In particular, GEM implements many of the accident sequence precursor (ASP) program analysis methods. Using GEM, an analyst can estimate the risk associated with operational events very efficiently and expeditiously.

This manual is designed to guide the user through the basic procedures necessary to enter PRA data into the SAPHIRE program using SAPHIRE's MAR-D ASCII-text (or "flat file") data formats. A simple sample database is presented in Section 3 that demonstrates the data loading process. Where applicable, the discussion includes how the processes for loading the sample database relate to the actual processes used to load a larger PRA or individual plant examination (IPE) database. The procedures in the manual were developed for use with SAPHIRE, Version 6 and Version 7, and may not apply to past or future versions. Procedures for version 6 and 7 are the same, except where noted. While this manual does provide guidance for efficient and accurate data entry, it is not intended to stand-alone but is meant to supplement existing documents. Therefore, this manual references the SAPHIRE User's Guide, the SAPHIRE Technical Reference Manual, and the SAPHIRE Tutorial as information sources.

1.2 Loading PRA Results Program Overview

There is a continual need for nuclear plant risk information documented in PRAs and IPEs. The INL has been under contract with the U.S. NRC to collect and load data and information for internally generated events from PRAs and IPEs into databases. The NRC programs and projects for which these databases are useful include (a) prioritization, evaluation, and resolution of generic safety issues, (b) risk monitoring of plants, (c) assessment of operational events (i.e., event analysis), and (d) evaluation of changes to technical specifications.

The databases were developed using SAPHIRE, which also were produced by the INL under contract with the NRC. Prior to version 6, SAPHIRE was actually a suite of programs:

1. Integrated Reliability and Risk Analysis System (IRRAS)

2. System Analysis and Risk Assessment (SARA)

3. Models and Results Database (MAR-D)

4. Fault Tree, Event Tree, and Piping and Instrumentation Diagram Editors (FEP)

Starting with version 6, the combined functionality of these programs has been integrated into a single, Windows based program called SAPHIRE. Note that SAPHIRE's data load and extract functionality is still referred to as the MAR-D interface, but is embedded within SAPHIRE.

Some electronic data generated by other computer applications can also be directly loaded (in ASCII format) into the SAPHIRE 6.0 and 7.0 software. For example, SAPHIRE can output and input data in the Set Equation Transformation System (SETS) format.

1.3 Assumptions and Recommendations

We assume that the SAPHIRE software has been loaded as described in the SAPHIRE User's Guide. We assume that the user is knowledgeable in the use of SAPHIRE. We also assume that the user has a basic level of knowledge concerning the use of event trees and fault trees in a PRA.

It is recommended that the user read Sections 1 and 2 of the SAPHIRE User's Guide. These sections provide an overview of SAPHIRE with discussions concerning how to get around in the program menus, and SAPHIRE database concepts. These concepts will be discussed only briefly in Section 2 of this document.

2. OVERVIEW OF DATABASE CONCEPTS

2.1 SAPHIRE Database Unit - The Project

The SAPHIRE analysis structure is divided into projects. Since access to any SAPHIRE database is obtained through the appropriate project, a project is the first thing that must be created. A project is any grouping of fault trees and event trees with their associated basic events, cut sets, reliability data, and descriptions. When a database project is created, a corresponding Windows folder, usually located beneath the Saf60 or Saphire7 folder, is also created (this assumes that SAPHIRE was installed in its default folder). When multiple projects are created, it is necessary to select one project to work with at a time. The procedures for adding and selecting a project in SAPHIRE are shown in Appendix A of this report.

SAPHIRE is structured so that major areas of functionality are grouped and accessed by main menu options. These main menu options will be referred to frequently throughout this manual. The main menu functions used predominantly in data loading are

- X File - options to create and select various projects.
- X Generate - options to transfer base case event data to current case data.
- X Fault Tree - options to create and modify fault tree logic, analyze and solve logic.
- X Event Trees - options to create and modify event tree and sequence logic.
- X Modify - options to edit descriptive and rate information, add and delete items.
- X Utility - options to extract and load data using flat files

2.2 Base Versus Current Case Concepts

A database in SAPHIRE consists of both a current (or working case) and a base case. These two analysis cases are not necessarily identical. When working in SAPHIRE, particularly in the report and utilities modules, it is a common option to select whether to use the base or current case for a particular activity. This concept is very important when dealing with cut set generation and quantification for both fault trees and event trees. It is possible to have two sets of values for basic events, cut sets, and importance. This option allows you to maintain a base case database that can be transferred to the current case (via the Generate menu). Once in the current case, the data can be changed as necessary for analysis, without losing the base-case data values and results.

All basic event data entered through the Modify menu is automatically placed in the base case database. When loading values, SAPHIRE will allow information to be input to the base case and/or the current case database. Unless otherwise selected during the process, any analysis performed using the SAPHIRE program defaults to values and/or cut sets drawn from the current case database. Note that basic fault tree and event tree logic remains the same for both cases.

2.3 File Management

There are several types of external files important to SAPHIRE for storing and accessing database information. All files associated with a particular database are stored in the subdirectory representing the project. The project subdirectory is found in the Saf60 or Saphire7 Windows folder.

The external relation files reside in the project subdirectory and maintain the permanent SAPHIRE interactive database. This type of data includes project, basic events, attributes, fault trees, event trees, end states, accident sequences, etc. For each relation, the following relation files exist:

*.BLK
*.DAT
*.DFL
*.IDX

These file types should never be deleted unless the project is to be removed permanently from the users hard drive. Appendix A of the MAR-D User's Guide[1] contains a more detailed description of these database relation files.

In addition to the relation files, SAPHIRE can also produce external *flat* or ASCII files. These can be extracted or loaded to or from the Windows project subdirectory using SAPHIRE software. These flat files, grouped according to the type of data they contain, are listed in Table 2-1. In version 6, extracted flat files will always be located in the project subdirectory. To be loaded into a different project, they must first be moved or copied (via Windows Explorer or similar method) into the destination project folder. In version 7, flat files can be extracted to any Windows folder, and can be loaded into the current project from any Windows folder as well.

Once the data contained in the flat files have been entered into the SAPHIRE database, they are stored permanently in the relation files. Therefore, flat files can be deleted from the subdirectory to conserve disk space and later extracted from the interactive database if necessary. For example, these flat files may be used to verify data entry. Another important use of these MAR-D files is using an extracted file as a template to add additional data to the database (i.e., via copy and paste type of editing functions).

There are two methods to create flat files. The first is to enter data into the interactive database using the Modify menu options. Once the data are entered manually, the flat files containing this information can be extracted, as described in Appendix A. The second is to create and enter data into an ASCII flat file with the correct format and file name (as shown in Appendix B). These files can then be loaded into the database, as described in Appendix A.

SAPHIRE also produces external report files. Report options are available in many sections of the software. The software allows the options to send reports to a printer, the computer screen, or to a file on any directory. Version 7 provides additional report formats that are compatible with major word processing software and browsers.

Note: Empty flat files can be extracted and serve as a template for the proper data entry format. These templates are available for those files listed in Table 2-1.

Table 2-1 SAPHIRE database file names and descriptions

File name	Version specific	Description	Applicable section in this manual
		Project Information	
ProjectName.FAD	6,7	Project name and descriptions	4.2.3
ProjectName.FAA		Project attributes	4.2.3
ProjectName.FAT	6,7	Project text	4.2.3
ProjectName.FAY		Project event tree recovery rules	-
ProjectName.FAS		Project fault tree recovery rules	-
ProjectName.FAP		Project end state partition rules	-
		Basic Event Information	
ProjectName.BED	6,7	Basic event names and descriptions	4.6.2
ProjectName.BEI		Basic event failure rates	4.6.3
ProjectName.BEA		Basic event attributes	4.6.3
ProjectName.BET	6,7	Event transformations and compound events	- -
ProjectName.BEC	7	Basic event compound information	-
ProjectName.BEN	7	Basic event notes	-
		Basic Event Attributes	
ProjectName.FMD		Failure mode descriptions	Appendix B
ProjectName.CTD		Component type descriptions	Appendix B
ProjectName.STD		System type descriptions	Appendix B
ProjectName.LCD		Location descriptions	Appendix B
ProjectName.TTD		Train descriptions	Appendix B
		Fault Tree Information	
FaultTree.DLS		Fault tree graphics	4.5.1
ProjectName.FTD	6,7	Fault tree names and descriptions	4.5.2
FaultTree.FTT	6,7	Fault tree text	4.5.2
ProjectName.FTA		Fault tree attributes	Appendix B
ProjectName.FTL		Fault tree logic	4.5.1
ProjectName.FTC		Fault tree cut sets	4.5.4
FaultTree.FTY		Fault tree recovery rules	-
FaultTree.PID		Fault tree P&ID	-
		Event Tree Information	
EventTree.ETG		Event tree graphics	4.3.1
ProjectName.ETD	6,7	Event tree names and descriptions	4.3.5
ProjectName.ETT	6,7	Event tree text	4.3.5
ProjectName.ETA		Event tree attributes	Appendix B
EventTree.ETL		Event tree logic	4.3.1
ProjectName.ETR		Event tree rules	4.3.4
EventTree.ETY		Event tree recovery rules	-

EventTree.ETP		Event tree end state partition rules	-
		End State Information	
ProjectName.ESD	6,7	End state names and descriptions	4.4.3
ProjectName.EST	6,7	End state text	4.4.3
ProjectName.ESC		End state cut sets	-
		Sequence Information	
ProjectName.SQD	6,7	Sequence names and descriptions	4.7.2
ProjectName.SQC		Sequence cut sets	4.7.1
ProjectName.SQA		Sequence attributes	Appendix B
ProjectName.SQT	6,7	Sequence text	4.7.2
ProjectName.SQL		Sequence logic	4.3.6
ProjectName.SQY		Sequence recovery rules	-
ProjectName.SQP		Sequence end state partition rules	-
		Gate Information	
ProjectName.GTD	6,7	Gate description	4.5.3
ProjectName.GTA		Gate attributes	4.5.3
		Change Set/Flag Set Information	
ProjectName.CSD	6,7	Change/flag set description	4.10.1/Appendix A
ProjectName.CSI		Change/flag set information	4.10.1/Appendix A
ProjectName.CSA	7	Change/flag set alternate names	-
		Histogram Information	
ProjectName.HIA		Histogram attributes	-
ProjectName.HID	6,7	Histogram descriptions	-
ProjectName.HII		Histogram information	-
ProjectName.HIA	7	Histogram alternate name	
		Slice Information	
ProjectName.SLA	7	Slice alternate names	-
ProjectName.SLB		Slice basic event logic	-
ProjectName.SLD		Slice description	-
ProjectName.SLI	6,7	Slice information (combo importance values)	-

3. THE SAMPLE DATABASE

This section presents the sample database used to describe the data loading process in Section 4. Section 3.1 presents the basic assumptions concerning use of this manual. Sections 3.2 through 3.9 contain the actual data and a discussion of the sample database.

3.1 The Sample Database

Several assumptions concern the presentation of the sample database:

1. The SAPHIRE software has been loaded as described in the SAPHIRE User's Guide.

2. The user has a basic knowledge of using SAPHIRE to analyze event trees and fault trees.

3. The user has read the sections of the SAPHIRE User's Guide that provide an overview of the use of the software and the program menus, modules, and database concepts.

In the SAPHIRE Tutorial, a simple example shows the quantification for the frequency of getting to work. The tutorial leads the student through (a) the basic construction of event tree and fault trees, (b) entering basic event data, and (c) generation and quantification of both fault tree and sequence cut sets. In this report, we use a modification of the simple "getting to work" example to demonstrate the data loading process. Sections 3.2 through 3.9 present the sample database in a fashion similar to that found in a typical PRA. However, unlike most PRAs, the sample database contains only those data essential to constructing a workable database in SAPHIRE.

3.2 The Sample Database Event Tree

Using failure-success logic, we developed an event tree to calculate the frequency that a worker will arrive on time, be late, or miss a day of work. The event tree (WORK) is shown in Figure 3-1. It was determined that the average working person is required to work approximately 248 days a year. In the WORK event tree, going to work was used as the initiating event (WORK). Initiating events are occurrences in a certain length of time that initiate a sequence of events. In this case, being required to get to work initiates the sequence of events leading to either getting to work on time, being late to work, or missing work completely.

The first event that should occur on a normal workday is that the alarm clock rings. Therefore, the first question to ask is "did the ALARM go off?" If it did not go off, then the worker will be late to work. If the alarm successfully wakes the worker, then a personal reason (i.e., sickness) may cause the worker to miss work. Therefore, the second question to ask is "did a PERSONAL reason make the worker miss work?" Thus, the ALARM may be successful but a PERSONAL reason may cause the worker to miss work. Now, either the alarm succeeded in waking the worker or the alarm failed and the worker woke up late, and if no personal circumstances cause the worker to miss work, then transportation problems may occur that causes the worker to be even later to work. Therefore, the third question to ask is "did the available transportation (TRNSPRT) fail?" Finally, if the alarm succeeded, no personal reasons interfered, and transportation was available, then the worker will be successful in getting to work on time.

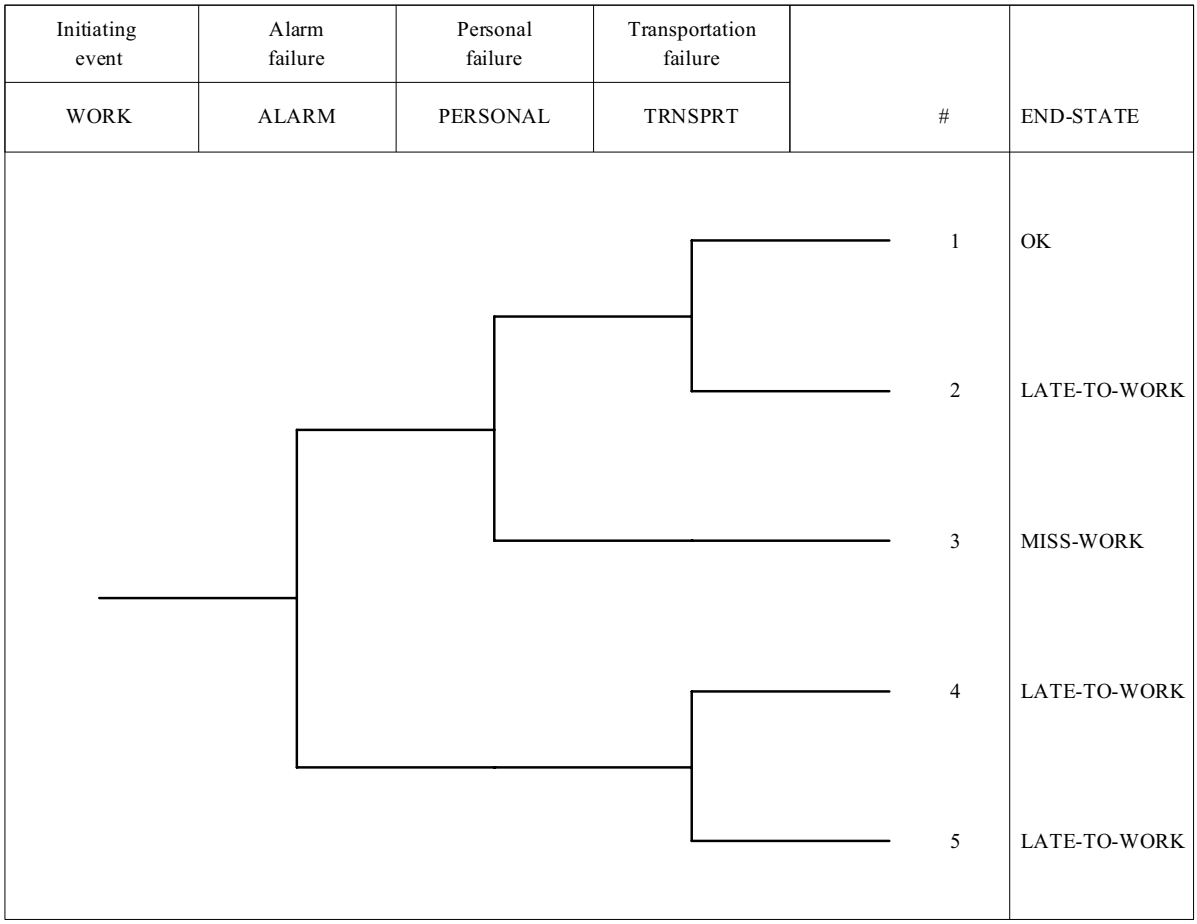


Figure 3-1 Going-to-work (WORK) event tree

We assume that the probability of public transportation (represented by the top event TRNSPRT) will change depending on the time that the person attempts to use this service. This assumption implies that the probability of failing TRNSPRT is conditional on the time that the service is needed. Therefore, if ALARM fails then it is necessary to substitute a different fault tree or probability for the original TRNSPRT top event. The database has another new fault tree called TRNSPRT-2. This fault tree will contain a different probability for the basic event that represents the failure of the public transportation fault tree when the demand for this service is later than the normal time to get to work.

The first four names along the topmost horizontal line of this figure represent the initiating event (WORK) and the top events (ALARM, PERSONAL, and TRNSPRT). Using the event tree in an analysis will enable the top events to be linked together. Standard practice depicts the initiating event as a horizontal line with fault trees connected in a branching structure, where an up branch indicates success and the down branch indicates failure. As the event tree logic is developed, a top event either can be passed (fault tree not questioned) or questioned (fault tree either succeeds or fails). Therefore, each pathway through an event tree has a combination of success, failure, or "pass" logic. This pathway of combinations is called a sequence. For example, following through the WORK event tree, sequence three (SEQ 3) is described as the success of ALARM, the failure of PERSONAL, and the pass of TRNSPRT.

3.3 The Sample Database Fault Trees

Each of the top events presented in the WORK event tree may be further developed as a fault tree or fault tree logic. Fault tree analysis is a technique where many events (basic events) that interact to produce a complex event (top event) can be related using logical relationships (AND, OR, etc.). This process permits the methodical building of a structure that can be used to analyze possible failures and to calculate the probability of failure. For this example, simple fault trees (shown in Figures 3-2 through 3-5) were developed. These fault trees are used to determine the probability of each top event occurring and to develop fault tree and sequence cut sets.

The ALARM fault tree (see Figure 3-2) is a representation of modeling alarm clock failure. Some common reasons for alarm clock failure include setting the wrong time, failing to set the alarm, mechanical failure, or power failure (either battery or commercial). The OR-gate ALARM has three inputs, one OR-gate, one AND-gate, and one undeveloped event. The OR-gate ALARM-1 has two basic events as input representing the probability of setting the wrong time or failing to set the alarm. Either of these events, the alarm being set to the wrong time [ALM-SWT (alarm-set wrong time)] or the alarm not being set [ALM-FTS (alarm fail to set)], can fail the alarm clock. The undeveloped event under the OR-gate ALARM, ALM-MECH (ALARM-mechanical failure), will represent the probability of any of the mechanical functions associated with the alarm failing. Any mechanical failure will prevent the alarm from performing its function. The AND-gate ALARM-2 has two basic events as inputs representing the probability that power has failed to the alarm. It is necessary that both the commercial power [ALM-CPF (alarm-commercial power failure)] and the battery [ALM-BPF (alarm-battery power failure)] not work to fail the alarm power.

The PERSONAL fault tree (Figure 3-3) is a simple representation modeling personal or human failure that results in missing work. Two common reasons for failure include sickness or sickness in family. There are also many additional reasons for personal failure that are less common occurrences than sickness related failures. The OR-gate PERSONAL has three inputs; two basic events and one undeveloped event. The two basic events will represent the probability of either missing work due to being sick (SICK) or family illness [SICK-FAM (sickness in family)]. The undeveloped event OTHER represents the probability that other personal reasons are responsible for failure.

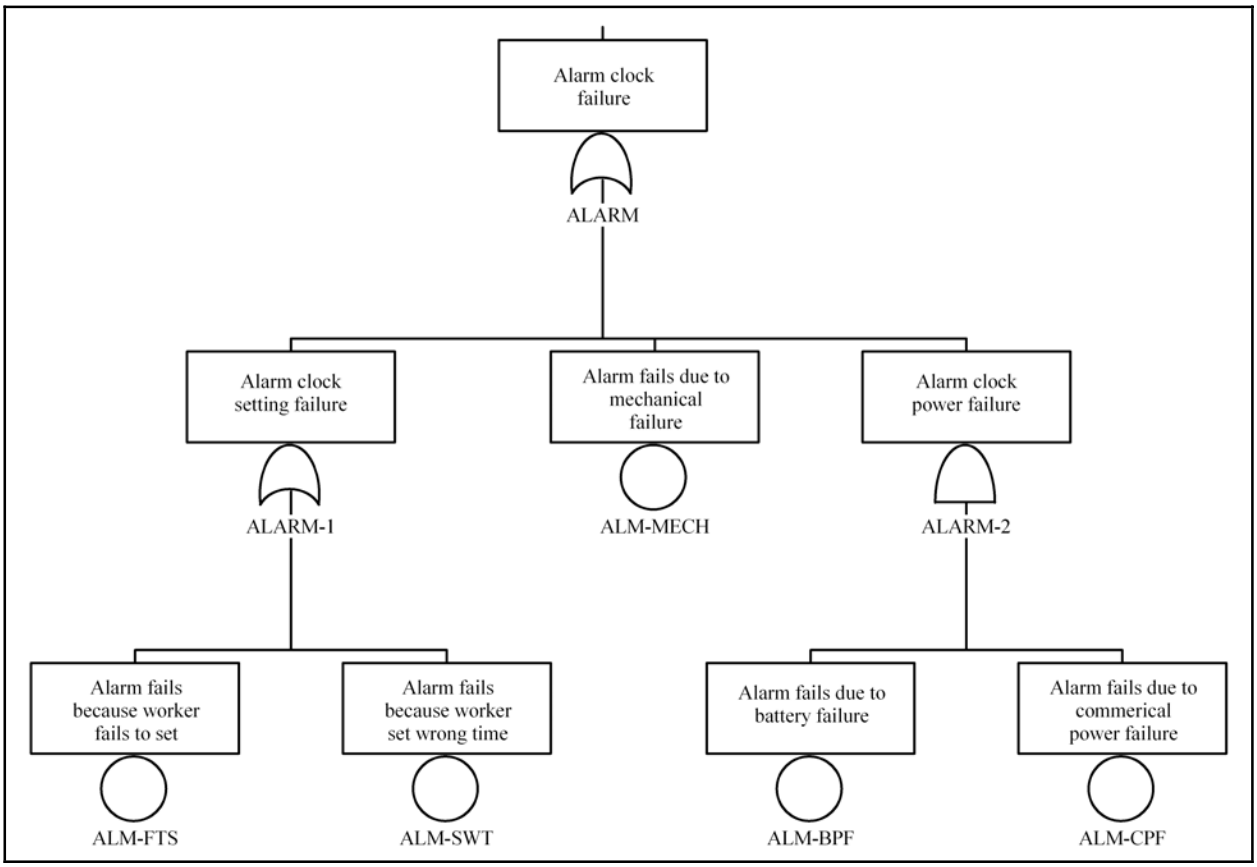


Figure 3-2 Alarm clock failure fault tree

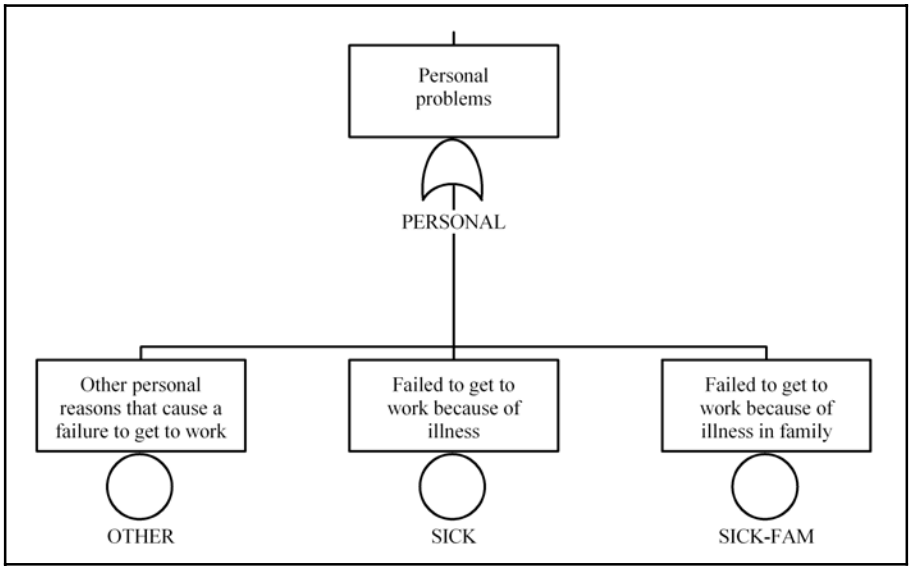


Figure 3-3 Personal problems fault tree

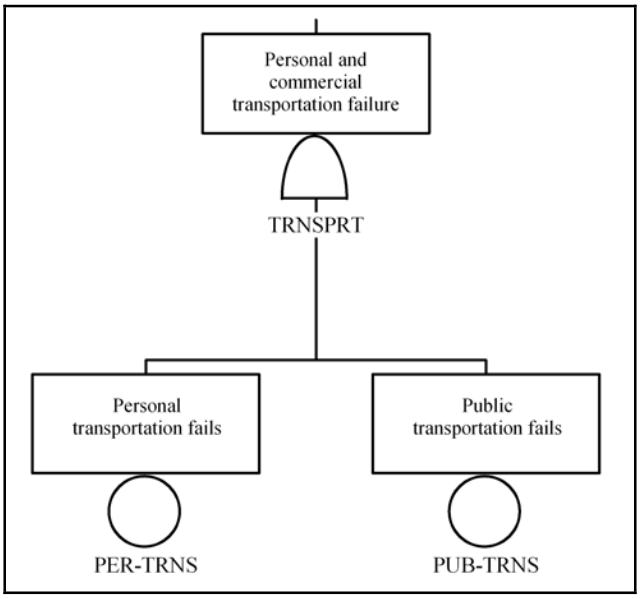


Figure 3-4 Transportation failure fault tree (normal time frame)

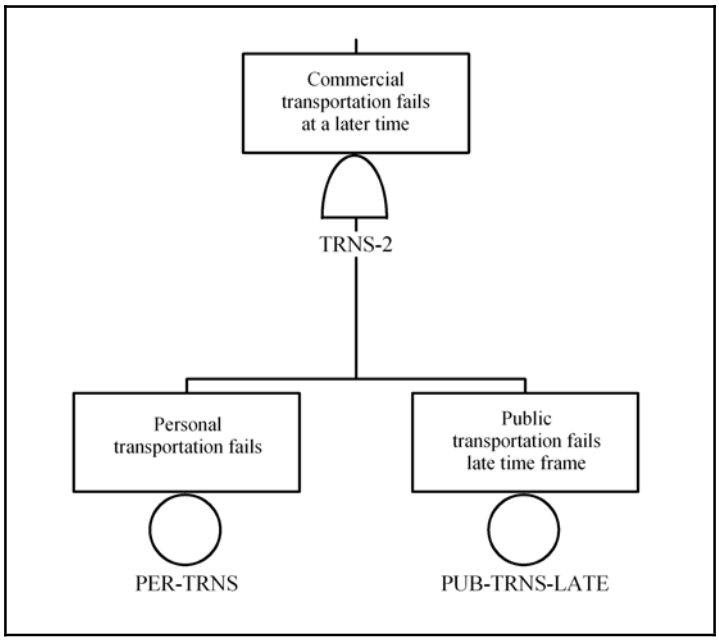


Figure 3-5 Transportation failure fault tree (late time frame)

The third fault tree TRNSPRT (Figure 3-4) is a simple representation modeling transportation failure. Two common modes of transportation include personal (such as a car) and public (such as a bus or train). The AND-gate TRNSPRT has two basic events as inputs. The two basic events will represent the probability of public transportation [PUB-TRNS (public transportation)] and personal transportation [ER-TRNS (personal transportation)] failure.

An additional fault tree TRNSPT-2 (Figure 3-5) is a modification of the TRNSPRT fault tree. Since the probability of obtaining public transportation is dependent upon the time of day, this fault tree is a representation modeling transportation at a time later than normal. In this situation, the probability of public transportation failing is less due to the lower demand. Then, if ALARM fails, the worker needs public transportation later than if the ALARM had succeeded. In this scenario, it is necessary to substitute a fault tree for the TRNSPRT top event (TRANSPT-2) that contains the probability of failure of the public transportation fault tree in a later period.

3.4 The Sample Database Basic Events

Information on the basic event values and descriptions for the sample problem is provided in **Table 3-1** and **Table 3-2**. The table provides the necessary basic event and initiating event information to duplicate the analysis performed on this problem. Typically, PRAs contain more basic event information (e.g., fault tree type, failure mode) that will need to be entered into the database to complete the analysis. Note that the uncertainty value contained in **Table 3-1** is the lognormal distribution error factor.

Table 3-1 Basic event values for the sample problem

Basic event	Distribution type	Calculation type	Mean value	Uncertainty value
ALM-BPF	Lognormal	1	9.0E&4	3
ALM-CPF	Lognormal	1	1.5E&2	3
ALM-FTS	Lognormal	1	5.5E&3	10
ALM-MECH	Lognormal	1	2.7E&4	3
ALM-SWT	Lognormal	1	2.7E&3	10
MEDICINE	Lognormal	1	8.1E&3	5
OTHER	Lognormal	1	5.0E&1	10
PER-TRNS	Lognormal	1	5.5E-3	3
PUB-TRNS	Lognormal	1	2.7E&3	3
PUB-TRNS-LATE	Lognormal	1	2.0E&3	3
SICK	Lognormal	1	8.1E&3	10
SICK-FAM	Lognormal	1	4.0E&3	10
WORK	Lognormal	1	2.48E+2/yr	10

Table 3-2 Basic event descriptions for the sample problem

Basic event	Description
ALM-BPF	Alarm fails due to battery failure
ALM-CPF	Alarm fails due to commercial power failure
ALM-FTS	Alarm fails because worker failed to set alarm
ALM-MECH	Alarm fails due to mechanical failure
ALM-SWT	Alarm fails because worker set wrong time
MEDICINE	Recovery for sickness preventing attending work
OTHER	Other personal reasons that cause a failure to get to work
PER-TRNS	Personal transportation
PUB-TRNS	Public transportation fails
PUB-TRNS-LATE	Public transportation fails late time frame
SICK	Failed to get to work because of illness
SICK-FAM	Failed to get to work because of illness in family
WORK	Event tree (WORK) initiating event

Since the sample database is simplified compared to traditional PRA databases, no advanced external event analysis features are covered. Consequently, fire, flood, and seismic analysis are not directly addressed by way of the sample database. However, details for data loading and manipulation for seismic analysis are presented in Appendix D.

3.5 Sample Database Fault Tree Cut Sets

Shown in Table 3-3 are the fault tree cut sets and minimal cut set (mincut) upper bound for those fault trees contained in the sample database. The fault tree modeling of "personal failure due to sickness and other reasons" has the greatest probability of occurring.

Table 3-3 Fault tree cut set results

Fault Tree: ALARM
Mincut Upper Bound: 2.705E-3

Cut No.	Total (%)	Set (%)	Probability	Cut sets
1	99.8	99.8	2.7E&3	ALM-SWT
2	100.0	0.2	5.5E&6	ALM-FTS
3	100.0	0.0	2.7E&8	ALM-MECH
4	100.0	0.0	1.3E&9	ALM-BPF, ALM-CPF

Fault Tree: PERSONAL
Mincut Upper Bound: 2.007E-2

Cut No.	Total (%)	Set (%)	Probability	Cut sets
1	40.3	40.3	8.1E&3	OTHER
2	80.7	40.3	8.1E&3	SICK
3	100.0	19.9	4.0E&3	SICK-FAM

Fault Tree: TRNS-2
Mincut Upper Bound: 1.100E-5

Cut No.	Total (%)	Set (%)	Probability	Cut sets
1	100.0	100.0	1.1E&5	PER-TRNS, PUB-TRNS-LATE

Fault Tree: TRNSPRT
Mincut Upper Bound: 1.485E-5

Cut No.	Total (%)	Set (%)	Probability	Cut sets
1	100.0	100.0	1.4E&5	PER-TRNS, PUB-TRNS

3.6 Sample Database Sequence Cut Sets

Shown in Table 3-4 are the cut sets and frequencies for the sequences from the WORK event tree. Since Sequence 1 represents successfully getting to work, it is not presented. Sequence 3 is the largest and only contributor to missing work. Sequence 4 is the largest contributor to being late-to-work.

Table 3-4 Sequence cut set results

Sequence: 2 (calculated frequency = 3.68E-3/yr)

Cut set	Frequency	Cut set
1	3.7E&3	WORK, PER-TRNS, PUB-TRNS

Sequence: 3 (calculated frequency = 3.99/yr)

Cut set	Frequency	Cut set
1	2.0E+0	WORK, OTHER
2	1.0E+0	WORK, SICK, MEDICINE
3	9.9E&1	WORK, SICK-FAM

Sequence: 4 (calculated frequency = 6.71E-1/yr)

Cut set	Frequency	Cut set
1	6.7E&1	WORK, ALM-SWT
2	1.4E&3	WORK, ALM-FTS
3	6.7E&6	WORK, ALM-MECH
4	3.3E&7	WORK, ALM-BPF, ALM-CPF

Sequence: 5 (calculated frequency = 7.38E-6/yr)

Cut set	Frequency	Cut set
1	7.4E&6	WORK, ALM-SWT, PER-TRNS, PUB-TRNS-LATE
2	1.5E&8	WORK, ALM-FTS, PER-TRNS, PUB-TRNS-LATE
3	7.4E&11	WORK, ALM-MECH, PER-TRNS, PUB-TRNS-LATE
4	3.7E&12	WORK, ALM-BPF, ALM-CPF, PER-TRNS, PUB-TRNS-LATE

3.7 Sample Database Recovery Actions

Sequence 3 shown in the sequence cut set list in Table 3-4 accounts for most of the days lost at work (4.0 times per year). Notice that a basic event, MEDICINE, has been added to the cut set containing sick. It was anticipated that 50% of the time it might be possible that an individual will take medicine and feel well enough to attend work. MEDICINE is a recovery action added *after* the sequence cut set generation.

3.8 Sample Database Uncertainty

The following tables summarize the sequence, fault tree, and end state uncertainty. All uncertainties were performed using a Monte Carlo simulation with 1,000 samples (using seed 123). Table 3-5 lists the uncertainty results for the fault trees, Table 3-6 lists the uncertainty results for each of the sequences, while Table 3-7 list the uncertainty results for the end states.

Table 3-5 Fault Tree uncertainty values report

Fault Tree	Mean	Min. Cut Upper Bound	Median	Std. Dev.	5th %	95th %	Minimum	Maximum
ALARM	2.62E-03	2.71E-03	9.80E-04	5.09E-03	1.20E-04	1.05E-02	1.97E-05	6.27E-02
PERSONAL	1.97E-02	2.01E-02	1.26E-02	2.37E-02	2.75E-03	5.81E-02	3.61E-04	2.37E-01
TRNS-2	1.07E-05	1.10E-05	5.31E-06	1.70E-05	7.18E-07	3.61E-05	1.42E-07	2.15E-04
TRNSPRT	1.44E-05	1.49E-05	7.16E-06	2.29E-05	9.69E-07	4.87E-05	1.92E-07	2.90E-04

Table 3-6 Sequence uncertainty values report

Event Tree Seq	Mean	Min. Cut Upper Bound	Median	Std. Dev.	5th %	95th %	Min	Max
WORK 2	3.83E-03	3.68E-03	1.67E-03	7.10E-03	2.14E-04	1.53E-02	6.50E-06	1.22E-01
WORK 3	3.46E+00	3.99E+00	1.96E+00	4.88E+00	3.62E-01	1.10E+01	5.85E-02	6.80E+01
WORK 4	7.42E-01	6.71E-01	2.32E-01	2.05E+00	2.15E-02	2.73E+00	1.43E-03	2.84E+01
WORK 5	6.73E-06	7.38E-06	1.26E-06	1.94E-05	5.33E-08	3.02E-05	8.89E-10	2.78E-04

Table 3-7 End state uncertainty values report

End State	Mean	Min. Cut Upper Bound	Median	Std. Dev.	5th %	95th %	Min	Max
LATE-TO-WORK	6.84E-01	6.75E-001	2.47E-01	1.49E+00	2.296E-02	2.56E+00	1.21E-03	2.21E+01
MISS-WORK	3.46E+00	3.99E+000	1.96E+00	4.88E+00	3.619E-01	1.10E+01	5.85E-02	6.80E+01

3.9 Sample Database Importance

The following is a report on the Fussell-Vesely importance measure for each basic event over the total end-state database analysis. Table 3-8 shows the results of the importance analysis for the sample database.

Table 3-8 Results of sample database importance analysis

Basic Event	Number of Occurrences	Probability	Fussell-Vesely	Risk Reduction Ratio	Risk Increase Ratio
WORK	12	2.480E+02	1.000E+00	----------	4.032E-03
OTHER	1	8.100E-03	4.276E-01	1.747E+00	5.337E+01
MEDICINE	1	5.000E-01	2.129E-01	1.271E+00	1.213E+00
SICK	1	8.100E-03	2.129E-01	1.271E+00	2.708E+01
SICK-FAM	1	4.000E-03	2.103E-01	1.266E+00	5.337E+01
ALM-SWT	2	2.700E-03	1.437E-01	1.168E+00	5.408E+01
PER-TRNS	5	5.500E-03	7.919E-04	1.001E+00	1.143E+00
PUB-TRNS	1	2.700E-03	7.904E-04	1.001E+00	1.292E+00
ALM-FTS	2	5.500E-06	2.919E-04	1.000E+00	5.408E+01
PUB-TRNS-LATE	4	2.000E-03	1.584E-06	1.000E+00	1.001E+00
ALM-MECH	2	2.700E-08	1.433E-06	1.000E+00	5.408E+01
ALM-CPF	2	1.500E-02	7.166E-08	1.000E+00	1.000E+00
ALM-BPF	2	9.000E-08	7.166E-08	1.000E+00	1.796E+00

4. LOADING THE SAMPLE DATABASE

4.1 Introduction

This section describes the process of loading the sample database presented in Section 3. The section is organized to reflect the methodology that has proven useful while working with actual PRA data. The section organization is as follows:

Section 4.2 Adding and Selecting the Database Project

Section 4.3 Loading the Event Tree Data

Section 4.4 Entering End State Data

Section 4.5 Loading the Fault Tree Data

Section 4.6 Loading Basic Event Data

Section 4.7 Loading Sequence Data

Section 4.8 Recovery Actions

Section 4.9 Analyzing Uncertainty

Section 4.10 Additional Features

Each section presents methods used for entering a specific type of data (there may be several methods possible). The merits of each data entry method are discussed and a brief overview of the actual steps used to enter the data using this method is presented. Manuals and guides that may add useful information to the method are also cited.

4.2 Adding and Selecting the Database Project

The necessary first step in loading a database is adding and/or selecting the project that will contain the database. Adding and selecting the database project includes

1. Adding the project (Section 4.2.1)
2. Selecting the project (Section 4.2.2)
3. Entering project information and text (Section 4.2.3)
4. Extracting and verifying the project flat files (Section 4.4.4).

4.2.1 Adding the Project

The SAPHIRE database structure is divided into projects. Since access to the SAPHIRE interactive database is obtained through the appropriate project, a project is the first thing that must be added. A project is any logical grouping of fault trees and event trees with their associated basic events, cut sets, reliability data, and descriptions. The project concept allows for the separation of any number of distinct databases. When a database project is created in one of the SAPHIRE programs, a corresponding Windows subfolder contained in the Saf60 or Saphire7 subdirectory is also created.

To add a project, use the New Project option. The procedure is shown in detail in Appendix A. The procedure requires giving the project a name and assigning the project to a folder.

4.2.2 Selecting the Project

Once a project has been added, it is automatically selected as the current project. It will remain the current project until you select a different one. The procedure to select a project is shown in detail in Appendix A.

The procedure requires selecting the File → Open Project menu option, navigating to the desired project folder, and selecting the project file name located in the project folder.

4.2.3 Entering Project Information, Description, and Text

Project description, information, and text can be entered into the database when the project is selected. To add the project description (“This is a sample database”) or the project text (text ("A simple example that models the probability of getting to work on time"), choose the Modify → Project option from the SAPHIRE main menu to open the Modify Project dialog (see Figure 4-1). Project information that can be stored using this option includes location, company, type, design, vendor, tree type, seismic histograms, operational date, qualification data, and mission time.

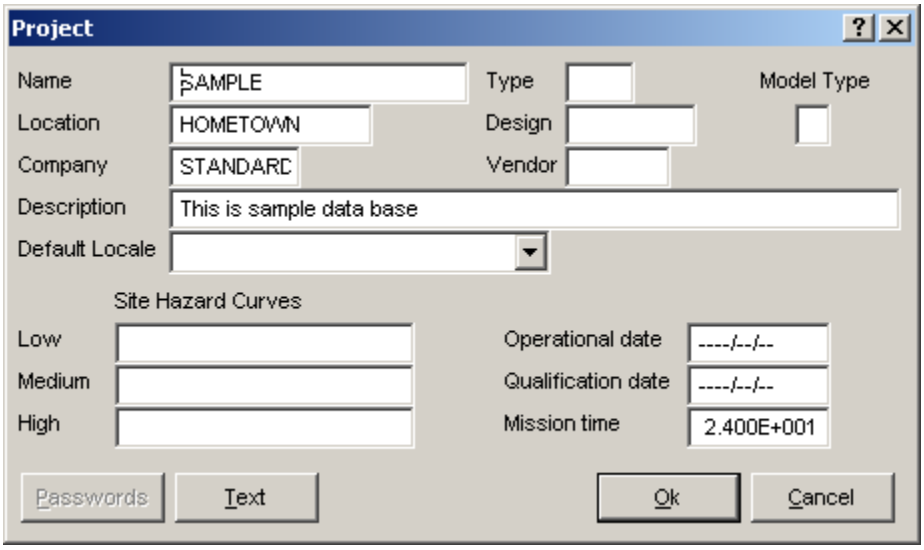


Figure 4-1 Modify project dialog

Below are the available methods for entering project information, description, and text.

Interactive Modify Project Method

The first step is to use the interactive database to enter the data. The procedure for using the interactive database is described above.

Load from Project Flat Files Method

Second, the .FAA, .FAD and .FAT project flat files can be extracted as shown in Table 4-1 and used as a template to enter information using a text editor. The procedure for using extracted flat files is as follows:

1. Extract the project files, .FAA, FAD, and .FAT (the extract and load processes are described in detail in Appendix A).
2. Use an editor to modify and add the project data to the extracted files. A detailed description of the flat file format is available in Appendix B.
3. Load the finalized files back into the interactive database. The interactive database should now contain the project data.

4.2.4 Extracting and Verifying the Project Data

It is often necessary to verify that data items are accurate. The SAPHIRE flat files are particularly useful for this task. The flat files extracted from the sample database (shown in Table 4-2) can be used to verify the project information entered in the interactive database. Notice that not all the possible entry fields (e.g., Design) have been filled. Many options are provided in SAPHIRE that may not be applicable to every database, and, subsequently, some areas may be blank.

Table 4- 1 Extracted project flat files

File	Extracted file information
.FAA	SAMPLE =
	* Name , Mission, NewSum, Company, Location, Typ, Design, Vendr, Arch Eng, OpDate, QualDate
	SAMPLE, 2.400E+001,-----E---, , , , , , , ,----/--/--
.FTD	SAMPLE ,
.FAT	SAMPLE =

Table 4- 2 Extracted project flat files

File	Extracted file information
.FAA	SAMPLE =
	* Name , Mission, NewSum, Company, Location, Typ, Design, Vendr, Arch Eng, OpDate, QualDate
	SAMPLE , 2.40E+001, -----E---, STANDARD, HOMETOWN , , , , ,----/--/--,----/--/--
.FTD	SAMPLE ,This is sample data base
.FAT	SAMPLE =
	A simple example that models the probability of getting to work on time.

4.3 Loading the Event Tree Data

The next step in loading a database is to enter the database event trees and verify their accuracy. The event-tree data entry is complicated by the fact that the SAPHIRE software uses an interactive database. Information entered during the process of graphical event tree construction will appear in other areas of the program.

Those event trees that contain an initiating event will be listed in the Event Tree main menu option at all times. Event trees without an initiating event will be included in the event tree list only when the Show Sub Trees option is checked. Popup menu options may vary according to whether the Show Sub Trees option is checked or not.

Top events can be found in the Fault Tree main menu option, as well as in the Modify Fault Trees main menu option. Top events are also listed as developed basic events in the Modify main menu option, while initiating events are only listed as basic events. The information in any of these internal lists can subsequently be extracted into SAPHIRE flat files.

It is not necessary to enter the event trees at this point, but it has proved to be the most efficient method for entering nuclear power plant PRAs. The sample database event tree to be loaded is shown in Figure 4- 2.

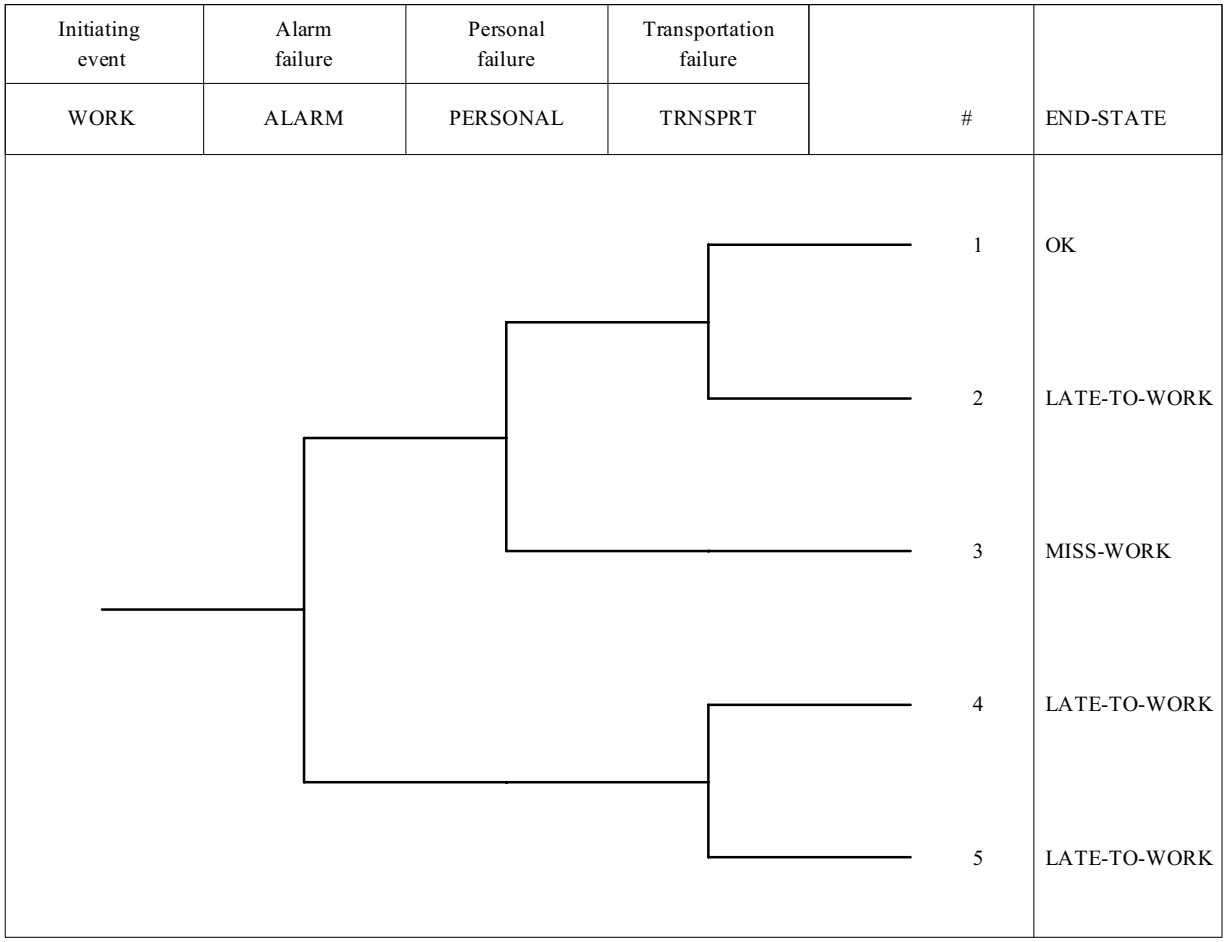


Figure 4- 2 Going to work event tree graphic

The process of loading an event tree includes

1. Entering the event tree structure (Section 4.3.1)
2. Entering sequence names in graphics (Section 4.3.2)
3. Entering top event descriptions (Section 4.3.3)
4. Entering link (substitution) rules (Section 4.3.4)
5. Entering event tree descriptions and text (Section 4.3.5)
6. Generating and verifying the event tree logic (Section 4.3.6)

4.3.1 Entering the Event Tree Logic

In the sample database, event tree logic is used as the basis for linking system fault trees and generating sequence logic to generate sequence cut sets. Some types of databases may not use event trees, but they are typically used to varying degrees in PRA methodology. SAPHIRE was originally designed to handle the more common type of approach, the large fault tree/small event tree approach, represented by the sample database. Other databases may use the large event tree/small fault tree approach. SAPHIRE can also handle the large event tree type of database. Note that SAPHIRE does not accept other software's event tree graphics; therefore, each event tree will have to be created individually.

Below are the available methods for entering the event tree logic.

Event Tree Graphical Editor Method

The most efficient method to load event tree logic is to enter the event tree structural information into SAPHIRE in the event tree graphical editor. It is straightforward to enter event tree logic into the graphical editor. The process of entering and saving an event tree similar to the sample database is discussed in detail in the SAPHIRE Tutorial.

The event tree creation procedure requires you to (a) select the Event Tree main menu option, (b) add an event tree by right clicking to choose the Add Event Tree popup menu option, (c) invoke the event tree graphical editor by right clicking to choose the Edit Graphics popup menu option, and (d) from there, entering the event tree structure, as shown in Figure 4- 2.

Load from Event Tree Logic Flat File Method

It is possible, but may be difficult, to enter the event tree graphic logic into a flat file and then load this file into SAPHIRE. As the development of the small WORK event tree is presented in the following sections, it will be obvious that the more highly branched the event tree becomes, the more confusing the resulting logic. Therefore, this method is not discussed.

Once an event tree is created, any of the flat files for this tree can be extracted. At this stage, many of the extracted flat files shown in Table 2-1 (Section 2.4) will not contain data other than the event tree and project name. The flat files that will contain data are the event tree graphics (.ETG) and the event tree logic (.ETL) shown in Table 4-3. These two files are identical in SAPHIRE.

Table 4-3 Extracted event tree flat files (with logic and sequence names only)

Files	Extracted file information
.ETG and .ETL	SAMPLE, WORK, IE-WORK = ^WINVER1.0 ^TOPS ALARM, PERSONAL, TRNSPRT ^LOGIC +1 +2 +3 -3 -2 3 -1 2 +3 -3 ^SEQUENCES N, SEQUENCE-NAMES, Y, END-STATE-NAMES, N, Frequency, N, EXTRA-#1, Y, A, Y, , Y, , Y, , Y, B, Y, , Y, , Y, , Y, C, Y, , Y, , Y, , Y, D, Y, , Y, , Y, , Y, E, Y, , Y, , Y, , ^PARMS START 0.00, 300.00 WINDOW 0.00, 0.00, 600.00, 600.00 ASPECTRATIO 0.74 HEADER 499.50, 499.50, 625.50, 751.50 STRING E DEFFONT 5 TOPWIDTH 10 TOPSIZE -13.00 TOPFONT 1 TOPFACE Times_New_Roman TOPPITCH 2 TOPCOLOR 15 DESHITE 3 DESSIZE -13.00 DESFONT 5 DESFACE Times_New_Roman DESCOLOR 15 DESPITCH 2 NODEHITE 25.00 ENDSIZE -13.00 ENDFONT 1 ENDFACE Times_New_Roman ENDPITCH 2 ENDCOLOR 15 BACKCOLOR 1 TOPBACKCOLOR 1 LINECOLOR 15 HILITECOLOR 1 LOCALE 1033 MODDATE 2003/10/08

Note:	
X	When saving an event tree graphics file, verify that the file name is the same name as the desired event tree name.
X	Remember that event trees cannot transfer to the middle of other event trees.
X	When possible, for ease of identification, identify initiating events by prefixing their names with the letters IE; for example, IE-xx.
X	Give all event trees unique names for identification and tracking. It may be useful to include the project name, the event tree name, and the document-related page number.

4.3.2 Entering Sequence Names in Graphics

Event tree sequence names are automatically and arbitrarily named in the graphical editor. Although the user can modify these names, they are merely placeholders for editing purposes, and will not be used further in any form by SAPHIRE, whether renamed by the user or not. Therefore, it is not recommended that the user modify sequence names. Table 4-3 shows the event tree graphics flat files that include the automatically named sequences (A, B, C, etc).

Sequences do not appear in the interactive database apart from the event tree graphical editor until the event tree/sequence logic has been linked. The linking step occurs after the event tree logic has been created and will be discussed in a later section. SAPHIRE generates a different, unique name for each event tree sequence when the logic is linked.

4.3.3 Entering Top Event Descriptions

Descriptions of top events are commonly found, as was shown in Figure 4- 2. They normally appear above the top event designator. Top event descriptions can either be added in the graphics option or the .ETG can be modified using a text editor. Table 4-4 shows the event tree graphics flat files that include the top event descriptions.

Table 4-4 Event tree file (with logic, sequence, end state name, and top event descriptions)

File	Extracted file information
.ETG	SAMPLE, WORK, I.E.,-WORK =
	^WINVER1.0
and	^TOPS
	ALARM, PERSONAL, TRNSPRT
.ETL	^LOGIC
	+1 +2 +3
	-3
	-2 3
	-1 2 +3
	-3
	^SEQUENCES
	N, SEQUENCE-NAMES, Y, END-STATE-NAMES, N, Frequency, N, EXTRA-#1,
	Y, A, Y, OK, Y, , Y, ,
	Y, B, Y, LATE-TO-WORK, Y, , Y, ,
	Y, C, Y, MISS-WORK, Y, , Y, ,
	Y, D, Y, LATE-TO-WORK, Y, , Y, ,

```
Y, E, Y, LATE-TO-WORK, Y, , Y, ,
^TOPDESC
 "Initiating ",
 "event"
!
 "Alarm ",
 "failure"
!
 "Personal ",
 "failure"
!
 "Transportation ",
 "failure"
!
^PARMS
START 18.00, 300.00 WINDOW 63.00, -100.54, 609.00, 445.46
ASPECTRATIO 0.74
HEADER 427.50, 513.00, 553.50, 679.50
STRING DEFFONT
DEFFONT 5
TOPWIDTH 10
TOPSIZE -13.00
TOPFONT 1
TOPFACE Times_New_Roman
TOPPITCH 18
TOPCOLOR 0
DESHITE 3
DESSIZE -13.00
DESFONT 5
DESFACE Times_New_Roman
DESCOLOR 0
DESPITCH 18
NODEHITE 90.16
ENDSIZE -13.00
ENDFONT 1
ENDFACE Times_New_Roman
ENDPITCH 18
ENDCOLOR 0
BACKCOLOR 15
TOPBACKCOLOR 15
LINECOLOR 0
HILITECOLOR 1
LOCALE 1033
MODDATE 2003/10/08
```

Below are the available methods for entering top event descriptions.

Graphical Event Tree Editor

Top event descriptions are easily entered in the graphics editor. This is potentially the most time consuming but the most straightforward. In this case, the event tree is finalized and files extracted without any intermediate step.

The procedure for entering the top event description into event tree graphics is to (a) select the desired top, and (b) enter the top event descriptions in the appropriate text box. In depth procedures for adding top event descriptions using the graphics editor is provided in the SAPHIRE Tutorial.

Load from Event Tree Logic Flat File

Top event descriptions also can be entered into the event tree flat file (.ETG or .ETL) that was extracted from the SAPHIRE program using a text editor (see Table 4-4). After modification, both files must be loaded as described in Appendix A. This may be the fastest method available but requires steps that are more substantial and is prone to errors since the information needs to be reloaded. This method is not discussed further.

4.3.4 Entering Link (Substitution) Rules

Substitutions of different fault trees or top event probabilities are very commonly used in event tree logic. In this sample problem, for example, there may be a different probability of failure for the transportation, depending on whether the alarm succeeds or fails. As discussed in Section 3, this is due to the increased availability of later public transportation. SAPHIRE uses link rules to allow substitutions of event tree top events. Table 4-5 shows the ETR file.

Link Rule Editor Method

Link rules can be entered from the Edit Rules menu option. This is the most straightforward and simplest, particularly when the rules are short.

The procedure for entering the link rules is to (a) choose the Event Tree main menu option, (b) highlight the desired event tree, (c) choose the Edit Rules option from the popup menu, and (d) enter the rule text.

Load from Event Tree Link Rule Flat File Method

Link rules can also be entered into an event tree rule flat file using the SAPHIRE format. After the file is developed, it is necessary to load this file. (The loading procedure is discussed in Appendix A.) This method may be the fastest (particularly with a large group of rules) but requires more substantial steps and is prone to errors since the rule information needs to be reloaded. This methodology is not presented. Note that once a rule has been entered for an event tree, the .ETR flat file can be extracted for use as a template for subsequent rules.

Table 4- 5 Extracted event tree rules flat file

File	Extracted file information

.ETR	SAMPLE, WORK=
	| rule to substitute TRNS-2 for TRNSPRT
	if ALARM then
	  TRNSPRT = TRNS-2;
	endif

4.3.5 Entering Event Tree Descriptions and Text

Many PRAs contain descriptions and extensive text concerning the event trees in the analysis. The sample database has an event tree description (WORK EVENT TREE) and text just for demonstration purposes. Table 4- 6 shows the extracted flat files containing the descriptions and text.

Below are the available methods for entering event tree descriptions and text.

Interactive Modify Event Trees Method

Event tree descriptions and text can be entered in the Modify → Event Tree main menu option. This is perhaps the easiest method since there are usually a limited number of event trees and it is done entirely within the SAPHIRE environment. Though it may be slower than the other method discussed here, it is recommended for most situations. Procedures for adding descriptions and text are in the SAPHIRE User's Guide.

The procedure for entering the event tree descriptions and text is to (a) select the Modify → Event Tree main menu option, (b) highlight the desired event tree, and (c) choose the Modify popup menu option to enter the description, or press the Text button to enter the event tree text.

Load from Event Tree Description Flat File Method

The description data can be entered into the event tree description flat file (.ETD) extracted from the SAPHIRE program, using a text editor. The event tree textual data can be entered into the event tree text flat file (.ETT) using the SAPHIRE format. (This is also true of the .ETD). After modification or development, both files must be loaded as described in Appendix A. This method is not discussed further.

Table 4- 6 Extracted event tree description and text flat files

File	Extracted file information
.ETD	SAMPLE = WORK ,WORK EVENT TREE
.ETT	SAMPLE, WORK= A FAIL-SUCCESS LOGIC WAS USED TO DEVELOP AN EVENT TREE TO CALCULATE THE FREQUENCY THAT THE AVERAGE PERSON WILL ARRIVE ON TIME, BE LATE OR MISS A DAY OF WORK.

4.3.6 Generating and Verifying Event Tree Logic

Basic event tree logic is verified using either the graphics visual picture or by linking trees to generate sequence logic and examining the results of the sequence generation process. We recommend that both these processes be performed after creating an event tree and entering all the associated data. The sequence logic flat file is shown in Table 4-7. The methods discussed below allow verification of all the data entered, as described in the previous section.

Below are the available methods for generating and verifying event tree logic.

Review Graphical Output Method

A graphical output can be obtained for each event tree. This graphic output can be sent directly to a printer, or to a Windows metafile (WMF), enhanced metafile (EMF) or rich text (RTF) file. Note that the graphical output can be verified as accurate, but any link rules will need to be examined.

The procedure for obtaining a copy of the event tree graphic requires you (a) enter the event trees graphical editor, and (b) select the Print option to print, or Save As option to create a WMF, EMF, or RTF file. To print or save multiple event trees at once, (a) select the Report main menu option, (b) select the Event Tree and Graphic radio buttons and press the Process button, then (c) select the desired event trees and press the Print or Export button.

Link Trees Method

In the process of linking trees, sequence logic will be generated, and event tree logic can be verified. This process produces the sequence logic that will be used by the interactive database to produce sequence cut sets.

The procedure for generating sequences and obtaining a printout for verification requires the following:

1. Choose the Event Tree main menu option.

2. Select the event tree(s) to link.

3. Choose the Link Trees popup menu option to open the Event Tree - Sequence Logic Generate dialog.

4. Select the Report Option to Send Report to Screen and choose the OK button.

The report will be similar to one shown in Figure 4-3.

Table 4-7 Extracted sequence logic flat files

File	Extracted file information
.SQL	SAMPLE, WORK, 2= /ALARM /PERSONAL TRNSPRT . ^EOS SAMPLE, WORK, 3= /ALARM PERSONAL . ^EOS SAMPLE, WORK, 4= ALARM /TRNS-2 . ^EOS SAMPLE, WORK, 5= ALARM TRNS-2 .

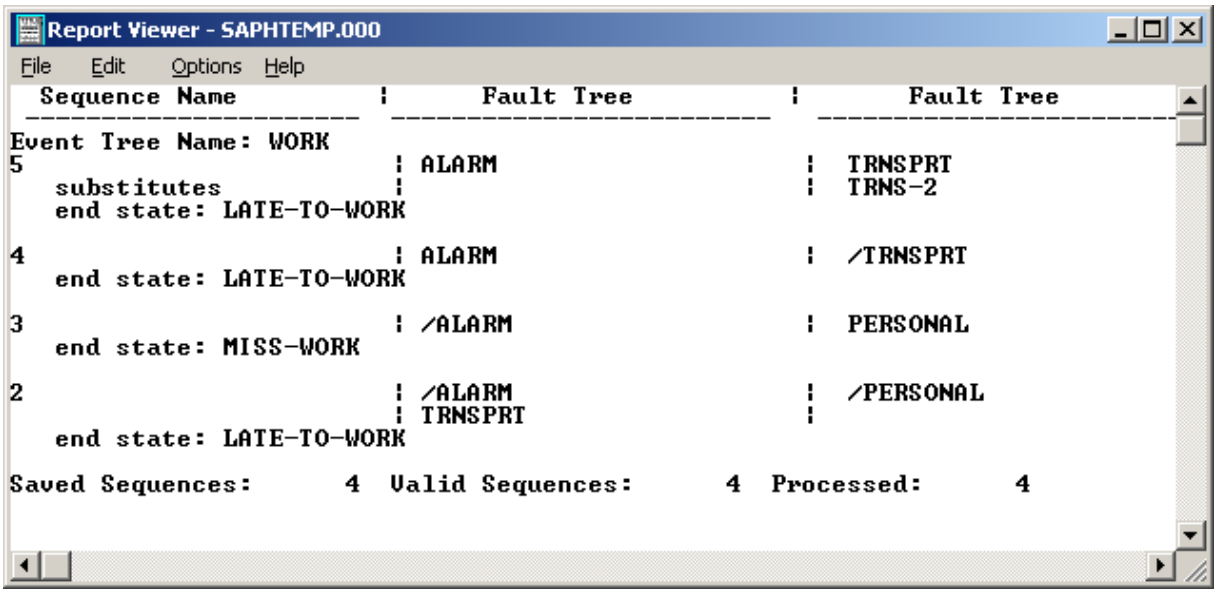


Figure 4-3 Sequence generation logic report

4.4 Entering End State Data

This section describes entering the end state data so that end state data are included in both the graphics and analysis portion of SAPHIRE. The following steps must be performed to actually load and verify the end state data:

1. Entering end state names in graphics (Section 4.4.1)
2. Entering end states for analysis (Section 4.4.2)
3. Entering the end state description and text (Section 4.4.3).

4.4.1 Entering End State Names in Graphics

End state data are used in a PRA analysis to group sequences that have similar outcomes for subsequent entry into the level 2 analysis. The sample database has four sequences that are grouped into two end states (late-to-work and miss-work). A subsequent analysis is possible on these two end states. Two flat files that can be obtained that contain end state data are shown in Table 4-8.

Table 4-8 Extracted end state flat files

File	Extracted file information
.ESD	SAMPLE =
	LATE-TO-WORK , This end state represents being late to work
	MISS-WORK , This end state represents missing work
.EST	SAMPLE, LATE-TO-WORK=
	THIS IS THE LATE TO WORK END STATE

Below are the available methods for entering end state names in graphics:

Event Tree Graphical Editor Method

End state names may be entered in the graphics editor. Using the graphics editor is potentially the most time-consuming but the most straightforward method. In this case, the event tree could be finalized and files extracted without any intermediate step. The event tree logic flat files shown in Table 4-4 contain end state names. In depth procedures for adding end state and sequence names using the graphics editor is provided in Appendix A. The SAPHIRE Tutorial contains details concerning this process.

The procedure for entering the end state name in the graphics editor requires the following:

1. Opening the event tree graphical editor.

2. Selecting a sequence/end state and right clicking it to bring up the Edit Sequence dialog.

3. Typing in the end state name.

4. If necessary, right clicking in the sequence header area and checking the end state display check box.

Sequence/End State Editor Method

End state names also can be entered using the sequence editor. This is an easy method, provided the event tree has already been constructed. Though it may be slower than the third method discussed here, it is recommended for most situations. One potential problem is that the headers cannot be toggled on and off in the sequence editor, and, even though end state or sequence names have been added, they may not automatically appear in the graphics display. If necessary, use the step-by-step guide for entering end state names provided in the tutorial.

The procedure for entering the end state name using the sequence editor requires the following:

1. Select the desired event tree and choose Edit End State from the popup menu option in the Event Tree List dialog.

2. Select the Edit End States option.

3. Select the desired sequence and press the Line Edit button.

4. Enter the end state name data.

End state name data can be entered into the event tree flat file (.ETG or .ETL extracted from the SAPHIRE program) using a text editor. After modification, the file must be loaded as described in Appendix A. This may be the fastest method available but requires steps that are more substantial and, therefore, has potential for error. This method is not presented.

4.4.2 Entering End States for Analysis

Like sequences, even though the end state names may appear in the graphics, they will not be available for analysis until the sequences in the event tree are generated. Unlike sequences, the assigned end state names will be preserved.

4.4.3 Entering End State Description and Text

Descriptions and text associated with event tree end states can also be entered, though it is unnecessary for analysis. Below are the available methods for entering end state description and text.

Interactive Modify End States Method

End state data can be entered from the Modify → End States main menu option. This is perhaps the easiest method as there are usually not a large number of end states and it is done entirely within the SAPHIRE environment. Though it may be slower than the other method discussed here, we recommend it for most situations.

The procedure for entering the end state descriptions and text is to (a) select the Modify → End State main menu option, (b) highlight the desired end state, and (c) choose the Modify popup menu option to enter the description, or press the Text button to enter the end state text.

Load from End State Flat Files Method

This data can be entered into the end state flat file (.ESD and/or .EST extracted from the SAPHIRE program), using a text editor. After modification, the files must be loaded as described in Appendix A. This method is not discussed further.

4.5 Loading the Fault Tree Data

This section describes loading the database fault trees and associated data and verifying their accuracy. Again, it may be more appropriate to enter data in a different order, depending on the type of data. For nuclear power plant PRAs, the order of data loading presented in this manual has been found to be the most efficient. Fault trees are used in PRAs to represent system failure logic. The sample database has four fault trees, each representing a different top event in the event trees as shown in the figures from Section 3.

The SAPHIRE software contains an option for using the "alpha to graphics" feature to convert the alphanumeric logic structure to a fault tree graphics file (.DLS). The alpha-to-graphics conversion will automatically build the graphical fault tree using the fault tree logic (.FTL). It will recognize and place into the fault tree graphic (1) the fault tree description (as found in the .FTD file), (2) the descriptions of any basic events used in the logic (as found in the .BED file), and (3) all gate descriptions used in the logic (as found in the .GTD file). If, at the time of conversion, this information is not loaded into the interactive database, SAPHIRE will use default names or blanks. The alpha-to-graphics conversion procedure is provided in Appendix A. The alpha-to-graphics conversion is a very powerful tool but will require some familiarity before it is possible to take full advantage of its usefulness.

Note:

- A .DLS file will be generated during the alpha-to-graphics conversion process and will be located in the project directory.
- Changes to gates and basic events can be made in the MODIFY BASIC EVENTS menu. An alpha-to-graphics conversion performed on the fault tree after the change will implement the change in the graphics.
- Fault tree, basic event, and gate descriptions will not appear in the graphics text boxes (the default is blank) if the appropriate data have not been loaded into the database.

There are four methods to develop fault tree graphics that represent the logic, depending on whether the data is available electronically or in hardcopy.

1. If hard copy data is available that contains the fault tree structure in graphics form,

   a. Create the fault tree graphics files (.DLS) in either the SAPHIRE fault tree graphical or logic editor, adding the basic event and gate names.

   b. Add the basic event, fault tree, and gate descriptions through either editor, or

      extract the necessary flat files to enter the basic event descriptions (.BED - Section 0), fault tree descriptions (.FTD -Section 0), and gate descriptions (.GTD - Section 0). Load these modified files and use the alpha-to-graphics conversion option (Appendix A) to enter the data into the graphics.

2. If hard copy data contain the fault tree structures defined as logic,

a. Use a text editor to enter the logic in the fault tree logic file (.FTL) format.

b. Use a text editor to develop files that contain the basic event descriptions (.BED - Section 0), fault tree descriptions (.FTD - Section 0), and gate descriptions (.GTD - Section 0) in the correct formats.

c. Load these files into SAPHIRE (see Appendix A for the procedure.)

d. Use the alpha-to-graphics conversion to develop the graphical representation of the fault trees (see Appendix A for procedure).

3. If electronic data contain fault tree logic structures that are compatible with SAPHIRE, directly load the file into SAPHIRE.

4. If electronic data contain a fault tree defined as logic that is not compatible with SAPHIRE,

   a. It may be possible to convert these files into a form that can be entered directly into SAPHIRE using programming (e.g., BASIC, Fortran, C) or an editing tool with a macro language (e.g., Excel or Multi-Edit). This requires either editing and/or programming skills that are beyond the scope of this manual. If it is not possible to develop a program to convert the files, it may be possible to use available hard copy graphics or print out logic and use the methods discussed above to enter the data.

The following steps must be performed to actually load and verify all the fault tree data.

1. Entering the fault tree logic (section 4.5.1)

2. Entering the fault tree descriptions and text (section 4.5.2)

3. Entering the gate descriptions and attributes (section 4.5.3)

4. Generating fault tree cut sets (section 4.5.4).

4.5.1 Entering Fault Tree Logic

The fault tree data entry is complicated by the fact that SAPHIRE uses an interactive database. Information entered in the process of graphical fault tree construction is used in many areas of the program. Graphical data structure translated into logic and other information are entered into the interactive database using internal lists. Such information includes the type of gates and basic events used, the textual descriptions entered in gate and basic event boxes, and the textual descriptions added for a fault tree description. The information on these internal lists can subsequently be extracted into SAPHIRE flat files. Conversely, SAPHIRE can be used to build fault tree graphics from logic and descriptions entered in the database.

Note: When a new fault tree is saved, a .DLS file is automatically created in the project subdirectory. The graphics file is translated into internal fault tree logic. Because of entering the fault tree graphics, the .FTL, .FTD, .GTA, and .GTD (fault tree logic, fault tree description, fault tree gate attributes, and fault tree gate descriptions, respectively) files can be extracted from the interactive database. SAPHIRE will provide default gate and basic event names. Therefore, we recommend that both gate names and the basic event names be entered at the time the fault tree is built.

There are different methods to enter fault tree logic, depending on what data type is available. The quickest way is to enter existing files (either graphic or logic) if available and compatible. The next best method is to enter the fault-tree structure information into SAPHIRE in the graphical editor or logic editor. It is largely a matter of personal preference as to which editor to use.

It is also possible, but may be difficult, to develop logic to enter into a flat file from a graphic and then load this file into SAPHIRE. It is relatively straightforward to enter fault tree logic in the graphical or logic editor. The process of entering and saving fault trees is discussed in detail in the tutorial. The fault tree flat files that contain the graphics and logic information for the sample database are shown in Table 4-9. Below are the available methods for entering fault tree logic.

Fault Tree Graphical Editor Method

If only hard copy data are available in graphics form, then create the fault tree graphics files (.DLS) in the SAPHIRE graphical editor.

The fault tree creation procedure requires you to (a) select the Fault Tree main menu option, (b) add a fault tree by right clicking to choose the Add Fault Tree popup menu option, (c) invoke the fault tree graphical editor by right clicking to choose the Edit Graphics popup menu option, and (d) from there, entering the fault tree structure, as shown in the fault tree figures from Section 3. An example display from the graphical editor is shown in Figure 4-4.

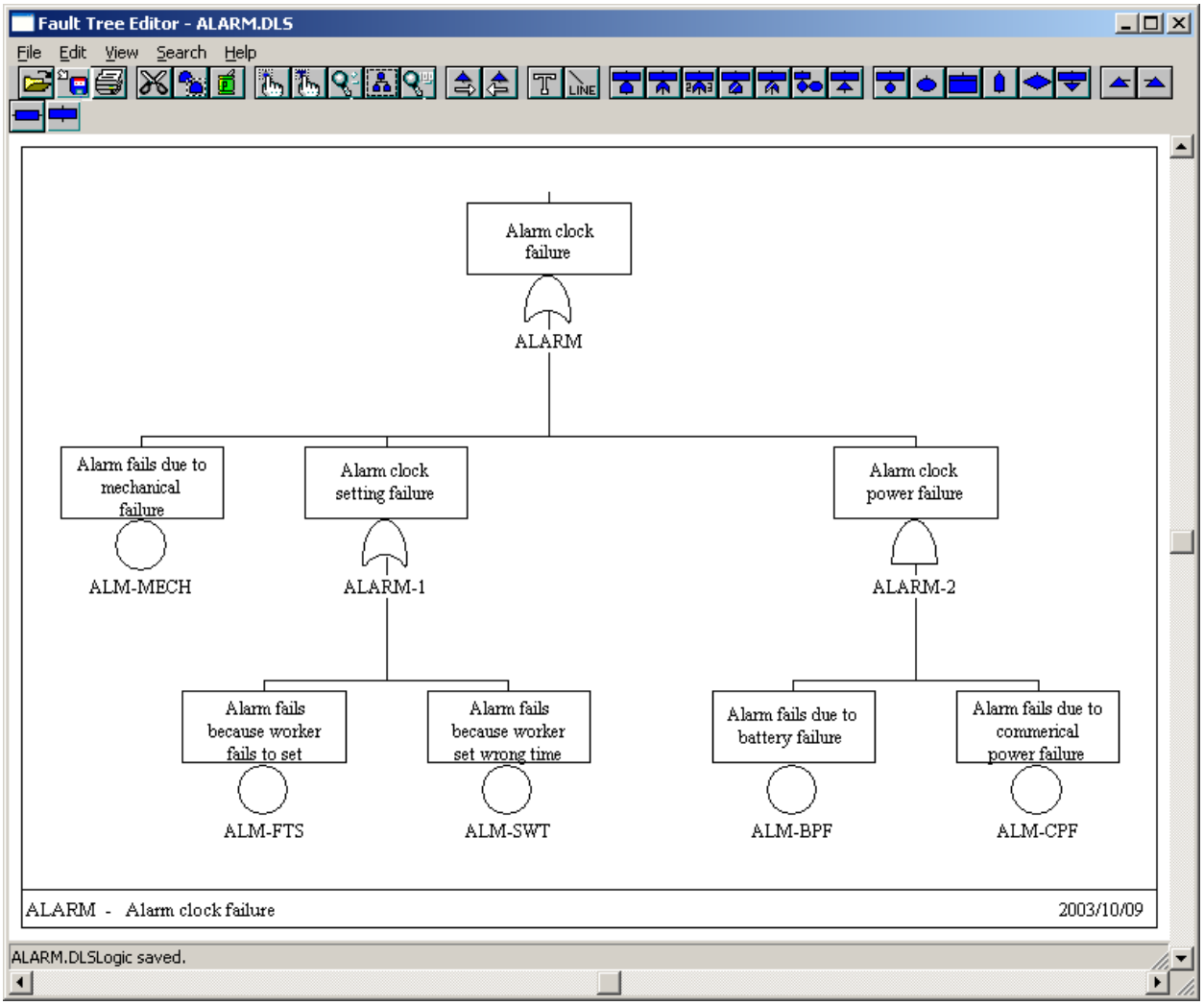


Figure 4- 4 Fault tree graphical editor.

Note:

- X IMPORTANT: The fault tree top gate name must be named the same as the fault tree file name.

- X The description of the top gate will be used for the fault tree, but only if the interactive database does not already have a description for the fault tree.

- X The .DLS file contains fault tree graphical information. To view and modify a fault tree, the .DLS flat file for that fault tree must be available on the subdirectory. Once the graphics file has been loaded into the interactive database, it is not necessary to have the graphics available (for cut set generation and quantification). The .DLS files can be cleared and extracted using the Utility → Fault Tree → Extract Graphics main menu option (see the SAPHIRE User's Guide). Also, .DLS files can be extracted from the database as described in Appendix A.

- X When building fault trees, ensure that there are no discontinuities in lines connecting gates, events, and transfers. Discontinuities in these lines will interrupt fault tree logic.

- X For ease of document control, consider including the project name, the fault tree name, the title, and the document-related page number in the graphics, or use the Page Info preference option to display some of this information.

- X While building large fault trees, save them periodically to prevent loss of data due to a power failure.

- X IMPORTANT: SAPHIRE uses gates names to optimize solving fault trees. A unique gate name must be used for each gate. Only when multiple gates share the identical inputs, may they also share the same name. When this happens, it is good practice to turn the gate and its inputs into a sub tree and reference it as a transfer, to minimize the possibility of differing inputs.

- X A transfer is usually made to the top gate of another fault tree. However, you can transfer to a gate on the same page but not to the middle of another fault tree.

- X All fault trees are entered into the interactive database system listing as top gate fault trees. It is up to the user to designate these as sub-trees in the Modify → Fault Trees main menu option. This does not affect the analysis except that the fault tree list displayed in the Fault Tree main menu option can then be toggled via the Show Sub Trees check box to display either only top level or all fault trees.

Fault Tree Logic Editor Method

Alternatively, if hard copy data are available that contain the fault tree structure in graphics form, you may create the fault tree logic (and subsequently, the graphics files) using the SAPHIRE fault tree logic editor.

As with the graphical creation procedure, you must (a) select the Fault Tree main menu option and (b) add a fault tree by right clicking to choose the Add Fault Tree popup menu option.

Then, invoke the fault-tree logic editor by right clicking to choose the Edit Logic popup menu option and (d) from there, enter the fault tree logic. An example display from the logic editor is shown in Figure 4- 5. The process of entering the logic into the editor is discussed in detail in the SAPHIRE User's Guide and the SAPHIRE Tutorial.

Once the logic has been defined, the logic editor gives you the option to convert the logic into graphical (.DLS) format.

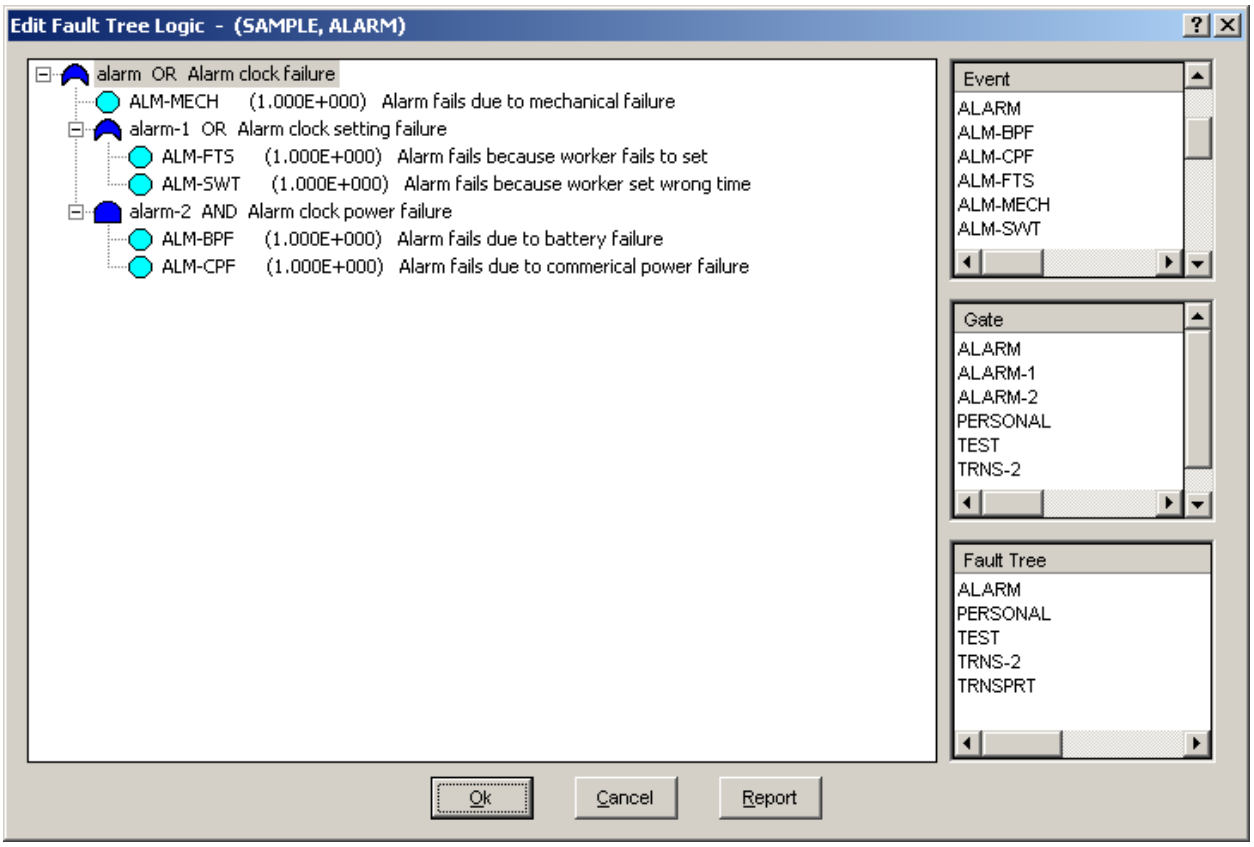


Figure 4- 5 Fault tree logic editor.

Load Fault Tree Logic from Flat (.FTL) File Method

If hard copy data contain the fault tree structures defined as logic, then you may use a text editor to enter the logic in the fault tree logic file (.FTL) format and load this file into SAPHIRE. An example of the .FTL file format is shown in 4-9. This method is not discussed further.

Table 4- 9 Extracted fault tree logic and graphic flat files.

File	Extracted file information
.FTL	SAMPLE, ALARM = ALARM OR ALARM-1 ALARM-2 ALM-MECH ALARM-1 OR ALM-SWT ALM-FTS ALARM-2 AND ALM-CPF ALM-BPF ^EOS SAMPLE, PERSONAL = PERSONAL OR SICK SICK-FAM OTHER ^EOS SAMPLE, TRNS-2 = TRNS-2 AND PER-TRNS PUB-TRNS-LAT ^EOS SAMPLE, TRNSPRT = TRNSPRT AND PER-TRNS PUB-TRNS
.DLS	Is not in ASCII format and therefore cannot be viewed or edited.

Load Fault Tree Logic from Graphics (.DLS) File Method

If electronic data contain the fault tree structures defined as logic that are compatible with SAPHIRE, directly load the file into SAPHIRE.

To load one or more DLS files into SAPHIRE, choose the Utility → Fault Trees → Load Graphics main menu option. From there you may select the desired DLS files to load.

Graphics files can also be loaded via the Utility → Load and Extract main menu option. To use this option, select the Load data action, the Fault Tree data type, Graphics file type, and press the Process button. From there you may select the desired DLS files to load.

4.5.2 Entering Fault Tree Descriptions and Text

As with event trees, many PRAs will contain descriptions and in depth textual discussion on those fault trees considered important to the analysis. The sample database has both description and text for all the fault trees developed for demonstration. Table 4-10 contains the fault tree descriptions and text extracted. Below are the available methods for entering the fault tree descriptions and text.

Table 4- 10 Extracted fault tree descriptions and text flat files.

File	Extracted file information
.FTD	SAMPLE = ALARM ,ALARM CLOCK FAILURE PERSONAL ,PERSONAL PROBLEMS TRNS-2 ,COMMERCIAL TRANSPORTATION FAILS AT A LATER TIME TRNSPRT ,PERSONAL AND COMMERCIAL TRANSPORTATION FAIL
.FTT	SAMPLE, ALARM= The ALARM fault tree (Figure 3-2) is a simple representation modeling alarm clock failure. Some common reasons for alarm clock failure include setting the wrong time, mechanical failure, or power failure (either battery or commercial).

Interactive Modify Fault Trees Method

Fault tree descriptions and text can be entered in the Modify → Fault Trees main menu option. Using this module is perhaps the easiest method as it is done entirely within the SAPHIRE environment. Though it may be slower than the other methods discussed here (depending on the number of fault trees), we recommend it for most situations. Use of the SAPHIRE text editor is described in the SAPHIRE User's Guide.

The procedure for entering the fault tree descriptions and text is to (a) select the Modify → Fault Tree main menu option, (b) highlight the desired fault tree, and (c) choose the Modify popup menu option to enter the description, or press the Text button to enter the fault tree text.

Load from Fault Tree Flat File Method

Fault tree descriptions and text can also be entered into the fault tree flat file (.FTD) extracted from the SAPHIRE program using a text editor. The fault tree textual data can be entered into the fault tree text flat file (.FTT) using the SAPHIRE format. (This is also true of the .FTD). After modification or development, both files must be loaded as described in Appendix A. This method is not discussed further.

4.5.3 Entering Gate Descriptions and Attributes

Gate descriptions are usually available in PRAs. They are useful and necessary for clarifying how the system logic was developed for use in future analysis. For example, gate descriptions may designate where certain train logic begins in the fault tree logic so that the branch can be eliminated for analysis. In the sample database, descriptions are available even though they do not provide any additional information concerning the analysis. Note that the SAPHIRE attribute is the type of gate, (i.e., OR, AND, and TRANSFER). The gate name and this information should already be present in the Modify → Gates main menu option from entering the fault tree logic into the interactive database. Table 4-11 shows the fault tree gate files extracted.

Below are the available methods for entering gate descriptions and attributes.

Table 4- 11 Extracted fault tree gate flat files.

File	Extracted file information
.GTD	SAMPLE =
	ALARM , ALARM CLOCK FAILURE
	ALARM-1 , ALARM CLOCK SETTING FAILURE
	ALARM-2 , ALARM CLOCK POWER FAILURE
	PERSONAL , PERSONAL PROBLEMS
	TRNS-2 , COMMERCIAL TRANSPORTATION FAILS AT A LATER TIME
	TRNSPRT , PERSONAL AND COMMERCIAL TRANSPORTATION FAILURE
.GTA	SAMPLE =
	* Name , Type
	ALARM , OR
	ALARM-1 , OR
	ALARM-2 , AND
	PERSONAL , OR
	TRNS-2 , AND
	TRNSPRT , AND

Fault Tree Graphical or Logic Editor Method

Gate descriptions and attributes are easily entered into the graphics editor. This method is potentially the most time consuming but the most straightforward. In this case, the fault tree could be finalized and files extracted without any intermediate steps. Both the SAPHIRE Reference Manual and the SAPHIRE Tutorial contain details concerning this process.

To enter the data using the graphical editor, select and right click on the desired gate and choose the Edit popup menu option. To enter the data using the logical editor, select and right click on the desired gate and choose the Modify popup menu option.

Interactive Modify Gates Method

Gate descriptions and attributes can be entered using the Modify → Gate main menu option and then performing an alpha-to-graphics conversion to place the description in the graphics. This is perhaps the easiest method as it is done entirely within the SAPHIRE environment.

A possible advantage of this method (and the following method) over the graphical or logical editor method, is that descriptions need only be entered one time, whereas if a gate is referenced in multiple places in the logic, you may end up typing the description in several times.

Though it may be slower than the final method discussed here, we recommend it for most situations. See the SAPHIRE User's Manual for additional information.

The procedure for entering the gate descriptions is to (a) select the Modify → Gate main menu option, (b) highlight the desired gate, and (c) choose the Modify popup menu option to enter the description.

When data entry is finalized, perform an alpha to graphic conversion (see Appendix A) to enter this information into the fault tree graphics.

Load Gate Data from Flat File Method

Gate descriptions and attributes can be entered using a text editor into the gate description flat file (.GTD) that was extracted from the SAPHIRE program. After modification, the file must be loaded as described in Appendix A. The attribute file data will have been entered in the process of entering the fault tree logic. It may be useful to extract the gate attribute flat file (.GTA) for some other purpose. This method is not presented.

4.5.4 Generating Fault Tree Cut Sets

It has been noted that some PRAs provide in depth fault tree cut set information while others do not. Having the original fault tree cut sets is very helpful in verifying that the correct logic has been entered into the database. Since most PRAs comprised large system fault trees, it is possible to generate many more cut sets than what may have been reported. In these cases, to duplicate the PRA fault tree cut sets, it may be necessary to vary the probability cutoff used to generate them. In addition, for some databases, it may be impossible to match the fault tree cut sets that are reported in the PRA with those generated in SAPHIRE. This can be due to many reasons, one of which is poor documentation for the original analysis performed. In this case, it may be necessary to enter manually the cut sets into the database. For the sample database, the fault tree cut sets were presented in Section 3. It is important to note that for cut set generation and quantification, SAPHIRE uses only the logic and not the graphical representation of the fault tree. The graphics are useful for easy visualization of the fault tree. Table 4-12 shows the system cut set flat files extracted. Below are the available methods for creating fault tree cut sets.

Solve Fault Tree Logic Method

In the process of generating fault tree cut sets, fault tree logic can be verified. The SAPHIRE User's Guide and the SAPHIRE Tutorial provide additional information on this process.

The procedure for generating fault tree cut sets and obtaining a report for verification requires the following:

a. Entering the Fault Tree Analysis module from the main menu.

b. Selecting the Analyze Systems option.

c. Highlighting the fault tree to analyze and select the Generate Cut Sets option. When Generate Cut Sets is chosen, an intermediate screen will appear that queries Cut Set Generation Cutoff Values. This is where the probability cutoff can be set to limit the cut sets produced, or can be varied to duplicate the original PRA. See the SAPHIRE User's Guide and the SAPHIRE Technical Reference Manual for a discussion of these features.

d. Pressing enter after selecting the appropriate cutoff values.

e. Entering Display Results from the Fault Tree Analysis menu.

f. Highlighting a fault tree and selecting Cut sets to view cut sets and to produce a report.

Table 4- 12 Extracted fault tree cut sets flat files.

File	Extracted file information
.FTC	SAMPLE, ALARM, 0001= ALM-FTS + ALM-MECH + ALM-SWT + ALM-BPF * ALM-CPF . ^EOS SAMPLE, PERSONAL, 0001= OTHER + SICK + SICK-FAM . ^EOS SAMPLE, TRNS-2, 0001= PER-TRNS * PUB-TRNS-LAT . ^EOS SAMPLE, TRNSPRT, 0001= PER-TRNS * PUB-TRNS .

Load from Fault Tree Cut Set Flat File Method

Cut set data can be entered by first using a text editor to edit the fault tree cut set flat file (.FTC) developed using the SAPHIRE format. After development, the file must be loaded as described in Appendix A. This would only be used in a case where it is impossible to match the database files with the generated cut sets. (This may occur even when the fault tree graphics appear identical.) This is a slower method, and because it requires more steps in the data entry process may be prone to errors. This method is not presented.

4.5.5 Verifying the Fault Tree Data

After the logic and data for each fault tree are entered, it is a necessary step to verify that the information entered into the database is correct before proceeding. The recommended method to check the fault tree data is to extract those flat files, reports, and graphics that are the most similar to what is presented in the database.

4.6 Loading Basic Event Data

This section discusses loading the basic event information such as probabilities, calculation types, and attributes. As event tree files (see Section 4.2) and fault tree files (see Section 4.5) are created or loaded, SAPHIRE constructs an internal list of all basic events, undeveloped events, gates, initiating events, and top events. These are added to the interactive database Basic Event list found in the Modify → Basic Events main menu option. This list will not be complete. You will still need to enter probability values, descriptions, and other detailed information, as necessary. Additionally, new basic events may need to be added to account for beta factors, recovery actions, and other factors. For more information on SAPHIRE operation as it relates to basic event information, consult the reference and technical manuals.

SAPHIRE offers two main methods to add basic events and their associated information into the project: using the Modify→Basic Events main menu option, and using a flat file to load text based files through the Utility → Load (and Extract) main menu option.

To achieve the optimum combination of speed and accuracy, a combination of these methods may be utilized. It is generally recommended that basic events be added using the interactive option, and modified (when large numbers of events must be edited) by using the flat file method.

SAPHIRE basic events can contain a wide range of detail, including failure rate and uncertainty data, general attributes, process flags, and compound and transformation data. It is beyond the scope of this manual to address the details of the basic event data feature content. Appendix B enumerates the available field options, which are discussed in more detail in the SAPHIRE Users Guide.

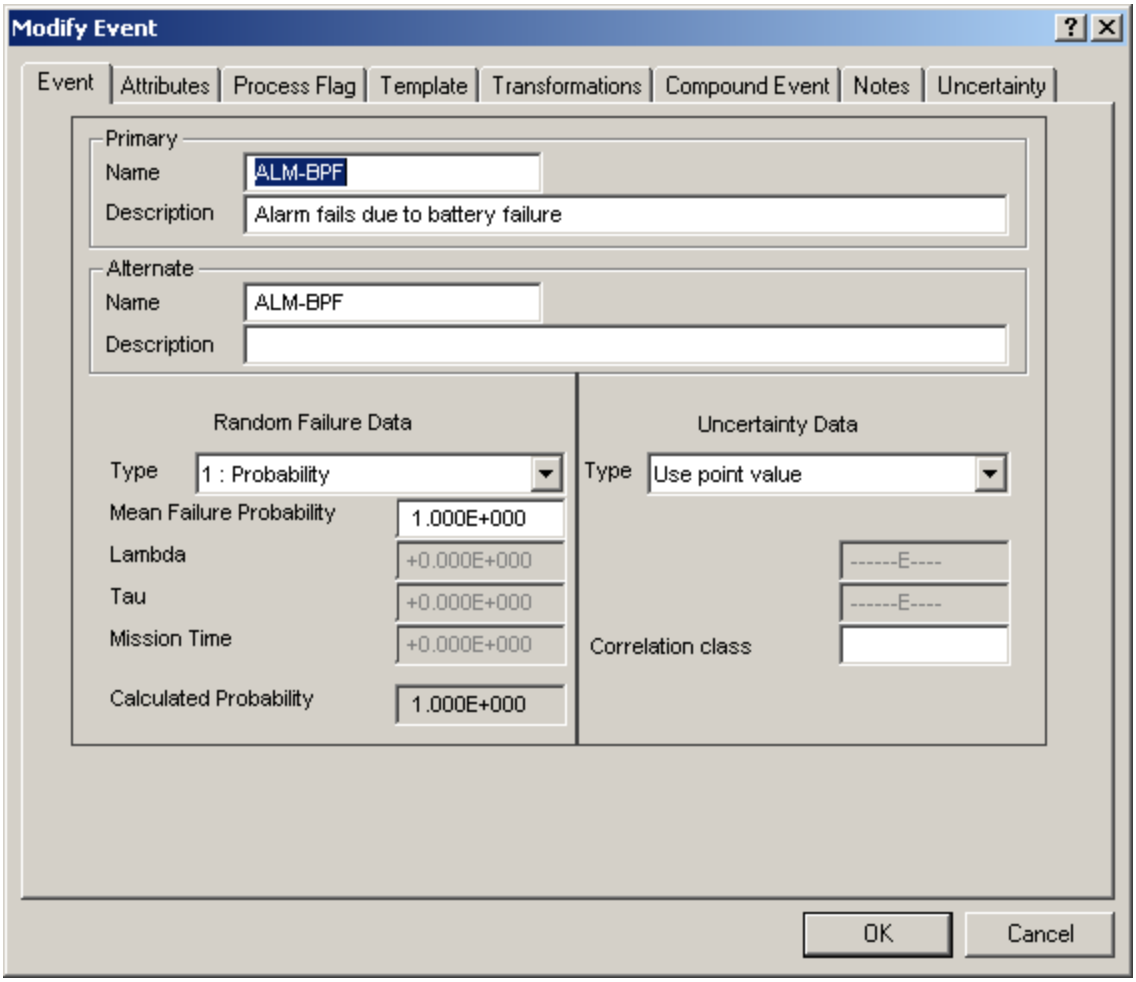


Figure 4- 6 The Modify Event dialog.

The following steps must be performed to actually load and verify all the basic event data:

1. Adding basic events (Section 4.6.1)

2. Entering the basic event descriptions (Section 4.6.2)

3. Entering the basic event data (availability and uncertainty) (Section 4.6.3).

4.6.1 Adding Basic Events

Basic events not listed in either the fault tree or event tree may be necessary in a PRA to accommodate special situation such as substitutions or recovery actions. The sample database requires the entry of one recovery action basic event. This is shown in the basic event listing in Section 3. (Note that SAPHIRE will also allow you to enter a new basic event in the Recovery Rule Editor.)

Below are the available methods for adding basic events.

Interactive Modify Basic Events Method

Basic events can be entered through the Modify → Basic Events main menu option. Using this method is perhaps the easiest because it is done entirely within the SAPHIRE environment. Though it may be slower than the other method discussed here, it is recommended for most situations. See the SAPHIRE User's Guide and the SAPHIRE Tutorial for more information.

The procedure for entering the basic event requires you to (a) select the Modify → Basic Events main menu option, and (b) select Add or Modify from the right click popup menu option.

Load from Basic Event Flat File Method

Basic events also can be entered using a text editor by modifying the basic event flat file (.BED) that can be extracted from the SAPHIRE program. After modification, the file must be loaded as described in Appendix A. This may be the fastest method available but requires more substantial steps and may be prone to errors. This method is not discussed further.

4.6.2 Adding Basic Event Descriptions

Basic event descriptions are commonly used in PRAs. When entered into the interactive database, the alpha-to-graphics conversion can be used to place the descriptions into the fault tree graphics. Table 4-14 shows the basic event description flat file extracted.

Below are the available methods for adding basic event descriptions.

Interactive Modify Basic Event Method

Basic events can be edited through the Modify → Basic Events main menu option. This method is perhaps the easiest because it is done entirely within the SAPHIRE environment, but it is not generally recommended for most databases since the number of basic events is large. See the SAPHIRE User's Guide for more information.

The procedure for entering the basic event requires you to (a) select the Modify → Basic Events main menu option, (b) highlight the desired event, and (c) select Modify from the right click popup menu option.

Load from Basic Event Description Flat File Method.

Basic event descriptions can be entered using a text editor by modifying the basic event flat file (.BED) that can be extracted from the SAPHIRE program. After modification, the file must be loaded as described in Appendix A. This is fastest method available and, due to the large number of basic events common in most PRAs, we recommend it over method A. This method is not discussed further.

Table 4- 13 Extracted basic event descriptions flat file.

File	Extracted file information	
.BED	SAMPLE	=
	<FALSE>	,System Generated Success Event
	<PASS>	,System Generated Ignore Event
	<TRUE>	,System Generated Failure Event
	ALARM	,Alarm system fault tree
	ALM-BPF	,Alarm fails due to battery failure
	ALM-CPF	,Alarm fails due to commercial power failure
	ALM-FTS	,Alarm fails because worker fails to set
	ALM-MECH	,Alarm fails due to mechanical failure
	ALM-SWT	,Alarm fails because worker set wrong time
	MEDICINE	,Recovery for sick failure preventing attending work
	OTHER	,Other personal reasons that cause a failure to get to work
	PER-TRNS	,Personal transportation
	PERSONAL	,Personal reasons for failure system fault tree
	PUB-TRNS	,Public transportation fails
	PUB-TRNS-LAT	,Public transportation fails late time frame
	SICK	,Failed to get to work because of illness
	SICK-FAM	,Failed to get to work because of illness in project
	TRNS-2	,Transportation system fault tree-late time frame
	TRNSPRT	,Transportation system fault tree
	WORK	,Event tree (WORK) initiating event

4.6.3 Entering Basic Event Data

To determine the frequency of failure in a SAPHIRE analysis, it is necessary to enter the probability or frequency of failure for each basic event. Most PRAs may have several calculation types, the most common being failure on demand, failure over a mission time, and standby failure rates. In addition, PRAs generally address uncertainty and will provide applicable uncertainty parameter information. It is beyond the scope of this document to present all the possible applications available. The SAPHIRE Technical Reference Manual provides a detailed discussion on many of the features available. The sample database contains limited examples and is presented for illustration only. Table 4-15 shows the basic event data flat files extracted.

Below are the available methods for entering basic event data.

Interactive Modify Basic Event Data Method

Basic events can be edited through the Modify → Basic Events main menu option. Using this method is perhaps the most straightforward method, as it is done entirely within the SAPHIRE environment. However, in this case we do not recommend method because of the many keystrokes that may be necessary. See the SAPHIRE User's Guide for more information.

Table 4- 14 Extracted basic event data flat files.

File	Extracted file information
.BEA	**SAMPLE =** *** Name ,AltName ,Typ,Sys,Fail,Loc,CompId, GroupName ,Train,Attributes** **<FALSE> ,<FALSE> , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **<PASS> ,<PASS> , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **<TRUE> ,<TRUE> , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALARM ,ALARM ,DE , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALM-BPF ,ALM-BPF , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALM-CPF ,ALM-CPF , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALM-FTS ,ALM-FTS , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALM-MECH ,ALM-MECH , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **ALM-SWT ,ALM-SWT , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **MEDICINE ,MEDICING , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **OTHER ,OTHER , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **PER-TRNS ,PER-TRNS , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **PERSONAL ,PERSONAL ,DE , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **PUB-TRNS ,PUB-TRNS , , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N** **PUB-TRNS-LAT ,PUB-TRNS-LAT, , , , , , , ,N,N,N,N,N,N,N,N,N,N,N,N,N,N**

```
SICK          ,SICK          ,   ,   ,  ,   ,        ,             ,
,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N

SICK-FAM      ,SICK-FAM      ,   ,   ,  ,   ,        ,             ,
,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N

TRNS-2        ,TRNS-2        ,DE ,   ,  ,   ,        ,             ,
,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N

TRNSPRT       ,TRNSPRT       ,DE ,   ,  ,   ,        ,             ,
,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N

WORK          ,WORK          ,   ,   ,  ,   ,        ,             ,
,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N
```

.BEI

```
SAMPLE            =

*  Name          ,FdT,UdC ,UdT,UdValue, Prob      , Lambda    , Tau       , Mission
,Init

<FALSE>          ,1,    ,L, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

<PASS>           ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

<TRUE>           ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALARM            ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALM-BPF          ,1,    ,L, 3.000E+000, 9.000E-008,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALM-CPF          ,1,    ,L, 3.000E+000, 1.500E-002,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALM-FTS          ,1,    ,L, 1.000E+001, 5.500E-006,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALM-MECH         ,1,    ,L, 3.000E+000, 2.700E-008,+0.000E+000,+0.000E+000,+0.000E+000, ,

ALM-SWT          ,1,    ,L, 1.000E+001, 2.700E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

MEDICINE         ,1,    ,L, 5.000E+000, 5.000E-001,+0.000E+000,+0.000E+000,+0.000E+000,R,

OTHER            ,1,    ,L, 1.000E+001, 8.100E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

PER-TRNS         ,1,    ,L, 5.000E+000, 5.500E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

PERSONAL         ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

PUB-TRNS         ,1,    ,L, 3.000E+000, 2.700E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

PUB-TRNS-LAT     ,1,    ,L, 3.000E+000, 2.000E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

SICK             ,1,    ,L, 1.000E+001, 8.100E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

SICK-FAM         ,1,    ,L, 1.000E+001, 4.000E-003,+0.000E+000,+0.000E+000,+0.000E+000, ,

TRNS-2           ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

TRNSPRT          ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,+0.000E+000,+0.000E+000, ,

WORK             ,1,    ,L, 2.000E+000, 2.480E+002,+0.000E+000,+0.000E+000,+0.000E+000,I,
```

The procedure for entering the basic event requires you to (a) select the Modify → Basic Events main menu option, (b) select Modify from the right click popup menu option, and (c) enter the basic event values and press the OK button to save.

Load from Basic Event Information Flat File Method.

Basic event data also can be entered using a text editor by modifying the basic event information flat file (.BEI) that can be extracted from the SAPHIRE program. After modification, the file must be loaded as described in Appendix A. Using this technique is the recommended method (though the file will need to be reloaded after modification) since it requires substantially fewer keystrokes and is the fastest method available. This method is not discussed further.

Note:

- Not all the information for a basic event needs to be entered for calculation purposes. The information required is the primary name, the initiating event indication, the calculation type, the probability value, and the uncertainty distribution type and value (uncertainty is only necessary if an uncertainty calculation is to be performed).
- When a basic event is added to the SAPHIRE internal list, it is assigned default values for uncertainty and failure data.
- *Important:* remember to generate event data as described in Appendix A to obtain an updated current case database. Any basic event values input into the Modify → Basic Events module may appear only in the base case data (which is not used for analysis) until this procedure has been performed. See the discussion on the base and current case in Section 2.

4.7 Loading Sequence Data

This section discusses the loading of sequence data, including cut sets, text, and descriptions. Sequences are used in PRAs to develop the overall CDF value and to identify those scenarios of events that are of concern to plant safety. Sequences with similar outcomes are grouped by end states for evaluation in the level 2 and 3 analysis. Most PRAs present the dominant (or greatest contributors to CDF) sequence cut sets.

The following steps must be performed to actually load and verify all the sequence data

1. Generating sequence cut sets (Section 4.7.1)
2. Entering the sequence description and text (Section 4.7.2).

4.7.1 Generating Sequence Cut Sets

Since some PRAs have event trees that link to large system fault trees, it is possible to generate a large number of cut sets. The probability cutoff option and the size cutoff limits the number of cut sets to those above a certain value and order. This cutoff can be manipulated so that the cut sets match those produced by the PRA. For certain databases, it may be impossible to match the sequence cut sets that are reported in the PRA with those generated by SAPHIRE. This difference can be due to many reasons, one of which is poor documentation for the original analysis performed. In this case, it may be necessary to manually enter the cut sets into the database.

The sequence cut sets for the sample database are reported in Section 3. There was no cutoff used for this very simple problem. It is important to note that for cut set generation and quantification, SAPHIRE uses only the logic and not the graphical representation of the fault tree. Table 4-16 shows the sample database sequence cut sets.

Below are the available methods for generating sequence cut sets.

Solve Sequence Logic Method

To have SAPHIRE generate sequence cut sets, use the Analyze Sequences module. The SAPHIRE User's Guide and the SAPHIRE Tutorial provide additional information on this process.

The procedure for solving sequence logic for cut sets requires you to

1. Select the Sequence main menu option.

2. Select the sequence(s) to solve.

3. Choose the Solve option from the popup menu to bring up the Cut Set Generation Cutoff Values dialog. This is where the probability cutoff can be changed to limit the cut sets produced or can be varied to duplicate the original PRA. See the SAPHIRE User's Guide and the SAPHIRE Technical Reference Manual for a discussion of these features.

4. Select the Display→Cut Sets popup menu option to view the cut sets and, if desired, create a report.

Table 4- 15 Extracted sequence cut sets flat files.

File	Extracted file information
.SQC	**SAMPLE, WORK, 2, 0001=** **PER-TRNS * PUB-TRNS .** **^EOS** **SAMPLE, WORK, 3, 0001=** **OTHER +** **SICK * MEDICINE +** **SICK-FAM .** **^EOS** **SAMPLE, WORK, 4, 0001=** **ALM-FTS +** **ALM-MECH +** **ALM-SWT +** **ALM-BPF * ALM-CPF .** **^EOS** **SAMPLE, WORK, 5, 0001=** **ALM-FTS * PER-TRNS * PUB-TRNS-LAT +** **ALM-MECH * PER-TRNS * PUB-TRNS-LAT +** **ALM-SWT * PER-TRNS * PUB-TRNS-LAT +** **ALM-BPF * ALM-CPF * PER-TRNS * PUB-TRNS-LAT .**

Load from Sequence Cut Set Flat File Method

Using a text editor, cut set data can be entered into a sequence cut set flat file (.SQC) format. After development, the file must be loaded as described in Appendix A. This would only be used in a case where it is impossible to match the database files with the generated cut sets. (This may occur even when the logic appears identical.) This method will not be presented.

4.7.2 Entering the Sequence Description and Text

It is common that PRAs will discuss in detail the dominant sequences that were identified. The accident scenarios and recovery actions applied may be described in detail. The sample database contains a brief description and some text information for the sequences. Table 4-17 shows the sample database sequence description and text flat files.

Below are the available methods for entering the sequence description and text.

Interactive Modify Event Tree Sequence Method

The sequence description and text can be entered in the Modify → Event Tree. This technique is perhaps the easiest method as it is done entirely within the SAPHIRE environment. Although it may be slower than the other method discussed below, it is recommended for most situations. Additional information concerning adding descriptions and text is contained in the SAPHIRE User's Guide.

The procedure for entering the sequence description and text requires

1. Choose the Modify → Event Tree main menu option.

2. Highlight the event tree containing the desired sequence(s), and press the Sequences button. (Note that sequences will appear her only if they have been previously generated using the Link Trees method described earlier).

3. Select the sequence and choose Modify from the popup menu to add a description, or press the Text button to add text.

Table 4- 16 Extracted sequence description and text flat files.

File	Extracted file information
.SQD	**SAMPLE, WORK=**
	2 ,LATE-TO WORK
	3 ,MISS-WORK
	4 ,LATE-TO-WORK
	5 ,LATE-TO-WORK
.SQT	**SAMPLE, WORK, 3=**
	Sequence 3 sample text.

Load from Sequence Flat File Method

Using a text editor, the sequence description can be entered into the sequence description flat file (.SQD) format. The sequence textual data can be entered into the sequence text flat file (.SQT) using the SAPHIRE format. After modification or development, both files must be loaded as described in Appendix A. This method is not discussed further.

4.8 Recovery Actions

This section discusses the addition of recovery actions to sequence cut sets. PRAs often have recovery actions applied to a specific scenario of events that may occur in a sequence or fault tree cut set. These recovery actions are not directly modeled in either an event tree or fault tree and may be required to be added to the cut sets to obtain a result comparable to the PRA. The sample database has a very simple recovery action that will be applied to one sequence cut set. Recovery actions or recovery rules can be applied to fault tree cut sets using Fault Tree and Fault Tree-Project Rules. Recovery actions can also be applied to event tree sequence cut sets by using Project, Event Tree, and Sequence Rules. Methods used for both are similar. An example of a Project Rule recovery action being applied to sequence cut sets would be the case of double maintenance events not allowed by technical specifications. The sample database contains a simple example of a recovery action applied to a sequence cut set.

The following method discusses how to use SAPHIRE to apply recovery actions from the Sequences main menu option. The method will apply recovery actions to sequence cut sets, but fault tree cut set recovery actions are similar. The SAPHIRE User's Guide and the SAPHIRE Tutorial provide additional information on this process.

The procedure for applying recovery requires the following steps:

1. Select Sequences from the SAPHIRE main menu. The Sequences list will appear.

2. Highlight an event tree sequence, right click to invoke a pop up menu and choose the Cut Sets → Recover →Edit Event Tree option. (Depending on the desired applicable scope of the rule, Edit (sequence) Rule or Edit Project could also be selected.)

3. Type the recovery rule text into the rule editor and save it.

4. The recovery action MEDICINE can be viewed in sequence 3 in the Display Results option under the Event Tree Analysis menu.

Detailed steps for adding recovery actions are described in Appendix A.

4.9 Analyzing Uncertainty

Uncertainty of the cut set and end state results is commonly reported in the PRAs. Both Monte Carlo and Latin Hypercube options are available in SAPHIRE. It is sometimes difficult to compare SAPHIRE results with those reported in a PRA, because there will be an expected variability between the uncertainty runs depending on the algorithms used, the number of samples, and the seed numbers chosen.

The following steps must be performed to generate an uncertainty analysis for the database and verify it against the PRA:

1. Generate uncertainty for fault tree cut sets (Section 4.9.1)

2. Generate uncertainty for sequence cut sets (Section 4.9.2)

3. Generate uncertainty for end states (Section 4.9.3)

4. Generate uncertainty for groups of sequences or the project (Section 4.9.4).

4.9.1 Generating Uncertainty for Fault Tree Cut Sets

It is usual to find that a fault tree uncertainty analysis was reported for those PRAs that provided fault tree cut sets. The sample database provides the results to an uncertainty analysis. Uncertainty summary information is shown in Table 4- 17. Uncertainty can only be produced after cut sets have been generated. Further discussions on uncertainty analysis are found in both the reference and technical manuals.

The procedure for calculating fault tree uncertainty requires the following

1. Choose the Fault Trees main menu option.

2. Highlight the fault tree(s) and choose the Uncertainty option from the popup menu.

3. Select the uncertainty types and values to use in the Uncertainty Calculation Values dialog, then click OK.

4. Wait for the calculation to complete and press OK.

5. To view the uncertainty stored in the database, select a fault tree and choose Display → Uncertainty from the popup menu. (Either the current case or both the current and base case uncertainty values will be displayed, depending on whether a base case update has been performed. See Section 2 for a discussion of the base case update feature.)

6. To view detailed quantile information, choose either the Current or Base Quantile Values button.

Table 4- 17 Extracted fault tree attributes (uncertainty) flat file.

File	Extracted file information
.FTA	SAMPLE, 0001 = * Name , Level, Mission , MinCut , Def ProCut,Used ProCut,Sample,Seed,Siz,Sys, Cuts,Events, UdValues, Def Flags, Used Flags,S QMethod, S QPasses, R QMethod, R QPasses, Alt Name ALARM ,0, 2.400E+001, 2.706E-003,------E----,------E----, 5000, 4321,--, , 4, 003,5, 1.032E-004, 1.018E-003, 2.577E-003, 9.309E-003, 4.912E-006, 1.228E-001, 5.489E- 7.906E+000, 1.032E+002, , , ,----, , 0,ALARM PERSONAL ,0, 2.400E+001, 2.007E-002,------E----,------E----, 5000, 4321,--, , 3, 002,3, 2.759E-003, 1.198E-002, 1.950E-002, 5.977E-002, 4.544E-004, 6.530E-001, 2.731E- 7.855E+000, 1.219E+002, , , M,----, , 0,PERSONAL TRNS-2 ,0, 2.400E+001, 1.100E-005,------E----,------E----, 5000, 4321,--, , 1, 005,2, 7.801E-007, 5.441E-006, 1.039E-005, 3.676E-005, 8.671E-008, 3.369E-004, 1.549E- 5.348E+000, 6.211E+001, , , M,----, , 0,TRNS-2 TRNSPRT ,0, 2.400E+001, 1.485E-005,------E----,------E----, 5000, 4321,--, , 1, 005,2, 1.053E-006, 7.345E-006, 1.403E-005, 4.963E-005, 1.171E-007, 4.548E-004, 2.092E- 5.348E+000, 6.211E+001, , , M,----, , 0,TRNSPRT

4.9.2 Generating Uncertainty for Sequence Cut Sets

Most PRAs provide sequence cut set uncertainty. Again, it may be difficult to compare SAPHIRE results with those reported in a PRA because there will be an expected variability between the uncertainty runs, depending on the algorithms used, the number of samples, and the seed numbers chosen. The sample database provides the seed number and was developed on SAPHIRE using the Monte Carlo algorithm and, therefore, it should be possible to produce the same results.

Uncertainty can only be produced after cut sets have been generated. Further discussions on uncertainty analysis are found in both the SAPHIRE User's Guide and the SAPHIRE Technical Reference Manual.

The procedure for generating sequence uncertainty requires the following:

1. Choose the Sequences main menu option.

2. Highlight a sequence and choose the Uncertainty option from the popup menu.

3. Select the uncertainty types and values to use in the Uncertainty Calculation Values dialog. If several sequences are selected, choose the Single uncertainty type option to calculate uncertainty for each sequence individually. Press OK to begin the uncertainty analysis.

4. Wait for the calculation to complete and press OK.

5. To view the uncertainty stored in the database, select a sequence (or all sequences for project uncertainty) and choose Display → Uncertainty from the popup menu. (Either the current case or both the current and base case uncertainty values will be displayed, depending on whether a base case update has been performed. See Section 2 for a discussion of the base case update feature.)

6. To view detailed quantile information, choose either the Current or Base Quantile Values button.

4.9.3 Generating Uncertainty for End States

Very few PRAs provide end state uncertainty. Again, it may be difficult to compare SAPHIRE results with those reported in a PRA because there will be an expected variability between the uncertainty runs, depending on the algorithms used, the number of samples, and the seed numbers chosen. The sample database provides the seed number and was developed on SAPHIRE using the Monte Carlo algorithm and, therefore, it should be possible to reproduce the uncertainty results. The flat file results for end state uncertainty are shown in table x.

Uncertainty can only be produced after sequence and end state cut sets have been generated. Further discussions on uncertainty analysis are found in both the SAPHIRE User's Guide and the SAPHIRE Technical Reference Manual.

The procedure for generating end state uncertainty requires

1. Choose the End States main menu option.

2. Highlight the end state(s) and choose the Uncertainty option from the popup menu.

3. Select the uncertainty types and values to use in the Uncertainty Calculation Values dialog, then click OK.

4. Wait for the calculation to complete and press OK.

5. To view the uncertainty stored in the database, select an end state and choose Display → Uncertainty from the popup menu. (Either the current case or both the current and base case uncertainty values will be displayed, depending on whether a base case update has been performed. See Section 2 for a discussion of the base case update feature.)

6. To view detailed quantile information, choose either the Current or Base Quantile Values button.

4.9.4 Generating Uncertainty for Groups of Sequences or the Project

Most PRAs provide sequence uncertainty, but only a few may perform uncertainties on groups of sequences that are not grouped previously by end state. In addition, some PRAs provide the results of a project uncertainty. The procedure is the same to generate either groups or project uncertainty and, therefore, is presented together. It may be difficult to compare SAPHIRE results with those reported in a PRA because there will be an expected variability between the uncertainty runs, depending on the algorithms used, the number of samples, and the seed numbers chosen. The sample database provides the seed number and was developed on SAPHIRE using the Monte Carlo algorithm and, therefore, it should be possible to produce the same results.

Uncertainty can only be produced after sequence cut sets have been generated. Further discussions on uncertainty analysis are found in both the SAPHIRE User's Guide and the SAPHIRE Technical Reference Manual.

The procedure for generating sequence group or project uncertainty requires the following:

1. Choose the Sequences main menu option.

2. Highlight the group of sequences (or all sequences) and choose the Uncertainty option from the popup menu.

3. Select the uncertainty types and values to use in the Uncertainty Calculation Values dialog. Choose the group uncertainty type option. (Note that group uncertainty will not be stored in the database for later reporting.) If all sequences in the project are selected for uncertainty analysis, an option to perform project uncertainty analysis will be available. (Project analysis results will be stored in the database for later reporting.) Press OK to begin the uncertainty analysis.

4. When the analysis is complete, the results will be displayed in a results dialog. If the group uncertainty option was selected, this dialog is the only place where the group results will be available. Press the OK button when finished viewing the results.

5. To view the project uncertainty stored in the database, select all sequences and choose Display → Uncertainty from the popup menu. (Either the current case or both the current and base case uncertainty values will be displayed, depending on whether a base case update has been performed. See Section 2 for a discussion of the base case update feature.)

6. To view detailed quantile information, choose either the Current or Base Quantile Values button.

4.10 Additional Features

This section discusses additional features that may be necessary in the data loading process, such as house events, change sets, mutually exclusive events, process flags, and importance measures.

The features in this section have proved useful for manipulation of the PRA databases. The sample database is limited in the amount of additional features that can be demonstrated in it while maintaining its simplicity. The features discussed briefly in this section include

1. Use of change sets (Section 4.10.1)

2. Use of house events (Section 4.10.2)

3. Use of process flags (Section 4.10.3)

4. Use of mutually exclusive event features (Section 4.10.4)

5. Use of flag sets (Section 4.10.5)

6. Use of importance measures (Section 4.10.6).

4.10.1 Use of Change Sets

Change sets are used to modify the current case basic event data to accommodate special situations (such as sensitivity analysis) in the data analysis. Modifications made possible by change sets include individual probability changes to a basic event and class probability changes to a group of basic events. A number of different change sets can be added to a database and many combinations of change sets can be implemented. These change sets, containing information about the probability/class changes, can be applied to basic events during system or sequence analysis. Change set modifications are used most often for setting house events with a calculation type F or T in the FAILURE DATA field and PROCESS FLAGS X and Y. A detailed description and example of using a change set is provided in Appendix A.

4.10.2 Use of House Events

A house event is useful in turning on and off sections of a fault tree. For example, often a system is modeled with AC power available. Given that a PRA is modeling a scenario where the offsite power had failed, then sections of the system may become unavailable for accident mitigation. A house event can be used to turn on and off those applicable sections of a system fault tree to provide the correct model. See the SAPHIRE Technical Reference Manual for a detailed description of how house events are used in SAPHIRE.

4.10.3 Use of Process Flags

The use of process flags allows the analyst to manipulate the evaluation of success and failure logic in the event tree analysis. A detailed description of the use of both the "X" and "Y" flags is provided in Appendix A.

4.10.4 Use of Mutually Exclusive Event Features

The Mutually Exclusive Top feature allows you to define impossible or undesired cut sets and automatically remove them from sequence cut sets. This approach can be traceable and less tedious than using the cut set editor. As an example to illustrate this feature, consider two pump trains in parallel, and each pump train has a test and maintenance (TM) outage event modeled in the fault tree logic. If the technical specifications did not allow both pumps to be in a TM outage during the operating mode represented in the event tree sequence to be analyzed, then the cut sets produced by the fault tree logic that included the TM of both pumps would not be correct and should be deleted.

SAPHIRE provides a variety of methods to remove mutually exclusive events. The preferred method, discussed below, is to write a recovery rule to remove any cut sets containing the mutually exclusive events. The other methods are discussed in the SAPHIRE User's Guide, Appendix G.

The recovery rules feature is so named because its initial purpose was to add appropriate recovery events to cut sets. However, the recovery rule feature has the ability to manipulate cut sets in general. This includes the removal of mutually exclusive events.

The following recovery rule could be written to remove the mutually exclusive events described in our example:

```
|  when two events named TMEVENT1 and TMEVENT2 occur in same cut set,
|  delete that cut set.

if  TMEVENT1 and TMEVENT2 then
  DeleteRoot;
endif
```

Appendix A more fully discusses the recovery rules feature.

4.10.5 Use of Flag Sets

The Flag Set feature automates the ability to specify flag sets that are sequence specific. This feature allows the same fault tree logic to work differently for various situations, depending on the particular setting of the house events in the logic. To illustrate the usefulness of this feature, consider an event tree sequence where the initiating event includes the loss of diesel power, and the fault trees called by the event tree include diesel power dependencies. If the basic event for failure of the diesel to run is DG-FR, then setting the DG-FR calculation type to True (failed) will effectively ensure that the diesel is not credited with successful operation even if there are other basic events that could also cause failure of the diesel generator. Without the Flag Set feature, you would need to build a change set or modify the database with DG-FR set to True, perform the Generate Changes option, and generate sequences cut sets for only the appropriate sequence or

sequences. For each sequence (or group of sequences) having special house event settings, these steps would have to be repeated.

To use the Flag Set feature, you would build a flag set (it is very similar to a change set) using the following process:

1. Choose the Modify → Flag Set main menu option

2. Choose Add from the popup menu, and enter a name for the flag set. For this example, we will name the flag set FLAGDG. Press OK to add the flag set to the database.

3. Then, with the flag set highlighted, press the Flags button, and choose Add from the popup menu.

4. Select one or more basic events (in our example, select DG-FR), then choose Add from the popup menu.

5. Enter the desired house event type or process flag to assign to the selected basic event(s) (in our example we set the calculation type to T for True, meaning guaranteed to fail) and press OK.

There are two ways to specify which event tree sequences should use the FLAGDG flag set. The first way is to assign the flag set to the sequence using the Modify main menu option:

1. Choose the Modify →Event Trees main menu option.

2. Select the event tree (WORK), and press the Sequences button.

3. Select the appropriate sequence and choose Modify from the popup menu.

4. Enter the name of the flag set (FLAGDG) into the flag set field and press OK.

The second way is to write a link rule that assigns the flag set to sequences that meet specified fault tree success/failure criteria, such as the following rule:

```
| to any sequences in event tree named ET in which the fault tree
| named LOSS-DG fails, assign the flag set named FLAGDG.
|
if LOSS-DG then
  eventree(ET) = flag(FLAGDG);
endif
```

The second method provides the advantage of retaining sequence - flag set relationships even when event tree logic changes and must be re-linked to create modified sequence logic.

You can review the flag set names specified for each sequence in the Reports menu by generating a sequence logic report. You can also use the MAR-D extract feature (in the Utilities menu) to extract or load the flag set name using the sequence attribute file (SQA).

Using the Flag Set feature, all sequences in the project can be generated in one-step while still ensuring that each sequence uses the proper house event settings. This method is traceable and involves less user manipulation to ensure that cut sets for each sequence are generated with the proper house event settings.

4.10.6 Use of Importance Measures

Importance measures are sometimes included with the PRA documentation. Importance measures can be used to help determine if sequence cut sets produced by the SAPHIRE database match the PRA document. SAPHIRE importance measures can be compared to the PRA document to see if the number of occurrences of each basic event in the PRA sequences is equal to those generated by SAPHIRE or if there is a mismatch.

This page left blank.

Appendix A

Procedures for Database Loading

A. Procedures for Database Loading

Procedure: Create a Project

To Add a Project Named SAMPLE.

1. Select the File → New Project option, as shown in Figure A- 1. Or, alternatively, click the New toolbar button. The New Project dialog will appear, as shown in Figure A- 2.

2. Click Yes when asked to create a new project. The New Project - Name Project dialog will appear, as shown in Figure A- 3.

3. Type the project name, as shown in Figure A- 1. Click OK. The New Project - Directory Info dialog will appear as shown in Figure A- 4.

Click OK to accept the default location for the project. Or, use the Browse button to select a different location for the new project, and then press OK. The new project is then created and selected.

The project name and folder will appear on the title bar of the SAPHIRE window.

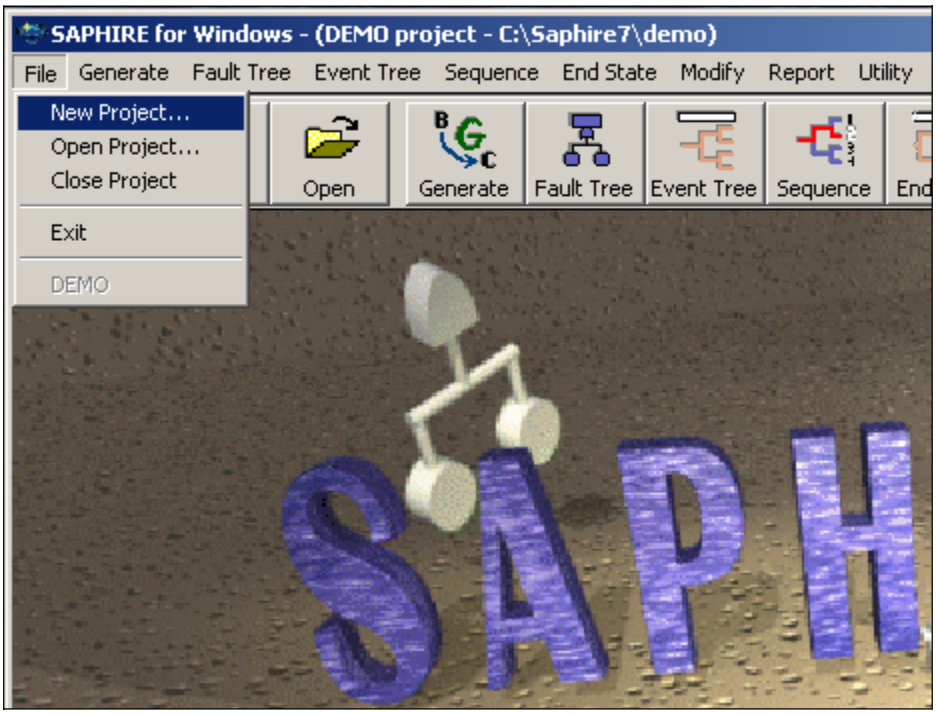


Figure A- 1 SAPHIRE New Project menu.

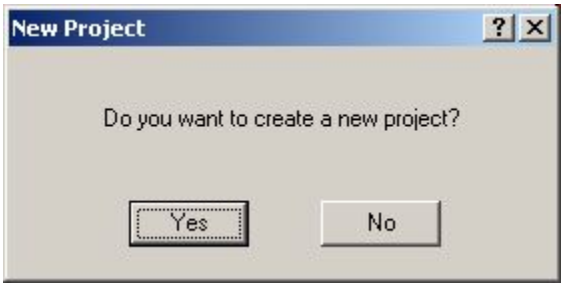


Figure A- 2 New Project prompt.

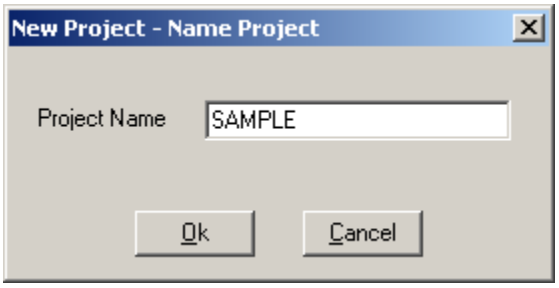


Figure A- 3 New Project Name dialog.

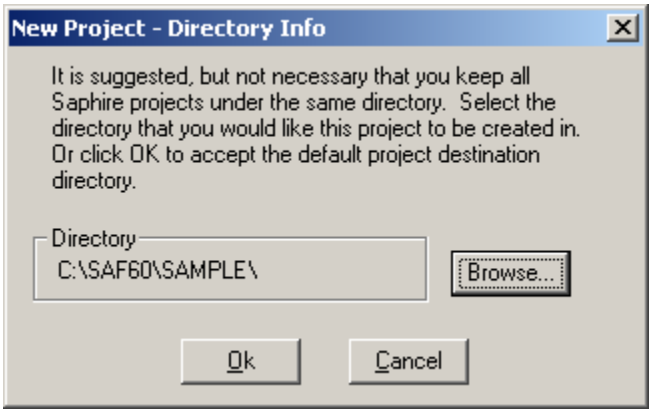


Figure A- 4 New Project Directory Info dialog.

Procedure: Loading and Extracting Flat Files

This procedure describes loading and extracting flat files.

Extracting Files

You may extract flat files from the interactive database by using either MAR-D or SAPHIRE. All extracted files will be sent to the subdirectory related to the project that is currently selected. The default names for extracted files are shown in Section 2.

To extract a flat file:

1. Select the Utility → Load and Extract menu option, as shown in Figure A- 5. Or alternatively, click the Utility toolbar button, followed by the MAR-D toolbar button. The Load and Extract Data dialog will appear, as shown in Figure A- 6.

2. Select the Extract button located in the Data Action area at the top left of the dialog.

3. Select the desired extraction Data Type from the left side of the dialog. The File Type on the right side of the dialog will change to show the available extraction options for the currently selected Data Type. Figure A- 6 illustrates the available Basic Event File Type options.

4. Select the desired Data Type.

5. Click the Process button. For Data Types other than All (version 7 only) and Project, a list of the selected Data Type items will be displayed. Figure A- 7 illustrates a Basic Events extraction dialog.

6. Mark one or more desired extraction items (Ctrl-A to select all).

7. Click the Extract button. The Get Output Destination dialog will be displayed, as shown for Basic Event descriptions in Figure A- 8.

8. To accept the default file name, click the OK button. You may first rename the file, but the extension should not be changed. In version 7 only, use the Browse button to select an alternative folder in which to create the extracted file.

The selected data will be extracted to an ASCII flat file.

Caution: SAPHIRE will overwrite any existing file with the extracted file of the same name.

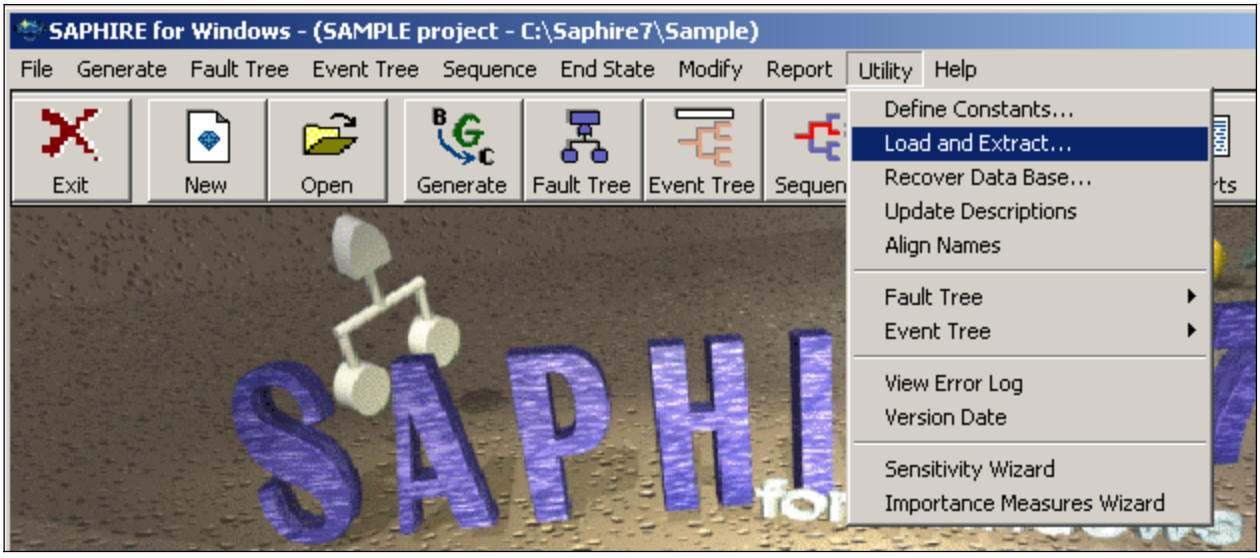


Figure A- 5 Load and Extract menu option.

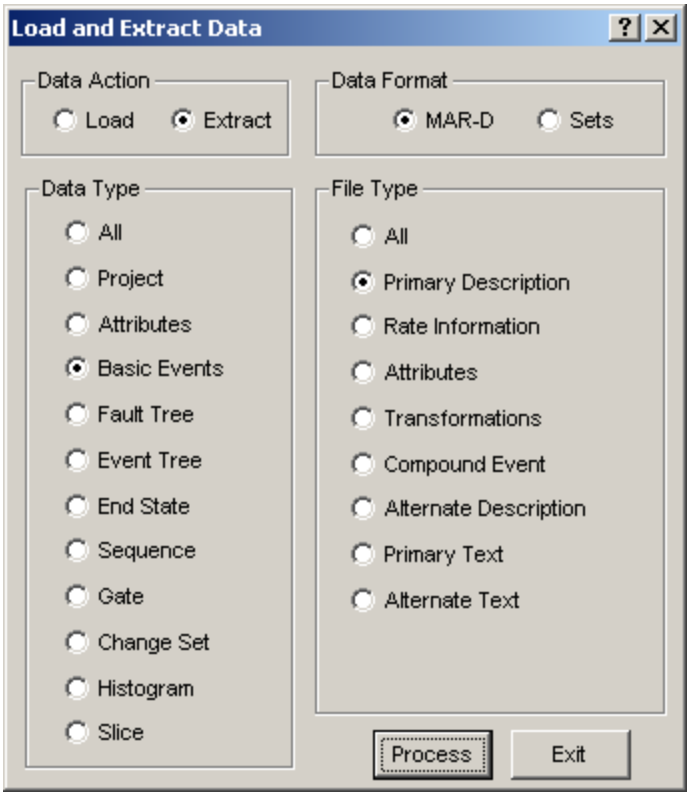


Figure A- 6 Extract menu option.

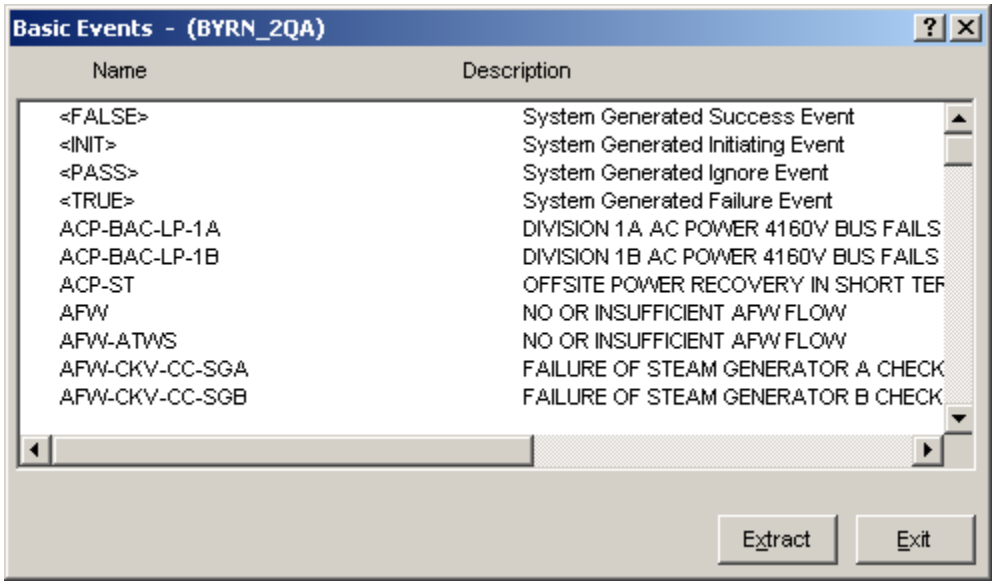


Figure A- 7 Example of a Basic Events extraction dialog.

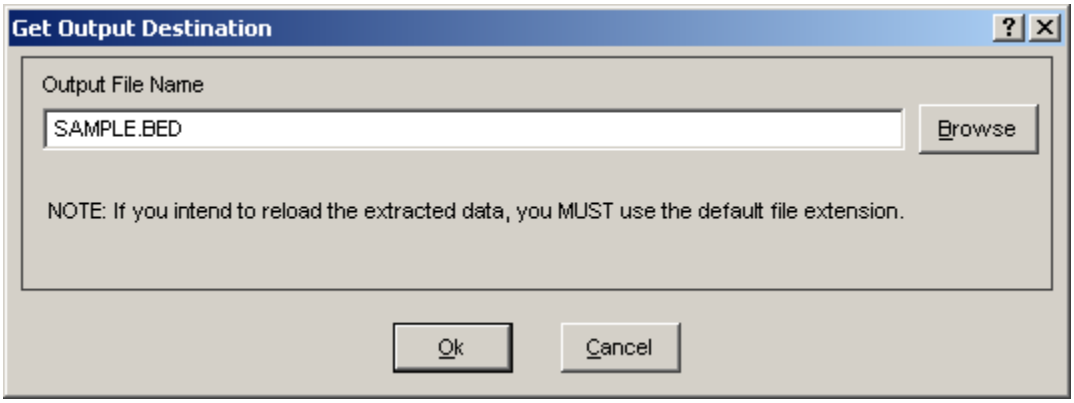


Figure A- 8 Example of an output destination prompt.

Loading Files

You may also use the Load And Extract option to load data into a project. After creating flat files in an ASCII format, you may load these files back into a database. In version 6 you must copy the flat files from the project folder from which they were extracted into the project folder to which you want to load them. In version 7, you may load flat files from any project folder.

To load a flat file:

1. Select the Utility → Load and Extract menu option, as shown in Figure A- 5. Or alternatively, click the Utility toolbar button, followed by the MAR-D toolbar button. The Load and Extract Data dialog will appear, as shown in Figure A- 9. Select the Load button located in the Data Action area at the top left of the dialog.

2. Select the desired load Data Type from the left side of the dialog. The File Type on the right side of the dialog will change to show the available load options for the currently selected Data Type. Figure A- 9 illustrates the available Basic Event File Type options.

3. Select the desired Data Type and click the Process button.

    a. In version 6, the Load File dialog will list the flat files in the current project folder that are available to be loaded for the selected Data Type. See Figure A- 10.

    b. In version 7, the Select an input file dialog will list the flat files available to be loaded for the selected Data Type. Files from any folder can be selected for loading. See Figure A- 11.

4. Select the desired flat file, and press Load (version 6), or Open (version 7).

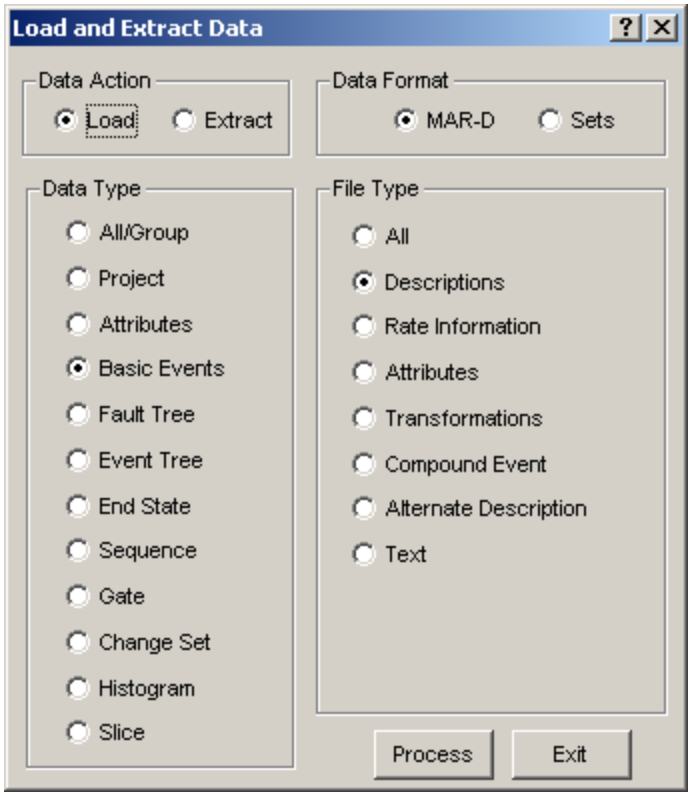


Figure A- 9 Load menu option.

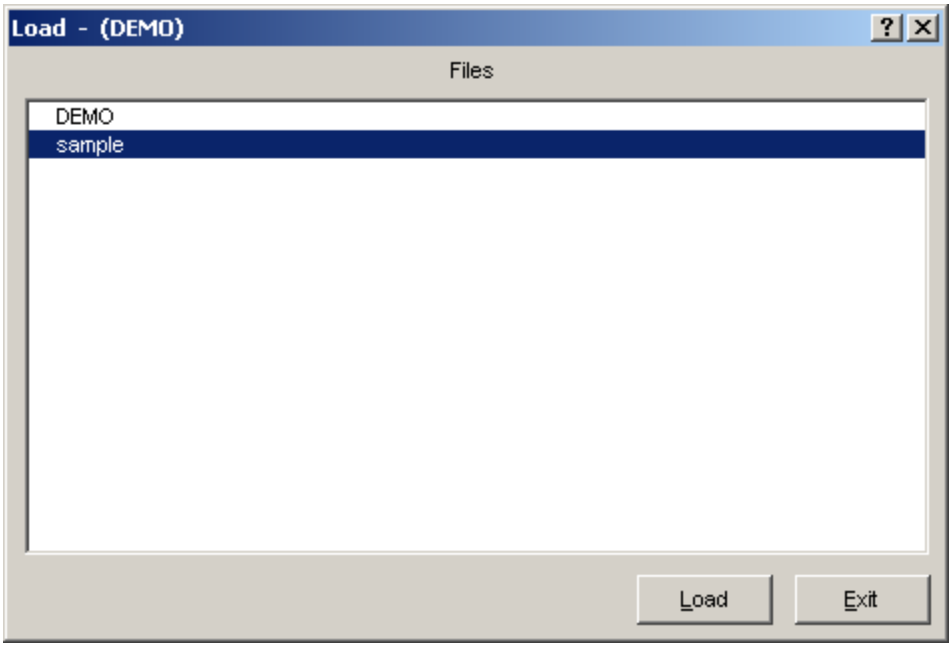


Figure A- 10 Version 6 Load data prompt.

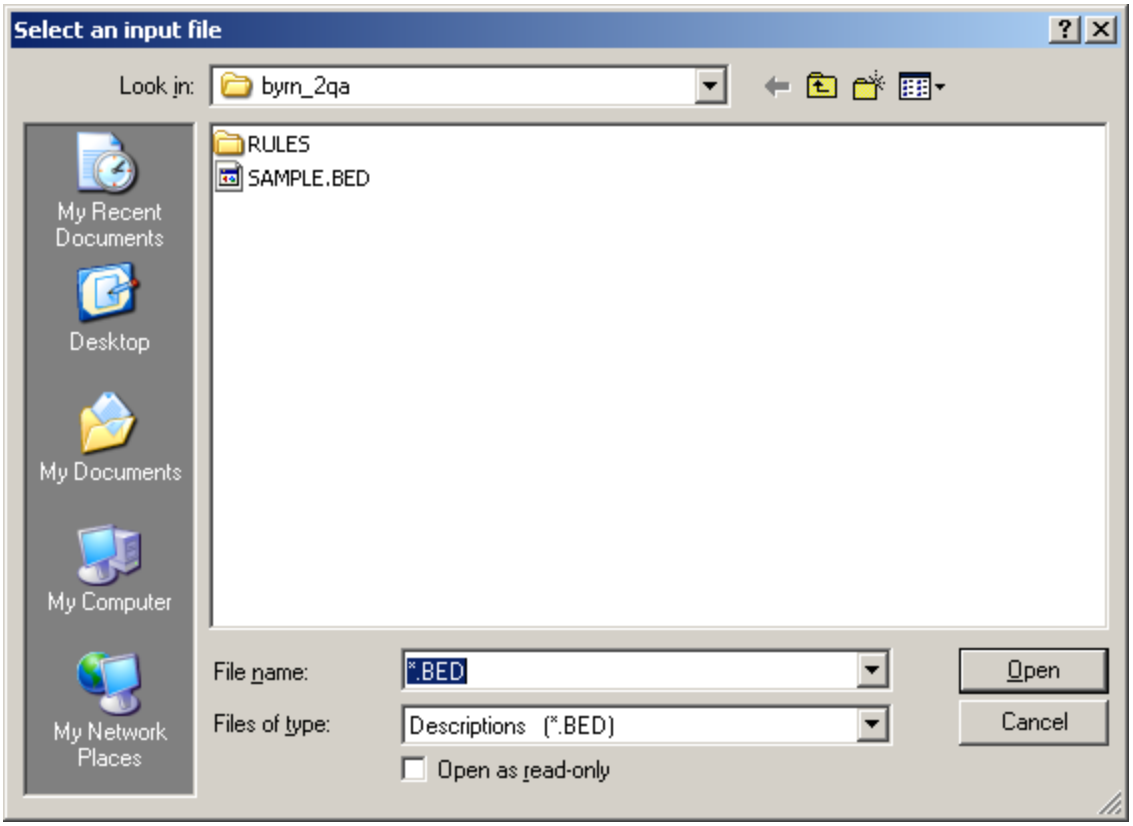


Figure A- 11 Version 7 Load data prompt.

Procedure: Alpha to Graphics Conversion Process

This procedure describes the alpha to graphics conversion.

The fault tree data entry is complicated by the fact that SAPHIRE utilizes an interactive database. Information entered in the process of graphical fault tree construction is implemented in many areas of the program. Graphical data structure is translated into logic, and other information is entered into the interactive database using internal lists. Such information includes the type of gates and basic events used, the textual descriptions entered in gate and basic event boxes, and the textual descriptions added for a fault tree description. The information on these internal lists can subsequently be extracted into SAPHIRE flat files. Conversely, SAPHIRE can be used to build fault tree graphics from logic and descriptions entered in the database using the alpha to graphics conversion.

It is important to note that for cut set generation and quantification, SAPHIRE uses only the logic and not the graphical representation of the fault tree. SAPHIRE can create logic from any fault tree graphics that are built, and the graphics are useful for easy visualization of the system.

When a newly developed fault tree is graphically saved, a .DLS file is automatically created in the project folder. The graphics file is translated into internal fault tree logic. Fault tree descriptions, basic event names and descriptions, and gate names, descriptions, and attributes are loaded into SAPHIRE internal database format.

When a fault tree is initially created using the graphical editor and saved, SAPHIRE will place the textual description of the top gate of the fault tree into the fault tree description. Thereafter, graphical editor modifications to fault tree, gate, and event description will *not* be automatically transferred to the internal lists.

If, however, an alpha to graphics conversion is done, SAPHIRE will use the fault tree, gate, and event descriptions from the internal lists to create the graphical picture of fault tree logic.
The .DLS file contains fault tree graphical information. To view and modify a fault tree, the file (.DLS) for that fault tree must be available on the subdirectory. Once the graphics logic has been loaded into the interactive database, it is not necessary to have the graphics available (for cut set generation and quantification). The .DLS files can be cleared and extracted utilizing the Extract Graphics option from the Utility → Fault Tree menu option.

To use the alpha to graphics conversion:

Load the fault tree, gate, and basic event files into the data base, including:

- .FTD (fault tree descriptions)
- .FTL (fault tree logic)
- .GTD (gate descriptions)
- .BED (basic event descriptions)

Perform the alpha to graphics conversion:

1. From the main menu, choose Utility ▯ Fault Tree ▯ Alpha to Graphics (Figure A- 12). The Alpha to Graphics dialog will be displayed (Figure A- 13).

2. Select the fault tree(s) to be converted.

3. Click the Convert button. SAPHIRE will prompt you concerning the use of tables and boxed events. Press OK to continue.

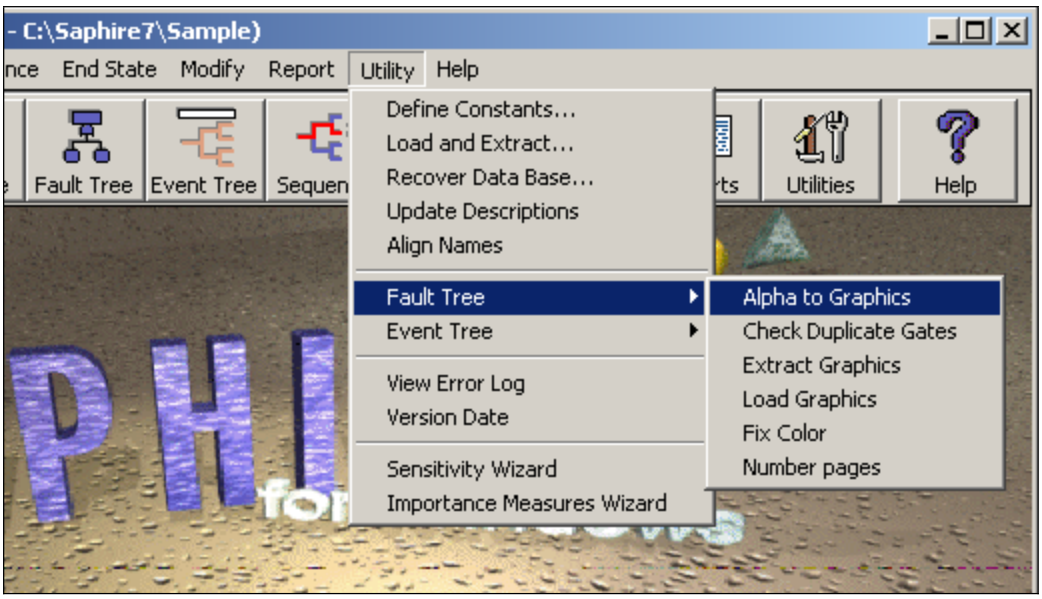


Figure A- 12 Alpha to Graphics menu option.

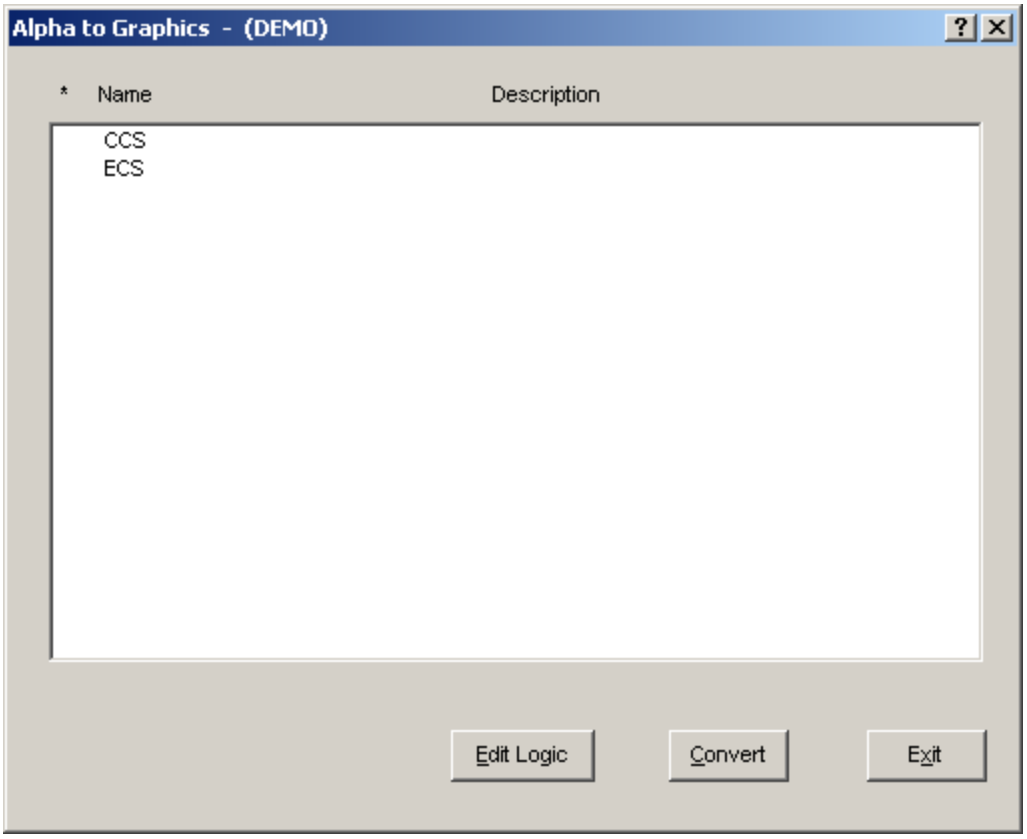


Figure A- 13 Alpha to Graphics dialog.

Procedure: Generate Event Data

This procedure describes the process of generating event data. This process will update the current case Values with the base case values.

All basic event data entered in the SAPHIRE Modify Event database is automatically placed in the base case database. When loading other values, SAPHIRE will allow information update to the base case and/or the current case database. Unless queried during the process, any analysis performed using the SAPHIRE program defaults to values and/or cut sets drawn from the current case database. Note that basic fault tree and event tree logic remains the same for both cases.

To generate event data:

1. Select the Generate menu option from the main SAPHIRE window (Figure A- 14). The Generate dialog will appear (Figure A- 15).

2. If necessary, modify the default mission time at the bottom of the dialog.

3. Click the Generate button. The dialog will close and the status bar of the main window will indicate when the operation is complete.

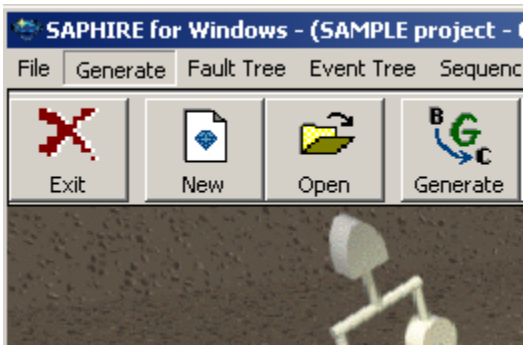


Figure A- 14 The Generate menu option.

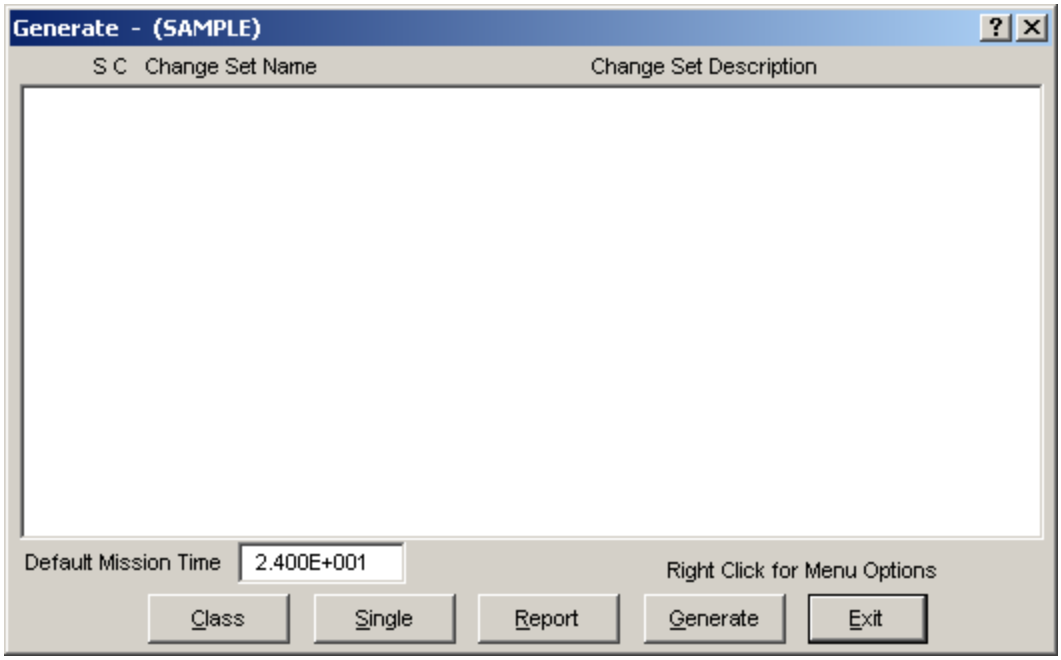


Figure A- 15 The Generate dialog.

Procedure: Change Sets

This procedure describes the process of adding and marking a change set Called CS-1. Implementing a change set allows you to perform a sensitivity study. The event values in the base case data are modified during the generate changes processes for use in the current case data.

This process includes:

1. Adding the change set

2. Making a class change to the CS-1 change set and/or making a probability change to the CS-1 change set

3. Marking the CS-1 change set and generating changes.

Change sets are used to manipulate the base case event data to examine the changes in the probabilities of plant accidents and accident sequence failures based on basic events. You may also generate change sets to be applied to basic events for later propagation through sequence cut sets generation.

IMPORTANT

SAPHIRE contains information in two databases, the current (or working case) and the base case. They are not necessarily the same. You may copy the base case data into the current case for analysis without modifying the base case data with a change set.

Current case data are ALWAYS used in any cut set quantifications.

Adding a Change Set

To add a change set named CS-1:

1. Select the Generate menu option from the main SAPHIRE window (Figure A- 14). The Generate dialog will appear (Figure A- 15).

2. Right click to invoke a pop up menu, and select the Add option. The Add Change Set dialog will appear.

3. Type the change set name (CS-1) and a description of your choice in the fields provided.

4. Click the OK button. The change record name will now appear in the list on the Generate dialog.

Making a Class Change

This option allows you to change event data parameters for a specified grouping of events. The event class is defined by entering data in the Event Attributes data fields. The more of these fields that are filled in, the finer the class definition becomes.

To use this option, you must have already added a change set as described above.

To make a class change:

1. Select the change set (in this case CS-1) and press the Class button. The Edit Event Class dialog will appear as shown in Figure A- 16.

2. Type NAME* in the Name field in the Event Attributes Mask section of the Class Change menu. (The asterisk (*) is a wildcard that acts as a substitute representing a whole word or a group of characters.)

3. Click on the Prob/Freq/Median Fail Accel field and type the new Probability.

4. Click OK. The dialog will close and the message "Class change added " will appear on the status bar of the main window.

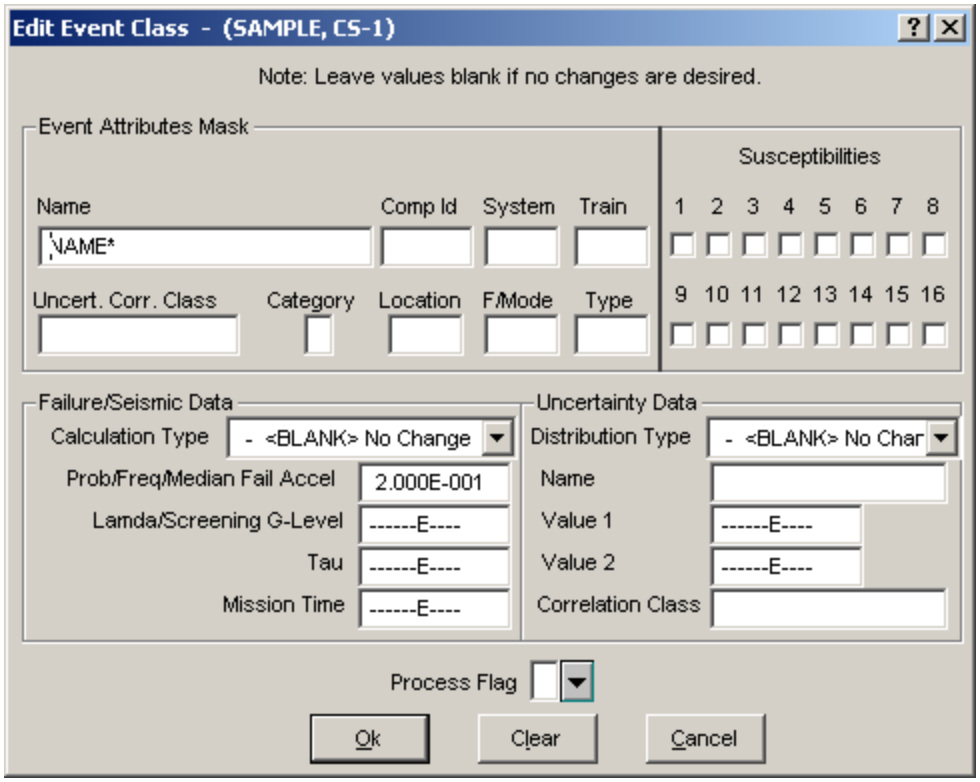


Figure A- 16 The Edit Event Class dialog.

Making a Single Change

This option gives you the flexibility to experiment with setting different basic event failure and uncertainty data. These data values may be set for a single event or for a specified group of events.

To use this option, you must have already added a change set as described above.

To make a probability change:

1. Select the change set (in this case CS-1) and press the Single button. The Change Set Events dialog will appear as shown in Figure A- 17.

2. Right click to invoke a pop up menu, and select the Add option. The Select Change Event dialog will appear (see Figure A- 18).

3. Select one or more basic events to include in the change set. Right click to invoke a pop up menu and select the Add menu option. The Event Probability Changes dialog will appear, as shown in Figure A- 19.

4. Click on the Mean Failure Probability field and type the new Probability.

5. Click the OK button. The selected events will appear in the Change Set Events list. (The letter "S" will appear to the left of the event(s). Notice that the letter "C" is also present when a class change also applies to the event. Note the symbol explanations at the bottom of the dialog.)

6. Click the Exit button to return to the Generate dialog.

   Or

   If you want to continue the probability change process, repeat steps #2 through #4 immediately after step #4 since you are still in the Single Change mode.

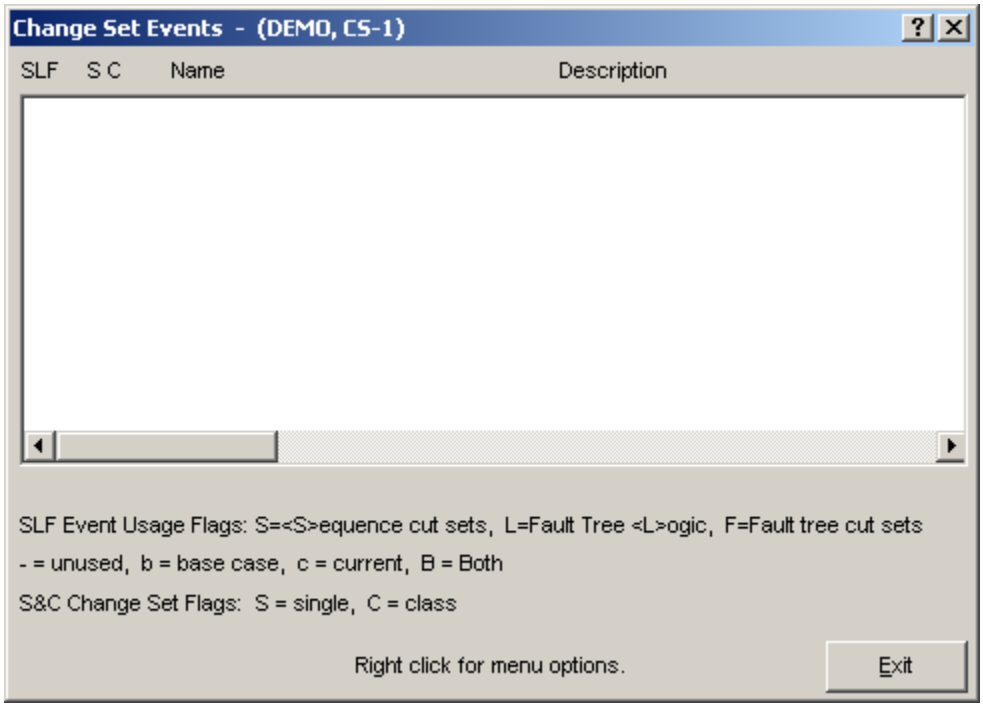


Figure A- 17 Change Set Events dialog.

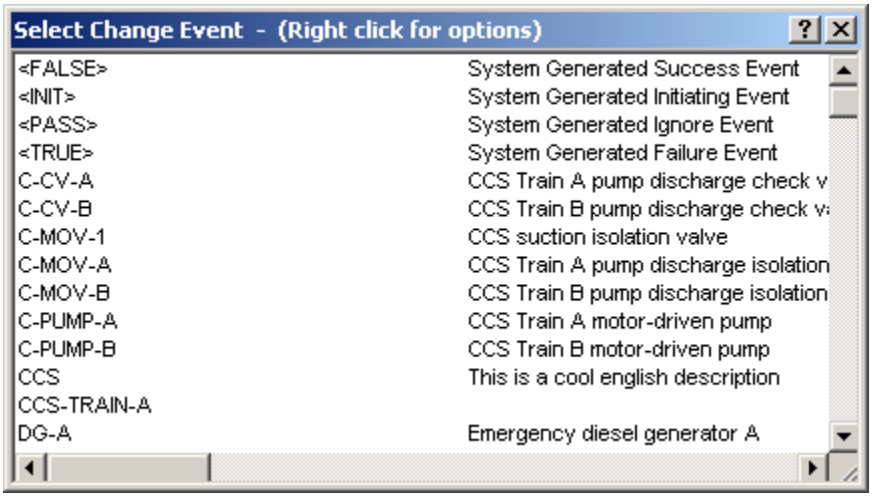


Figure A- 18 Select Change Event dialog.

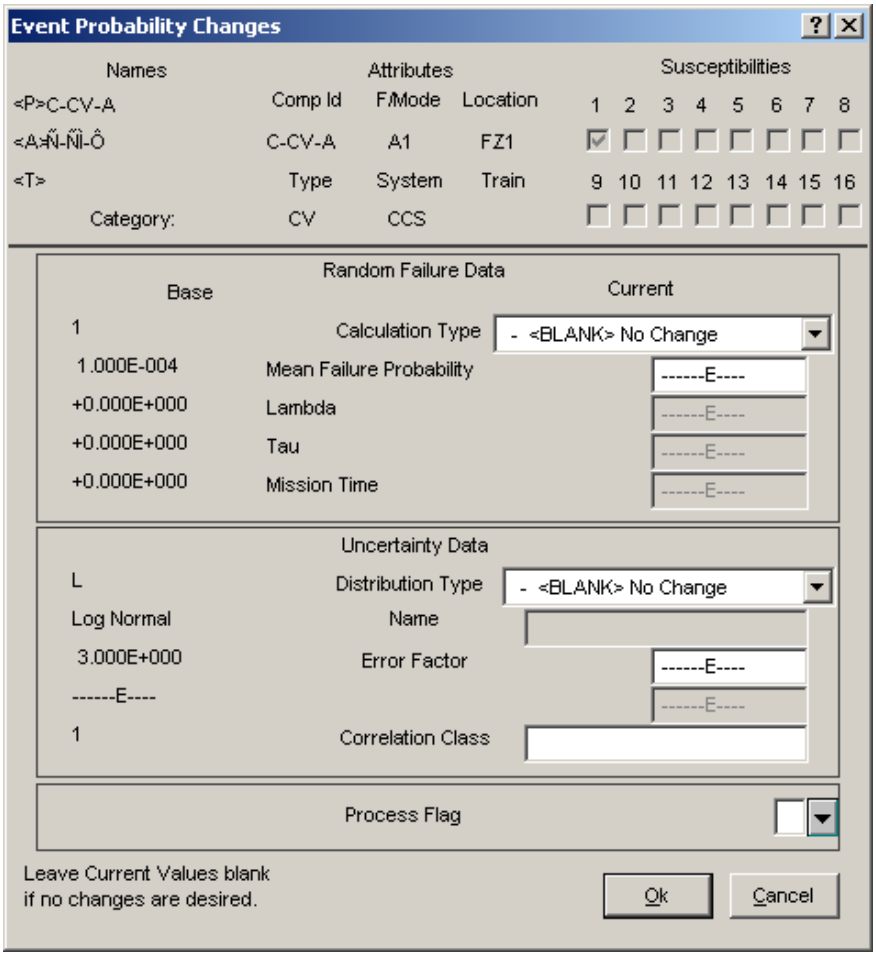


Figure A- 19 Event Probability Changes dialog.

Marking a Change Set

The mark option allows you to mark/unmark the change sets to be used during the generation process. If more than one change set is marked, then the probability and class changes in the change sets marked with the highest number will take precedence over any change from lower numbered change sets.

To use this option, you must have already added a change set as described above. Additionally, either a class or probability change should be implemented.

To mark a change set (CS-1):

1. Select CS-1 and right click to invoke a pop up menu. Select the Mark/Unmark menu option. (Or, double click the CS-1 change set.) A number (1) should appear in the far-left corner by the change set. See Figure A- 20.

2. Click the Generate button. The dialog will close and the status bar of the main window will indicate when the operation is complete.

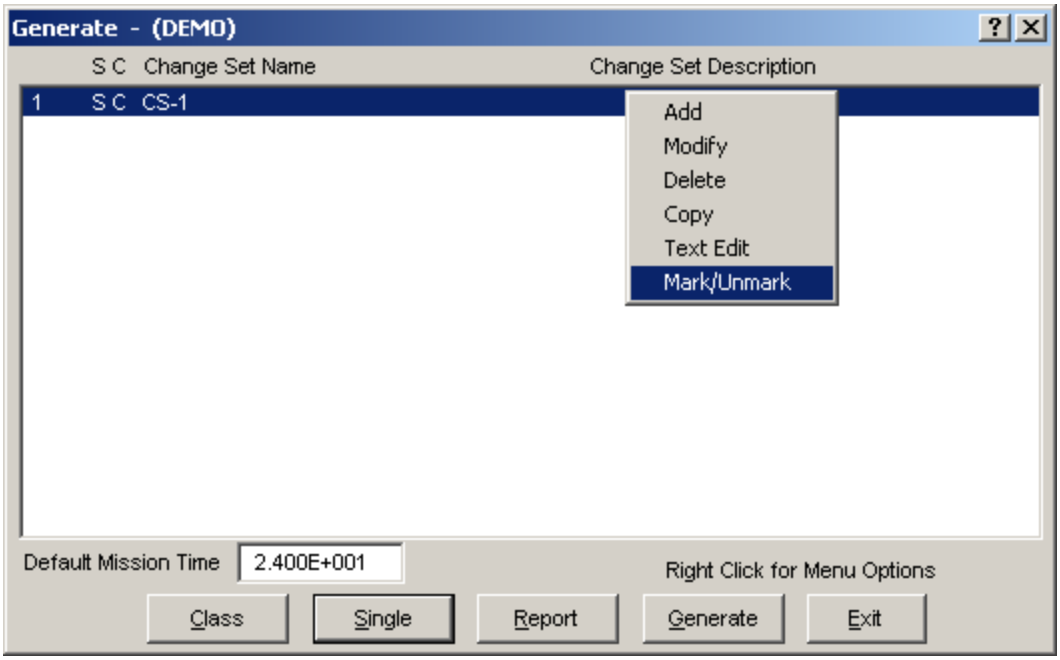


Figure A- 20 Marking a change set.

IMPORTANT

The change set will remain marked until you unmark it by selecting the Mark/Unmark menu option from the pop up menu or by double clicking the change set.
Marked sets are ALWAYS used to generate changes.

Procedure: X And Y Flags

This procedure describes the process of using the X and Y process flags.

Using the "Y" Process Flag

This section discusses the concept and use of the "Y" process flag. The Process Flag options appear in the Modify Event menu (Figure A- 22).

1. Add and select a project called COINTOSS.

2. Once in this database, you will need to create the event tree shown in Figure A- 23. This event tree will calculate the probability of combinations of heads (H) and tails (T) from tossing a coin twice. Possible combinations that can occur are HH, HT, TH, or TT. This can be observed in the sequence and end state names shown in Figure A- 24.

3. Since this is a simple problem, we can calculate by hand the probability for all possible end states. As is shown in Figure A- 24, the probability of tossing two heads is 0.25, the probability of tossing a head and a tail is 0.5 and the probability of tossing two tails is 0.25. This sums to 1.0 across all the possible sequences.

4. Enter the top event values. In the process of creating this event tree, SAPHIRE has added IE-TOSS as an initiating event and two top events (TOSS1 and TOSS2) as developed events into the Basic Event listing (under Modify Database → Basic Events). For calculation, it is necessary to enter the probability of a failure for TOSS1 and TOSS 2 as 0.5. Since SAPHIRE looks at failure space, this will be the probability of a tail given an unbiased coin. Leave the process flag space blank. The initiating event IE-TOSS should be 1.0 as default. Accept the default.

5. Generate changes (by clicking the Generate button from the Generate menu and dialog) to update the probability values that you just entered into the base case to the current (or working) case.

6. Generate sequence logic (by selecting the Event Tree main menu option and invoking the Link Trees popup menu item).

7. Generate sequence cut sets (by selecting the Sequence main menu option and invoking the Solve popup menu item). Accept the SAPHIRE defaults and click OK.

8. View the results. As shown in Figure A- 24, the results are not what was anticipated. This is because, by default, SAPHIRE will generate *sequences* utilizing both success and failure logic. This will include either the developed event or fault tree (if one was associated with that top event). But only the failure logic is presented and the failure probabilities are used in the quantification process.

9. Entering a process flag will allow you to indicate exactly how you want SAPHIRE to handle developed events (and/or fault trees). The "Y" process flag uses the developed event value for the probability of failure and 1 minus the probability of failure of the developed event for the success.

10. Enter a "Y" as the process flag for both TOSS1 and TOSS2. Repeat steps 4 to 7 and view results. As shown in Table A-2, the calculated results are now correct.

Toss	Result
HH	.25
HT	.25
TH	.25
TT	.25

Figure A- 21. Expected coin toss results.

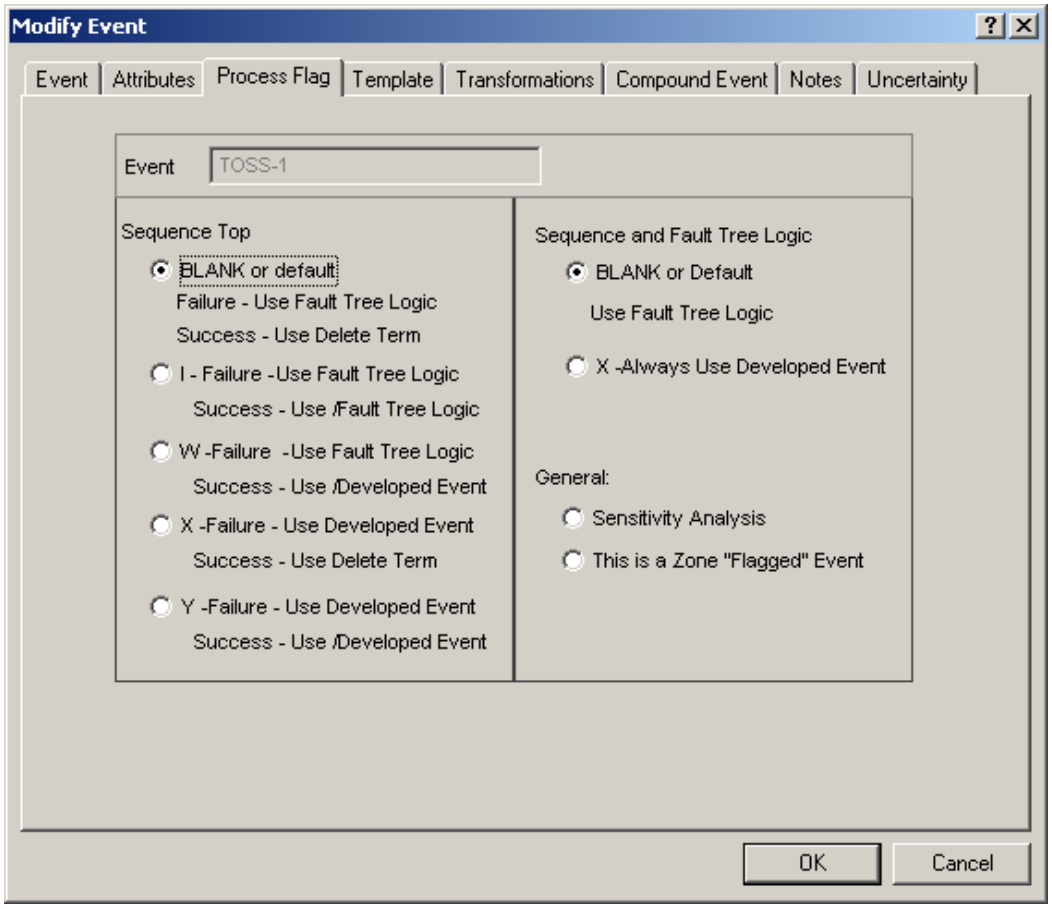


Figure A- 22. SAPHIRE 7.0 Modify Event / Process Flag tab.

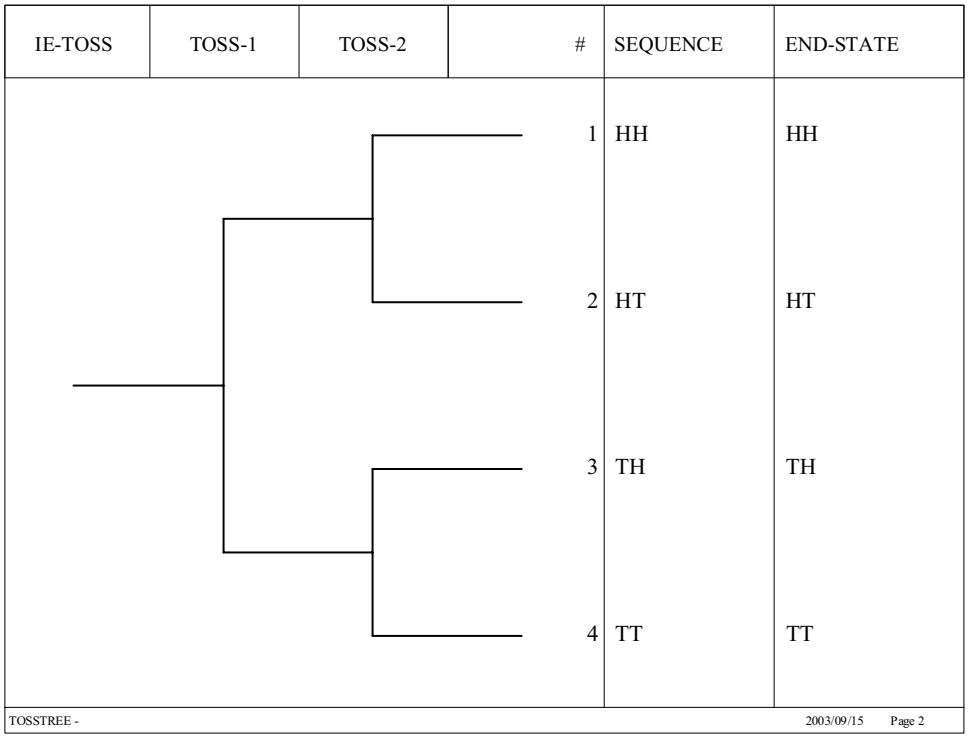


Figure A- 23. Tosstree event tree.

Standard Report

SEQUENCE CUT SETS REPORT

Project: COINTOSS
Analysis: RANDOM

Event Tree	Sequence	% Total	Cut Set %	Prob./Freq.	Inputs
TOSSTREE	1	100.0	100.0	1.0E+000	<PASS>
TOSSTREE	2	100.0	100.0	5.0E-001	TOSS-2
TOSSTREE	3	100.0	100.0	5.0E-001	TOSS-1
TOSSTREE	4	100.0	100.0	2.5E-001	TOSS-1, TOSS-2

Figure A- 24 SAPHIRE sequence cut sets report using default process flags.

Standard Report

SEQUENCE CUT SETS REPORT

Project: COINTOSS
Analysis: RANDOM

Event Tree	Sequence	% Total	Cut Set %	Prob./Freq.	Inputs
TOSSTREE	1	100.0	100.0	2.5E-001	/TOSS-1, /TOSS-2
TOSSTREE	2	100.0	100.0	2.5E-001	/TOSS-1, TOSS-2
TOSSTREE	3	100.0	100.0	2.5E-001	TOSS-1, /TOSS-2
TOSSTREE	4	100.0	100.0	2.5E-001	TOSS-1, TOSS-2

Figure A- 25 SAPHIRE sequence cut sets report using "Y" process flags.

Using the "X" Process Flag

The "X" flag behaves in the following manner. With the process flag for an event tree top event set to "X", SAPHIRE links in the associated system success and failure fault trees and solves the boolean logic. When reporting the results, successful system basic events are ignored although they have been accounted for in the logic.

Procedure: Recovery Rules

This procedure describes the process of developing and using the recovery rules.

This section discusses the concept, development, and use of the recovery rules. The Recovery Rules options appear in the Fault Trees List popup menu (Figure A- 26) and in the Sequences list for event tree/sequence rules. Recovery rules can be attached to a particular fault tree or across all fault trees in a project (Figure A-27). Likewise, they can be attached to a particular sequence, a particular event tree, or across all event trees in a project. The attachment level (or scope) of the rules determines the cut sets that may be affected by the rules.

The discussion of recovery rules will be by way of a few simple examples of the rules along with a detailed overview of the recovery rule keywords and symbols (see Table A-3).

The SAPHIRE recovery rules are "free-form" logic rules that allow you to alter or delete fault tree or sequence cut sets. Although called "recovery rules," the recovery rules have developed from the simple addition of recovery events onto specified cut sets into a powerful rule-based system for cut set manipulation. Thus, the "recovery rules" can now be used for advanced probabilistic risk assessment techniques such as (1) the automated addition of sequence recovery events, (2) the addition of common-cause cut sets, and (3) the elimination of mutually exclusive events (e.g. restricted or impossible combinations of events).

The rules are entered in a free-form text editor within SAPHIRE (similar to Notepad or WordPad). The rules have a very specific format and have certain command keywords that can be used. The structure of the rules and the keywords that are available are discussed below. It should be pointed out that the rules can be exported and loaded through MAR-D.

The rules follow a format similar to the structure that is found in traditional programming languages (e.g., BASIC, MODULA-2, or C). As such, the ability exists to define "macros" and "if...then" type of structures. But, before discussing the particular structure of the rules, it is best if the keywords and symbols were defined. Table A-3 contains a list of keywords and symbols that are used in rule editor. This table also includes the definition and usage of each keyword and symbol. Within the "usage" column in Table A-3, the particular keyword or symbol that is being presented is shown in bold face. Words or phrases that are italicized are intended to represent a particular command or group of commands and, as such, should not be included as part of the rule. Instead, an appropriate command (e.g., a specified search criteria, a keyword, or a logic expression) should replace the italicized text.

Now that the keywords and symbols have been defined, the structure of the rules will be discussed. This discussion will take place by way of specific examples.

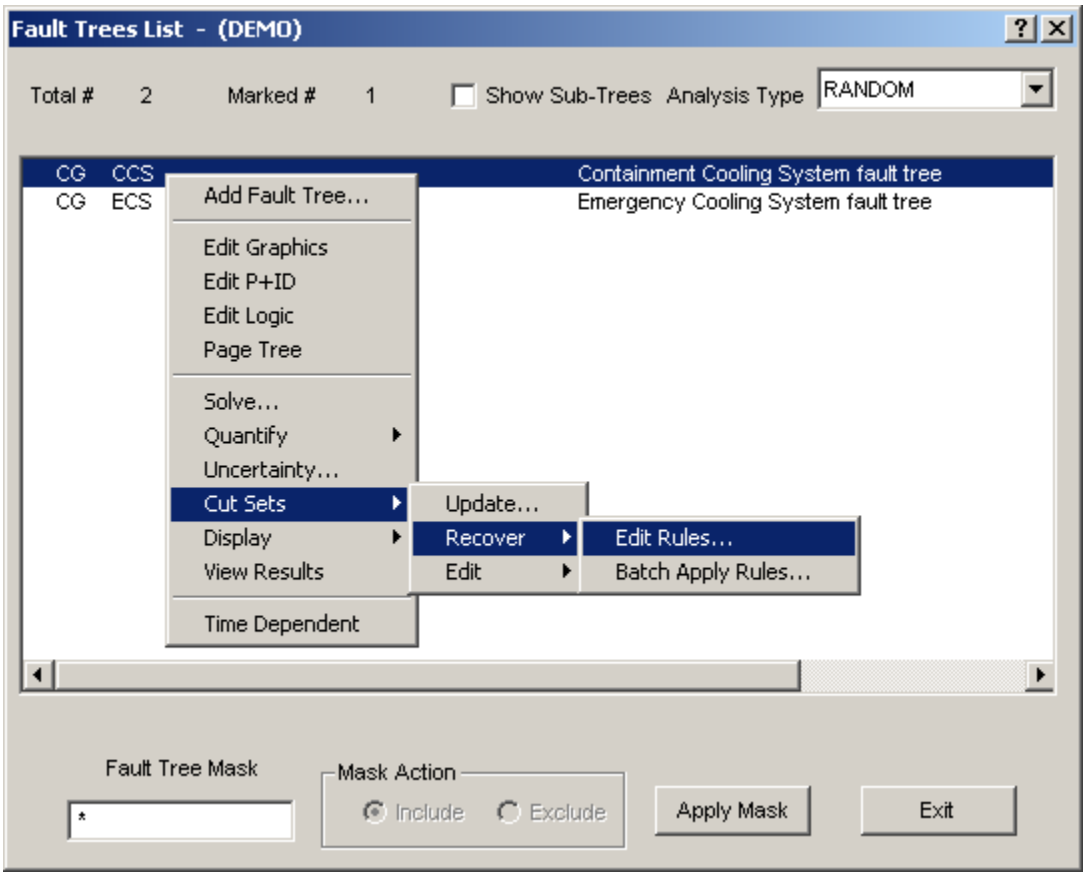


Figure A- 26 Fault Tree recovery rule menu option.

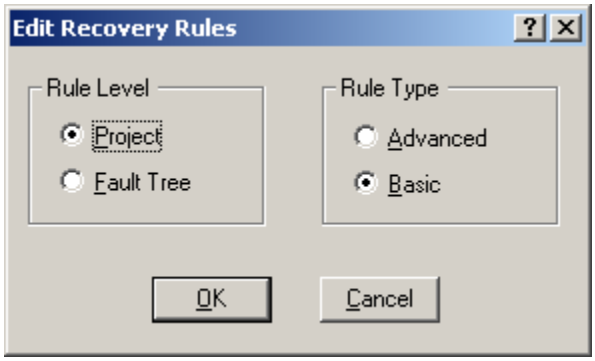


Figure A- 27 Fault tree recovery rules options.

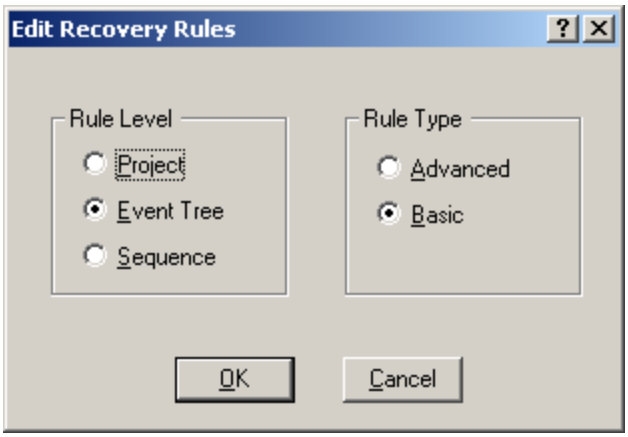


Figure A- 28 Event tree recovery rules options.

Table A- 1. List of keywords or symbols used in the SAPHIRE recovery rules.

Keyword or symbol	Definition	Usage

if then	Keyword pair that indicates that a search criteria is being specified. Note that the keywords are lower case.	if "search criteria" then
		perform some action on each cut set...;
		endif

endif	Keyword that indicates the end of a particular rule. Note that the keyword is one word and is lower case.	if "search criteria" then
		perform some action on each cut set...;
		endif

always	Keyword that indicates that every cut set that is being evaluated satisfies the search criteria. Note that the keyword is lower case.	if always 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. Note that the keyword is lower case.	if "search criteria" then
		recovery = NAME-OF-RECOVERY-EVENT;
		endif

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

~	Symbol used in the search criteria to indicate that a particular event will not be in the cut set that is being evaluated.	if (~SEARCH-CRITERIA) "and other search criteria" 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 either 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) "and 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!

Table A- 1. List of keywords or symbols used in the SAPHIRE recovery rules.

Keyword or symbol	Definition	Usage
;	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 either an initiating event, basic event, macro, or logic expression.
+	Symbol to indicate the logical OR command.	if SEARCH-CRITERIA1 + SEARCH-CRITERIA2 then The search criteria will be satisfied for all cut sets that match either SEARCH-CRITERIA1 or SEARCH-CRITERIA2. The SEARCH-CRITERIA# may be either 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].

AddEvent =	Key word that indicates that an event will be added to the cut set being evaluated. Note the particular capitalization of the keyword.	if "search criteria" then
		AddEvent = EVENT-NAME;
		endif

DeleteEvent =	Keyword that indicates that an event will be deleted from the cut set being evaluated. Note the particular capitalization of the keyword.	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 will then become the cut set that is being evaluated. Note the particular capitalization of the keyword.	if "search criteria" then
		NewCutset;
		now make additions to the empty cut set...
		endif

DeleteRoot;	Keyword that indicates that the original	if "search criteria" then

Table A- 1. List of keywords or symbols used in the SAPHIRE recovery rules.

Keyword or symbol	Definition	Usage
	cut set (i.e., the cut set that satisfied the search criteria) will be deleted. Note the particular capitalization of the keyword.	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 will then become the cut set that is being evaluated. Note the particular capitalization of the keyword.	if "search criteria" then
		CopyCutset;
		now make modification to a copy of the cut set...
		endif

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

MACROS	A macro is a user-definable keyword that specifies specified search criteria. Note that the macro name must be all upper-case, must be 24 characters or less, and must not include any of the SAPHIRE-restricted characters (e.g., a space, *, ?, \, /). The macro definition line can wrap around to more than one line, but the end of the macro must be indicated with a semicolon.	MACRO-NAME = SEARCH-CRITERIA;
		
		if MACRO-NAME "and other search criteria" then
		perform some action on each cut set...;
		endif
		
		The SEARCH-CRITERIA may be either an initiating event, basic event, macro, or logic expression.

Example: Add a recovery event to all cut sets.

The example shown in Table A- 2 demonstrates how the recovery rules can be used to add recovery actions. The rule in this example adds the recovery action NRAC-12HR to every cut set for a particular sequence. Consequently, this rule would have to be typed into the event tree sequence rule editor for the sequence of interest.

In addition, from within the rule editor, putting the cursor on NRAC-12HR and pressing <ALT>R will add the event into the database as a recovery action.

Table A- 2 Example: Add a recovery event to every cut set in a sequence.

```
| A rule to apply NRAC-12HR recovery event to all
| cut sets in the sequence.
if always then
  recovery = NRAC-12HR;
endif
```

Example: Add a recovery event to certain cut sets.

The second example (Table A- 3) demonstrates how recovery actions can be added to certain cut sets based on the attributes of the cut set and sequence. This example will be added as an event tree project rule (meaning it may affect all sequences in the project), but will key on only one of two different initiating events. Also, this example will demonstrate the use of a macro (the macro is called KEY-ON-INIT).

Table A- 3 Example: Add a recovery event to each cut set having a specified initiator.

```
| Define a macro in order to pick up only those
| sequences that have LOSP or SBO as initiators.
KEY-ON-INIT = init(LOSP) + init(SBO);
| Search on either the LOSP or SBO and basic events.
| DG-A or DG-B.
if KEY-ON-INIT * (DG-A + DG-B) then
  recovery = NRAC-12HR;
endif
```

Example: Add common cause failure cut sets.

The third example (Table A- 4) demonstrates how the "recovery" rules could be used to add common-cause events to the cut sets. Example 3 defines a search criterion that identifies the failure combination of two auxiliary feedwater pumps (pump A and pump B). If these two basic events are found in a cut set, then a new cut set will be created that substitutes the independent failures of the two pumps with a single common-cause basic event. Note that the original cut set will be retained, since the two failures could still be independent. This rule could be placed in either (or both) the fault tree project rules or the event tree project rules.

Table A- 4 Example: add common cause failure cut sets.

```
| Define a macro in order to pick up only those cut sets that
| have combinations of AFW-PUMP-A and AFW-PUMP-B.
CCF-AFW-PUMPS = AFW-PUMP-A * AFW-PUMP-B;
| Search for the AFW pump basic events and replace
| with a CCF event.
if CCF-AFW-PUMPS then
  | First make a copy of the original cut set.
  CopyRoot;
  | Now remove the two independent failure events.
  DeleteEvent = AFW-PUMP-A;
  DeleteEvent = AFW-PUMP-B;
  | Now add the CCF event.
  AddEvent = AFW-PUMP-CCF;
endif
```

Example: Remove a cut set.

The last example (Table A- 5) demonstrates how a particular cut set could be completely removed from the cut set list. There may be instances in which a cut set should be removed because the combination of basic events should not occur (say for example, 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.

Table A- 5 Example: Remove a cut set.

```
| Define a macro in order to pick up only those cut sets that
| have combinations of two diesel generators out for maintenance.
DIESELS-IN-MAINT = DG-A-MAINT * DG-B-MAINT
| Search for the maintenance events and then delete cut set.
if DIESELS-IN-MAINT then
  | Delete the cut set.
  DeleteRoot;
endif
```

The List menu option is a timesaving feature of the rule editor. It provides the ability to select and insert items from the database directly into the rule. To use this feature choose the Lists → Events menu option to open a list of events in the database. Place the cursor where the event is to be inserted in the rule, then select the desired event in the list and double click it. To add an event that does not yet exist in the database, right click on the event list to invoke a popup menu, and select the add option. You will be prompted to create an event, which will then appear in the event list.

In summary, the recovery rules provide a powerful rule-based method to modify fault tree or sequence cut sets. The keywords and symbols for the rules were defined in Table A- 1. The examples presented above suggest the potential applications that can be performed using the SAPHIRE rule.

Appendix B

General MAR-D Data Interchange Formats

B. General MAR-D Data Interchange Formats

This appendix enumerates the formats for each of the various data interchange formats as of August 2005.

Except where noted, file formats are the same for SAPHIRE versions 6 and 7. The primary version differences occur in the name and description files. This is because version 7 has the capability to provide an alternate name and description for each data type. In addition, descriptions in version 6 are 60 characters long, whereas in version 7, they are 120 characters long.

The file formats are backward compatible: version 6 files can be successfully loaded into version 7. It is not recommended that version 7 files be loaded into version 6, due to the presence of subtle format and content changes.

General Format Rules

1. All name references (project names, event names, etc.) must be upper-case alphanumeric. All lower-case characters will be converted to upper-case. Any alpha fields that are longer than the format specified will be truncated. No spaces are allowed in the middle of names.

2. Descriptions can have both upper-case and lower-case characters. No character checking will be done. No commas are allowed in the description.

3. Commas are used as field delimiters in most formats and can be used as placeholders for unknown fields. Any number of leading and trailing field spaces can be inserted. Exceptions to this format are detailed as needed.

4. Text rules:

    1. File is standard ASCII text, single spaced, upper- and lower-case.
    2. ^EOS signals the End of Section so that multiple names in the same project can be collected in one file.

These rules apply to all files unless specifically stated otherwise.

Project (Plant) Information

Project Names and Descriptions (Version 6)

File Name:
xxxxxx.FAD
File Format:
name,description
where

Name	24 character	Project name
description	60 character	Project description

Project Names and Descriptions (Version 7)

File Name:
xxxxxx.FAD
File Format:
name,description[,A]
where

name	24 character	Project name
description	120 character	Project description
A	1 character	If included indicates alternate description

Project Attribute File

File Name:
xxxxxx.FAA
File Format:
project=
name,mission,newSum,co,loc,type,design,vendor,AE,OpDate,QualDate
where

name	24 character	Project name
mission	Floating point	Default mission time in hours
newSum	Floating point	New sequence frequency sum
Co	10 character	Company name
Loc	16 character	Location name
type	3 character	Facility type
design	10 character	Facility design
vendor	5 character	Vendor name
AE	10 character	Architectural Engineer
OpDate	(yyyy/mm/dd)	Operational date

QualDate	(yyyy/mm/dd)	Qualification date

Project Recovery Rules

File Name:
xxxxxxxx.FAY
File Format:
project =
-- recovery rule text --
where

project	24 character	Project name

System (Fault Tree) Recovery Rules

File Name:
xxxxxxxx.FAS
File Format:
project =
-- recovery rule text --
where

project	24 character	Project name

Project Partition Rules

File Name:
xxxxxxxx.FAP
File Format:
project =
-- partition rule text --
where

project	24 character	Project name

Project Textual Information (Version 6)

File Name:
xxxxxx.FAT
File Format:
Project =
-- text --
where

project	24 character	Project name

Project Textual Information (Version 7)

File Name:
xxxxxx.FAT
File Format:
Project [,A] =
-- text --
where

project	24 character	Project name
A	1 character	If included indicates alternate description

Basic Event Information

Event Names and Descriptions (Version 6)

File Name:
xxxxxx.BED
File Format:
project =
name,description
. . . , . . .
where

project	24 character	Project name
name	24 character	Event primary name
description	120 character	Alphanumeric description

Event Names and Descriptions (Version 7)

File Name:
xxxxxx.BED
File Format:
project =
name,description[,A]
. . . , . . .
where

project	24 character	Project name
name	24 character	Event primary name
description	120 character	Alphanumeric description
A	1 character	If included indicates alternate description

Basic Event Rate Information (Version 6)

File Name:
xxxxxx.BEI
File Format:
project =
name, calc, udC, udT, udV, prob, lambda, tau, mission, init, Flag, udV2
. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . .
where

Project	24 character	Project name
Name	24 character	Basic event name
Calc	1 character	Calculation type

1	Probability
2	same as type 3
3	1 Exp(Lambda * Mission Time)
4	same as type 5
5	Operating component with full repair
6	same as type 7
7	1+(EXP(Lambda*Tau) 1.0)/(Lambda*Tau)
T	Set to House Event (Failed, Prob=1.0)
F	Set to House Event (Successful,Prob=0.0)
I	Set to ignore
S	Use fault tree min cut upper bound
E	Use end state min cut upper bound
G	Seismic event - Enter g level for screening
H	Seismic event - Use medium site hazard curve for screening

UdC	4 characters	Uncertainty correlation class Events in same class are 100% correlated.
UdT	1 character	Uncertainty distribution type

L	Log normal, error factor
N	Normal, standard deviation
B	Beta, b of Beta(a,b)
D	Dirichlet, b of Dirichlet(b)
G	Gamma, a Gamma(a)
C	Chi-squared, degrees of freedom
E	Exponential, none
U	Uniform, Upper end pt.
H	Histogram
M	Maximum entropy
S	Seismic Log Normal
O	Constrained non-informative

UdV	Floating point	Uncertainty distribution value
Prob	Floating point	Probability value
Lambda	Floating point	Basic event failure rate per hr.
Tau	Floating point	Time to repair in hours
Mission	Floating point	Mission time
init	Boolean	Initiating event flag (Y/N)
Flag	1-character	process flag
udV2	Floating point	Uncertainty distribution value #2

General Rules:

1. The name field is mandatory.

Basic Event Rate Information (Version 7)

File Name:
xxxxxx.BEI
File Format:
project =
name, calc, udC, udT, udV, prob, lambda, tau, mission, init, Flag, udV2
. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . .
where

Project	24 character	Project name
Name	24 character	Basic event name
Calc	1 character	Calculation type

1	Probability
V	Value event (input to compound event)
3	1 Exp(Lambda * Mission Time)
5	Operating component with full repair
7	1+(EXP(Lambda*Tau) 1.0)/(Lambda*Tau)
T	Set to House Event (Failed, Prob=1.0)
F	Set to House Event (Successful,Prob=0.0)
I	Set to ignore
C	Compound event
S	Use fault tree min cut upper bound
E	Use end state tree min cut upper bound
G	Seismic event - Enter g level for screening
H	Seismic event - Use medium site hazard curve for screening

UdC	24 characters	Uncertainty correlation class Events in same class are 100% correlated.
UdT	1 character	Uncertainty distribution type

L	Log normal, error factor
N	Normal, standard deviation
B	Beta, b of Beta(a,b)

D	Dirichlet, b of Dirichlet(b)
G	Gamma, a Gamma(a)
C	Chi-squared, degrees of freedom
E	Exponential, none
U	Uniform, Upper end pt.
H	Histogram
M	Maximum entropy
S	Seismic Log Normal
O	Constrained non-informative
T	Triangular, mode, upper end of Triangular(m, u)

UdV	Floating point	Uncertainty distribution value
Prob	Floating point	Probability value
Lambda	Floating point	Basic event failure rate per hr.
Tau	Floating point	Time to repair in hours
Mission	Floating point	Mission time
Init	Boolean	Initiating event flag (Y/N)
Flag	1-character	process flag
UdV2	Floating point	Uncertainty distribution value #2

General Rules:

1. The name field is mandatory.

Basic Event Attribute Codes

Basic event attributes are entered through MODIFY--Basic Event and stored in Event.

File Name:
xxxxxx.BEA
File Format:
project =
name,Aname,type,sys,fail,loc,compID,Gname,train,att1,..,att16
. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . .
where

project	24 character	Project name
name	24 character	Event name
Aname	24 character	Alternate event name
type	3 character	Event component type
sys	3 character	Event component system
fail	3 character	Failure mode
loc	3 character	Component location
compID	7 character	Component ID
Gname	24 character	Event group identifier
train	3 character	Train identifier
att1..att16	Class attribute	16 values of Y or N (yes or no) indicate whether the

	flags	attribute described in the class attribute file is applicable.

General Rules:

1. The name field is mandatory.

Basic Event Transformations (Version 6)

In SAPHIRE version 6.0, both transformation and compound information are extracted into and loaded from this file type.

File Name:
xxxxxx.BET
File Format:
project =
name1,level,type, library, procedure
bename1, bename2, . . . ,
. . . , benameN
^EOS
name2,level,type, library, procedure
bename1, bename2, . . . ,
. . . , benameN
^EOS
where

project	24 character	Project name
name	24 character	Event name
level	3 character	Transformation level (0..99)
type	4 character	Transformation type (AND, OR, ZOR, COM, blank)
library	60 character	name of plug in library (for COM events)
procedure	60 character	name of procedure in plug in library (for COM events)
bename1..N	24 character	Event name

Basic Event Transformations (Version 7)

File Name:
xxxxxx.BET
File Format:
project =
name1,level,type
bename1, bename2, . . . ,
. . . , benameN
^EOS
name2,level,type
bename1, bename2, . . . ,

. . . , benameN
^EOS
where

project	24 character	Project name
name	24 character	Event name
level	3 character	Transformation level (0..99)
type	4 character	Transformation type (AND, OR, ZOR, blank)
bename1..N	24 character	Event name

Basic Event Compound Information (Version 7 only)

In SAPHIRE version 7.0, compound information is extracted into its own file type. Compound events can still be loaded from .BET files (where version 6.0 extracts compound information).

File Name:
xxxxxx.BEC
File Format:
project =
name1,level,type
bename1, bename2, . . . ,
. . . , benameN
^EOS
name2,level,type, library, procedure
bename1, bename2, . . . ,
. . . , benameN
^EOS
where

project	24 character	Project name
name	24 character	Event name
level	3 character	0 or blank
type	4 character	COM
library	60 character	name of plug in library
procedure	60 character	name of procedure from plug in library
bename1..N	24 character	Event name

Event Attribute Descriptions

Failure Mode Descriptions (Version 6)

File Name:
xxxxxx.FMD
File Format:
project =
fail, description
. . . , . . .
where

project	24 character	Project name
Fail	3 character	Failure mode primary identifier
description	60 character	Failure mode description

Failure Mode Descriptions (Version 7)

File Name:
xxxxxx.FMD
File Format:
project =
fail,altFail,description[,A]
. . . , . . .
where

project	24 character	Project name
Fail	5 character	Failure mode primary identifier
altFail	5 character	Failure mode alternate identifier
description	120 character	Failure mode description
A	1 character	If included indicates alternate description

Component Type Descriptions (Version 6)

File Name:
xxxxxx.CTD
File Format:
project =
comp, description
. . . , . . .
where

project	24 character	Project name
comp	3 character	Component type primary identifier

description	60 character	Component type description

Component Type Descriptions (Version 7)

File Name:
xxxxxx.CTD
File Format:
project =
comp, altComp, description [,A]
. . . , . . .
where

project	24 character	Project name
comp	5 character	Component type primary identifier
altComp	5 character	Component type alternate identifier
description	120 character	Component type description
A	1 character	If included indicates alternate description

System Type Descriptions (Version 6)

File Name:
xxxxxx.STD
File Format:
project =
sys,description
. . . , . . .
where

project	24 character	Project name
sys	5 character	System primary identifier
description	60 character	System description

System Type Descriptions (Version 7)

File Name:
xxxxxx.STD
File Format:
project =
sys,altSys,description[,A]
. . . , . . .
where

project	24 character	Project name
sys	5 character	System primary identifier
altSys	5 character	System alternate identifier

description	120 character	System description
A	1 character	If included indicates alternate description

Location Descriptions (Version 6)

File Name:
xxxxxx.LCD
File Format:
project =
loc,description
. . . , . . .
where

project	24 character	Project name
loc	3 character	Location primary identifier
description	60 character	Location description

Location Descriptions (Version 7)

File Name:
xxxxxx.LCD
File Format:
project =
loc,altLoc,description[,A]
. . . , . . .
where

project	24 character	Project name
loc	5 character	Location primary identifier
altLoc	5 character	Location alternate identifier
description	120 character	Location description
A	1 character	If included indicates alternate description

Class Attribute Descriptions (Version 6)

File Name:
xxxxxx.TTD
File Format:
project =
attr,description
. . . , . . .
where

project	24 character	Project name
Attr	3 character	Class attribute primary name

description	60 character	Class attribute description

Class Attribute Descriptions (Version 7)

File Name:
xxxxxx.TTD
File Format:
project =
attr,altAttr,description[,A]
. . . , . . .
where

project	24 character	Project name
Attr	5 character	Class attribute primary name
altAttr	5 character	Class attribute alternate name
description	120 character	Class attribute description
A	1 character	If included indicates alternate description

Fault Tree Information

Fault Tree Names and Descriptions (Version 6)

File Name:

xxxxxx.FTD

File Format:

project =
name,description[,s]
. . . , . . .

where

project	24 character	Project name
Name	24 character	Fault tree primary name
description	60 character	Fault tree description
S	1 character	If included indicates fault tree is a sub-tree

Fault Tree Names and Descriptions (Version 7)

File Name:

xxxxxx.FTD

File Format:

project =
name,description[,s][,A]
. . . , . . .

where

project	24 character	Project name
Name	24 character	Fault tree primary name
description	120 character	Fault tree description
S	1 character	If included indicates fault tree is a sub-tree
A	1 character	If included indicates alternate description

Fault Tree Graphics

Fault tree graphics are stored in the block data file of the Graphics relation. The MAR-D file (.DLS) is a display list sequence for the graphics in a binary format. It is loaded and output as is with no conversion performed.

File Name:

xxxxxx.DLS

File Format:

IRRAS 2.5/4.0/5.0, SAPHIRE 6.0/7.0 Fault Tree Graphics file (DLS format)

Fault Tree Logic (Version 6)

Fault tree logic is stored in the block data file of the System relation.

File Name:
xxxxxx.FTL
File Format:

```
project, fault tree =
* gatename1,description
gatename1 gatetype input1 input2 . . . inputn
. . . . . .
* gatenamen,description
gatenamen gatetype input1 input2 . . . inputn
. . .
```

where

Project		24 character	Project name
fault tree		24 character	Fault tree name
Gatename		24 character	Gate name
Gatetype		4 character	Gate type
	AND		logical AND
	OR		logical OR
	TBL		table of events
	TRAN		transfer followed by a 24-character fault tree name
	NAND		logical NOT AND
	NOR		logic NOT OR
	N/M		N out of M logic gate
	CONT		continuation of inputs to the previous gate
Input		24 character	inputs to the gate (event or gate names)
description		60 character	gate name descriptions included as comment

General Rules:

1. A gate definition cannot exceed 255 characters. (Use the CONT gate to break up definitions.)

2. A line beginning with an asterisk (*) is a comment.

3. For each gate name a comment should be included giving the gate description.

Fault Tree Logic (Version 7)

Fault tree logic is stored in the block data file of the System relation.

File Name:
xxxxxx.FTL
File Format:

```
project, fault tree =
* gatename1,description
gatename1 gatetype input1 input2 . . . inputn
. . . . . . . . . . . . . . .
* gatenamen,description
gatenamen gatetype input1 input2 . . . inputn
. . .
```

where

Project		24 character	Project name
fault tree		24 character	Fault tree name
Gatename		24 character	Gate name
Gatetype		4 character	Gate type
	AND		logical AND
	OR		logical OR
	TBL		table of events
	TRAN		transfer followed by a 24-character fault tree name
	NAND		logical NOT AND
	NOR		logic NOT OR
	N/M		N out of M logic gate
	CONT		continuation of inputs to the previous gate
Input		24 character	inputs to the gate (event or gate names)
description		120 character	gate name descriptions included as comment

General Rules:

1. A gate definition cannot exceed 255 characters. (Use the CONT gate to break up definitions.)

2. A line beginning with an asterisk (*) is a comment.

3. For each gate name a comment should be included giving the gate description.

Fault Tree Cut Sets

File Name:
xxxxx.FTC

File Format:
project, fault tree, analysis =
eventname * eventname +
eventname * eventname * eventname *
eventname +
eventname * eventname.
^EOS
project, fault tree2 =

where

project	24 character	Project name	
fault tree	24 character	Fault tree name	
analysis	1 character	Analysis type	
1			Random
2			Fire
3			Flood
4			Seismic
5 through 8			Reserved
9 through 16			user-defined
eventname	24 character	Event names in the cut set	

General Rules:

1. An asterisk (*) separates cut set events. Spaces are ignored.

2. A plus sign (+) separates cut sets.

3. A period (.) denotes the end of a sequence.

4. A slash (/) precedes complemented events.

5. Event names are a maximum of 4 characters including the "/".

6. A line beginning with an asterisk (*) is a comment.

Fault Tree Attributes

File Name:
xxxxx.FTA
File Format:
project, analysis =
name,level,mission,mincut,proCut,sample,seed,sizCut,sys,cuts, events,value1,..,value9
. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . ,
where

project	24 character	Project name	
analysis	1 character	Analysis type	
1			Random
2			Fire
3			Flood
4			Seismic
5 through 8			Reserved
9 through 16			user-defined
name	24 character	Fault tree name	
level	Integer 2	0 = top level tree	
mission	Floating point	Mission time	
mincut	Floating point	Mincut upper bound	
proCut	Floating point	Probability cut off value	
sample	Integer 4	Sample size	
seed	Integer 8	Random number seed	
sizecut	Integer 2	Size cut off value	
sys	3 character	System identifier	
cuts	Integer 5	Base number of cut sets	
events	Integer 5	Base number of events	
value	Floating point	Base uncertainty values	

Fault Tree Recovery Rules

File Name:
xxxxxxxx.FTY
File Format:
project =
-- recovery rule text --
where

project	24 character	Project name

Fault Tree Textual Information (Version 6)

File Name:
xxxxxx.FTT

File Format:

project, fault tree =
-- text --
^EOS
project, fault tree2 =
. . .

where

project	- 24 character	Project name
fault tree	- 24 character	Fault tree name

Fault Tree Textual Information (Version 7)

File Name:
xxxxxx.FTT
File Format:
project, fault tree [,A]=
-- text --
^EOS
project, fault tree2 =
. . .
where

project	24 character	Project name
fault tree	24 character	Fault tree name
A	1 character	If included indicates alternate text

Fault Tree Graphical P&ID

The piping and instrumentation diagrams is a graphics file in binary format. It will be loaded and output as-is: no conversion will be performed.

File Name:
xxxxxxxx.PID
File Format:

IRRAS 4.0/5.0, SAPHIRE 6.0 and 7 .0 P&ID Graphics file (PID Format)

Event Tree Information

Event Tree Names and Descriptions (Version 6)

File Name:
xxxxxx.ETD

File Format:
project =
name,description[,s]
. . . , . . .

where

Project	24 character	Project name
Name	24 character	Event tree name
Description	60 character	Event tree description
S	1 character	If included indicates event tree is a transfer tree

Event Tree Names and Descriptions (Version 7)

File Name:
xxxxxx.ETD
File Format:
project =
name,description[,s][,A]
. . . , . . .
where

Project	24 character	Project name
Name	24 character	Event tree name
Description	120 character	Event tree description
S	1 character	If included indicates event tree is a transfer tree
A	1 character	If included indicates alternate description

Event Tree Attributes

File Name:
xxxxxx.ETA
File Format:
project =
name,init
. . . , . . .
where

project	24 character	Project name

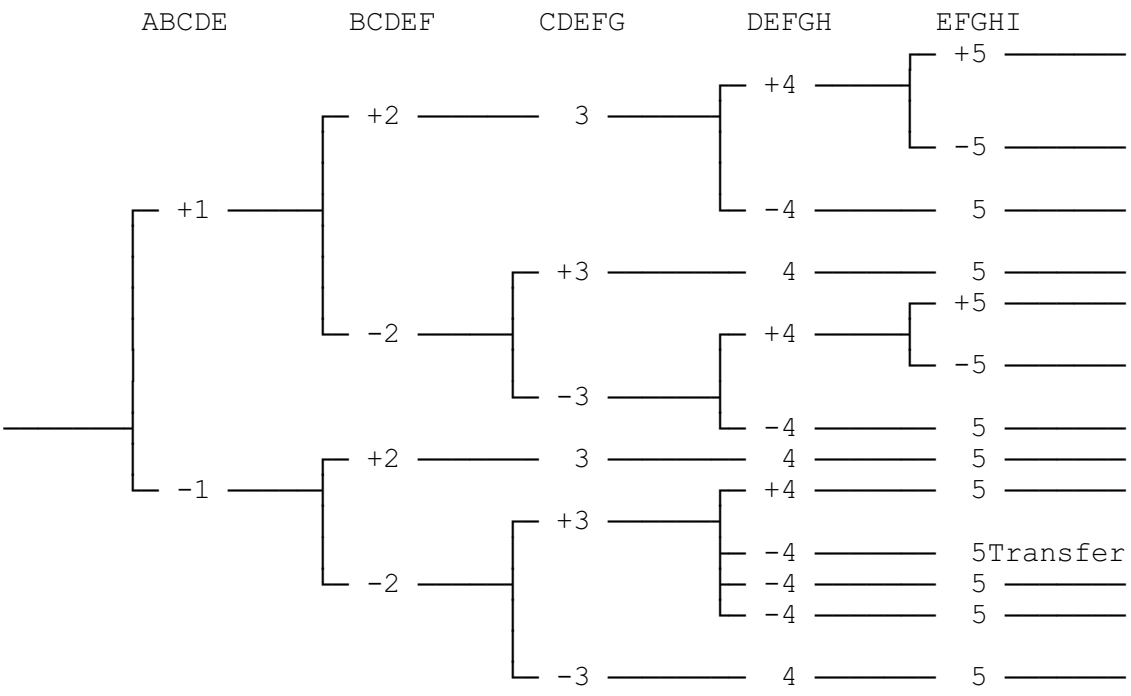
name	24 character	Event tree name
init event	24 character	Initiating Event

Event Tree Graphics

The SAPHIRE Event Tree Graphics file (*.ETG) is a display list sequence for the graphics. Its format and contents are the same as the Event Tree Logic File.

File Name:
xxxxxx.ETG
File Format:
See file format for the Event Tree Logic

SAMPLE GRAPHICAL EVENT TREE



Event Tree Logic

File Name:
xxxxxx.ETL
File Format:
project, event tree, init event [,T] =
^TOPS
*** 1 | 2 | 3 | 4 | 5 | this is a comment**
ABCDE BCDEF CDEFG DEFGH EFGHI
^LOGIC
+1 +2 3 +4 +5
5
4 5
2 +3 4 5
3 +4 +5
5
4 5
1 +2 3 4 5
2 +3 +4 5
4 5
4 5
4 5
3 4 5

^SEQUENCES

Y/N, header#1,	Y/N, header#2,	Y/N, header#3,	Y/N,header#4
Y/N, sequence#1,	Y/N, end state#1,	Y/N, xdata1#1,	Y/N,xdata2#1
Y/N, sequence#2,	Y/N, end state#2,	Y/N, xdata1#2,	Y/N,xdata2#2
Y/N, sequence#3,	Y/N, end state#3,	Y/N, xdata1#3,	Y/N,xdata2#3
Y/N, sequence#4,	Y/N, end state#4,	Y/N, xdata1#4,	Y/N,xdata2#4
Y/N, sequence#5,	Y/N, end state#5,	Y/N, xdata1#5,	Y/N,xdata2#5
Y/N, sequence#6,	Y/N, end state#6,	Y/N, xdata1#6,	Y/N,xdata2#6
Y/N, sequence#7,	Y/N, end state#7,	Y/N, xdata1#7,	Y/N,xdata2#7
Y/N, sequence#8,	Y/N, end state#8,	Y/N, xdata1#8,	Y/N,xdata2#8
Y/N, sequence#9,	Y/N, tran file#9,	Y/N, xdata1#9,	Y/N,xdata2#9, T
Y/N, sequence#10,	Y/N, end state#10,	Y/N, xdata1#10,	Y/N,xdata2#10
Y/N, sequence#11,	Y/N, end state#11,	Y/N, xdata1#11,	Y/N,xdata2#11
Y/N, sequence#12,	Y/N, end state#12,	Y/N, xdata1#12,	Y/N,xdata2#12
Y/N, sequence#13,	Y/N, end state#13,	Y/N, xdata1#13,	Y/N,xdata2#13

^TEXT
SIZE s
JUST j
COLOR j
XY xvalue,yvalue
"120 character line of text"
XY xvalue, yvalue

"120 character line of text"
"120 character line of text"

^PARMS
START yvalue
WINDOW x1,y1,x2,y2
HEADER x1,x2,x3,x4
^EOS
project, event tree2 =
(additional event trees)

where

Project	24 character	Project name
Name	24 character	Event tree name
init event	24 character	Initiating Event
[,T]	1 character	Optional flag indicating init event name is a Top event fault tree
TOPS	24 character	Top event/fault tree names
Y/N	Boolean	End state text displayed?
Header	24 character	Sequence header
Sequence	24 character	Sequence name
Endstate	24 character	End state name
tran file	24 character	Name of transfer file
xdata1	24 character	Information (optional)
xdata2	24 character	Information (optional)

General Rules:

1. A line beginning with an asterisk (*) is a comment.

2. Literal "^TOPS", "^LOGIC", "^SEQUENCES" labels must be present.

3. Logic is built according to the position of the top event in the definition.

   Plus sign (+)---the specified top event succeeded.

   Minus sign ( )---the specified top event failed.

   Blank ( )---the response of the indicated top event did not matter.

4. Header, Sequence name, End State name, Xdata1, Xdata fields associated with each sequence. "Y/N" indicates whether the specified field is visible. A "T" at the end indicates the sequence transfers to another tree.

5. User text is input following the ^TEXT command. Parameters include the size, justification, color, and location of the text block.

6. The ^PARMS command allows input of program control parameters.

Event Tree Rules

File Name:
xxxxxxxx.ETR
File Format:
project, event tree =
-- event tree rule text
. . .
^EOS
project, event tree2
where:

Project	24 character	Project name
Name	24 character	Event tree name
Tops	24 character	Top event/fault tree names

Event Tree Textual Information (Version 6)

File Name:
xxxxxx.ETT

File Format:
project, event tree =
-- text --
^EOS
project, event tree2 =
-- text --

where

project	24 character	Project name
event tree	24 character	Event tree name

Event Tree Textual Information (Version 7)

File Name:
xxxxxx.ETT
File Format:
project, event tree [,A]=
-- text --
^EOS
project, event tree2 =
-- text --
where

project	24 character	Project name
event tree	24 character	Event tree name
A	1 character	If included indicates alternate description

Event Tree Recovery Rules

File Name:
xxxxxxxx.ETY
File Format:
project, event tree =
-- recovery rule text --
^EOS
project, event tree2 =
where

project	24 character	Project name
event tree	24 character	Event tree name

Event Tree Partition Rules

File Name:
xxxxxxxx.ETP
File Format:
project, event tree =
-- partition rule text --
^EOS
project, event tree2 =
where

project	24 character	Project name
event tree	24 character	Event tree name

End State Information

Each sequence can be tied to a single plant damage state. The cut sets for a sequence can be partitioned to map to separate end state. The name and description data are loaded with the SARA *.PDS file.

End State Names and Descriptions (Version 6)

File Name:
xxxxxx.ESD

File Format:
project =
name,description
. . . , . . .

where

project	24 character	Project primary name
name	24 character	End state primary name
description	60 character	End state description

End State Names and Descriptions (Version 7)

File Name:
xxxxxx.ESD
File Format:
project =
name,description[,A]
. . . , . . .
where

project	24 character	Project primary name
name	24 character	End state primary name
description	120 character	End state description
A	1 character	If included indicates alternate description

End State Information

File Name:
xxxxxx.ESI
File Format:
project =
project =
Name, E-QMethod, E-QPasses, R-QMethod, R-QPasses,
. . . . . , ..., ..., ..., ...,
where

project	24 character	Project name
name	24 character	End state name
e-Qmethod	1 character	End state default quantification method
e-Qpasses	Integer 3	End state default min/max quantification passes
r-QMethod	1 character	Quantification method used for current results
r-Qpasses	Integer 3	Min/max quantification passes used for current results

End State Textual Information (Version 6)

File Name:
xxxxxx.EST

File Format:
project, end state =
-- text --
^EOS
project, end state2 =

where

project	24 character	Project name
end state	24 character	End state name

End State Textual Information (Version 7)

File Name:
end-state.EST
File Format:
project, end state[, A]=
-- text --
where

project	24 character	Project name
end state	24 character	End state name
A	1 character	If included indicates alternate description

End State Cut sets

The end state cut sets are the minimal cut sets for end state logic as derived from the fault tree logic. The cut sets are stored in the block data file of the Endstate relation.

The MAR-D end state cut sets are in a format similar to that of the fault tree cut sets.

File Name:
xxxxxx.ENC
File Format:
project, event tree, end state =
eventname * eventname +
eventname * eventname * eventname *
eventname +
eventname * eventname.
^EOS
project, event tree2, end state =
where

Project	24 character	Project name
event tree	24 character	Event tree name
end state	24 character	End state name
Eventname	24 character	Event names in the cut set

General Rules:

1. An asterisk (*) separates events in a cut set. Spaces are ignored.

2. A plus sign (+) separates cut sets.

3. A period (.) denotes the end of the end state cut sets.

4. A slash (/) precedes complemented events.

5. Event names have a maximum of 24 characters including the "/" character for complemented events.

6. A line beginning with an asterisk (*) is a comment.

Sequence Information

Sequence Names and Descriptions (Version 6)

File Name:

xxxxxx.SQD

File Format:

project,eventree =
name,description
. . . , . . .
^EOS

where

project	24 character	Project name
event tree	24 character	Event tree name
name	24 character	Sequence name
description	60 character	Sequence description

Sequence Names and Descriptions (Version 7)

File Name:

xxxxxx.SQD

File Format:

project,eventree =
name,description[,A]
. . . , . . .
^EOS

where

project	24 character	Project name
event tree	24 character	Event tree name
name	24 character	Sequence name
description	120 character	Sequence description
A	1 character	If included indicates alternate description

Sequence Cut sets

The sequence cut sets are the minimal cut sets for sequence logic as derived from the fault tree logic. The cut sets are stored in the block data file of the Sequence relation.

The MAR D sequence cut sets (.SQC) are in a format similar to that of the fault tree cut sets.

File Name:
xxxxxx.SQC
File Format:
project, event tree, sequence, analysis =
eventname * eventname +hjn
eventname * eventname * eventname *
eventname +
eventname * eventname.
^EOS
project, event tree2, sequence2 =
where

project	24 character	Project name	
event tree	24 character	Event tree name	
sequence	24 character	Sequence name	
analysis	1 character	Analysis type	
1			Random
2			Fire
3			Flood
4			Seismic
5 through 8			Reserved
9 through 16			user-defined
eventname	24 character	Event names in the cut set	

General Rules:

1. An asterisk (*) separates events in a cut set. Spaces are ignored.

2. A plus sign (+) separates cut sets.

3. A period (.) denotes the end of the sequence.

4. A slash (/) precedes complemented events.

5. Event names have a maximum of 24 characters including the "/" character for complemented events.

6. A line beginning with an asterisk (*) is a comment.

Sequence Attributes

File Name:

xxxxxx.SQA

File Format:

project, event tree, analysis =
name,endstate,mincut,mission,procut,sample,seed,size,cuts,
events,value1, . . . ,value9,default flags, used flags
. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . .
^EOS
project, event tree2 =

where

project	24 character	Project name
event tree	24 character	Event tree name
analysis	1 character	Analysis type

1	Random
2	Fire
3	Flood
4	Seismic
5 through 8	Reserved
9 through 16	user-defined

name	24 character	Sequence name
endstate	24 character	End State name
mincut	Floating point	Mincut upper bound
mission	Floating point	Mission time in hours
procut	Floating point	Probability cut off value
sample	Integer 4	Sample size
seed	Integer 8	Random number seed
size	Integer 2	Size cut off value
cuts	Integer 5	Base number of cut sets
events	Integer 5	Base number of events
value	Floating point	Base uncertainty values

value1	5^{th} percentile
value2	Median
value3	Mean
value4	95th percentile
value5	Minimum sample
value6	Maximum sample
value7	Standard deviation
value8	Skewness
value9	Kurtosis

Default flags	24 character	Default flag set for this sequence
Used flags	24 character	Flag set used to generate these cut sets

Sequence Logic

File Name:
xxxxxxxx.SQL
File Format:
project, event tree, sequence=
sys1 sys2 /sys3 sys4
. . .
^EOS
project, event tree2, sequence2=
where

Project	24 character	Project name
event tree	24 character	Event tree name
Sequence	24 character	Sequence name
Sys	24 character	Fault tree name

Sequence Textual Information (Version 6)

File Name:
xxxxxx.SQT

File Format:
project, event tree, sequence=
--- text ---
^EOS
project, event tree2, sequence2=
--- text ---

where

project	24 character	Project name
sequence	24 character	Sequence name
event tree	24 character	Event tree name
A	1 character	If included indicates alternate description

Sequence Textual Information (Version 7)

File Name:
xxxxxx.SQT
File Format:
project, event tree, sequence[, A]=
--- text ---
^EOS
project, event tree2, sequence2=
--- text ---
where

project	24 character	Project name
sequence	24 character	Sequence name
event tree	24 character	Event tree name
A	1 character	If included indicates alternate description

Sequence Recovery Rules

File Name:
xxxxxxxx.SQY
File Format:
project, event tree, sequence =
-- recovery rule text --
^EOS
project, event tree, sequence2 =
where

project	24 character	Project name
event tree	24 character	Event tree name
sequence	24 character	Sequence name

Sequence Partition Rules

File Name:
xxxxxxxx.SQP
File Format:
project, event tree, sequence =
-- partition rule text --
^EOS
project, event tree, sequence2 =
where

Project	24 character	Project name
event tree	24 character	Event tree name
Sequence	24 character	Sequence name

Gates

Gate Description (Version 6)

File Name:

xxxxxx.GTD

File Format:

project=
name,description

where

Project	24 character	Project name
Name	24 character	Gate name
description	120 character	Gate description

Gate Description (Version 7)

File Name:

xxxxxx.GTD

File Format:

project=
name,description[,A]

where

Project	24 character	Project name
Name	24 character	Gate name
description	120 character	Gate description
A	1 character	If included indicates alternate description

Gate Attributes

File Name:

xxxxxx.GTA

File Format

project=
name,attribute

where

Project	24 character	Project name
Name	24 character	Gate name
Attribute	4 character	Gate type

Change Sets

Change Set Description (Version 6)

File Name:

xxxxxx.CSD

File Format:

project=
name,description
...,...

where

project	24 character	Project name
name	24 character	Change set name
description	60 character	Change set description

Change Set Description (Version 7)

File Name:

xxxxxx.CSD

File Format:

project=
name,description[,A]
...,...

where

project	24 character	Project name
name	24 character	Change set name
description	120 character	Change set description
A	1 character	If included indicates alternate description

Change Set Information (Version 6)

File Name:
xxxxxx.CSI

File Format:
project,change=
^PROBABILITY
eventname,calc,udT,prob,lambda,tau,udV,udC,mission,init
^CLASS
eventname,group,compType,compId,system,location,failMode,train,init,att1,..att24
calcType,udT,prob,lambda,tau,udV,udC,mission,init
^EOS
project,change2=

where

change	24 character	change set name
eventname	24 character	name mask
group	24 characters	event group mask
compType	7 characters	component type mask
compId	3 characters	component ID mask
system	3 characters	system mask
location	3 characters	location mask
failMode	2 characters	failure mode mask
train	2 characters	train mask
init	1 character	initiating event (Y/N)
att1..att16	Class attribute flags	16 values of Y or N (yes or no) indicate whether the attribute described in the class attribute file is applicable.
calc	1 character	Calculation type

1	Probability
2	same as type 3
3	1 Exp(-Lambda * Mission Time)
4	same as type 5
5	Operating component with full repair
6	same as type 7
7	1+(EXP(Lambda*Tau) 1.0)/(Lambda*Tau)
8	Base Probability * Probability
9	Base Probability * Probability
T	Set to House Event (Failed, Prob=1.0)
F	Set to House Event (Successful,Prob=0.0)
I	Set to ignore
S	Use fault tree min cut upper bound
E	Use end state min cut upper bound

	G	Seismic event - Enter g level for screening
	H	Seismic event - Use medium site hazard curve for screening

udT	1 character	Uncertainty distribution type

P	use point estimate
L	Log normal, error factor
N	Normal, standard deviation
B	Beta, b of Beta(a,b)
D	Dirichlet, b of Dirichlet
G	Gamma, a Gamma(a)
C	Chi-squared, degrees of freedom
E	Exponential, none
U	Uniform, Upper end pt.
H	Histogram
M	Maximum entropy
S	Seismic log normal, betaR, betaU
O	Constrained non-informative

prob	Floating point	Probability value
lambda	Floating point	Basic event failure rate per hr.
tau	Floating point	Time to repair in hours
udV	Floating point	Uncertainty distribution value
udC	4 characters	Uncertainty correlation class. Events in same class are 100% correlated.
mission	Floating point	Mission time
init	Boolean (T/F)	Initiating event

Change Set Information (Version 7)

File Name:
xxxxxx.CSI
File Format:
project,change=
^PROBABILITY
eventname,calc,udT,prob,lambda,tau,udV,udC,mission,init
^CLASS
eventname,group,compType,compId,system,location,failMode,train,init,att1,..att16
calcType,udT,prob,lambda,tau,udV,udC,mission,init
^EOS
project,change2=
where

change	24 character	change set name
eventname	24 character	name mask
group	24 characters	event group mask
compType	7 characters	component type mask
compId	3 characters	component ID mask
system	3 characters	system mask
location	3 characters	location mask
failMode	2 characters	failure mode mask
train	2 characters	train mask
init	1 character	initiating event (Y/N)
att1..att16	Class attribute flags	16 values of Y or N (yes or no) indicate whether the attribute described in the class attribute file is applicable.
calc	1 character	Calculation type

1	Probability
3	1 Exp(-Lambda * Mission Time)
5	Operating component with full repair
7	1+(EXP(Lambda*Tau) 1.0)/(Lambda*Tau)
8	Base Probability * Probability
9	Base Probability * Probability
T	Set to House Event (Failed, Prob=1.0)
F	Set to House Event (Successful,Prob=0.0)
I	Set to ignore
S	Use fault tree min cut upper bound
E	Use end state min cut upper bound
G	Seismic event - Enter g level for screening
H	Use medium site hazard curve
B	Use base case (even if prior marked change sets have altered the value)

udT	1 character	Uncertainty distribution type

P	Use point estimate
L	Log normal, error factor
N	Normal, standard deviation
B	Beta, b of Beta(a,b)
D	Dirichlet, b of Dirichlet(a,b)
G	Gamma, a of Gamma(a)
C	Chi-squared, degrees of freedom
E	Exponential, none
U	Uniform, Upper end pt.
H	Histogram
M	Maximum entropy
S	Seismic log normal, betaR, betaU
O	Constrained non-informative

prob	Floating point	Probability value
lambda	Floating point	Basic event failure rate per hr.
tau	Floating point	Time to repair in hours
udV	Floating point	Uncertainty distribution value
udC	24 characters	Uncertainty correlation class. Events in same class are 100% correlated.
mission	Floating point	Mission time
init	Boolean (T/F)	Initiating event

Change Set Attributes (Version 7 only)

File Name:
xxxxxx.CSA
File Format:
project=
name,altName
...,...
where

project	24 character	Project name
name	24 character	Change set primary name
altName	24 character	Change set alternate name

Histograms

Histogram Description (Version 6)

File Name:

xxxxxxxx.HID

File Format:

project =
name,type,subtype,description

where

project	24 character	Project name	
name	24 character	Histogram primary name	
type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range
H			Hazard
Description	60 character	Histogram description	

Histogram Description (Version 7)

File Name:

xxxxxxxx.HID

File Format:

project =
name, type, subtype, description[, A]

where

project	24 character	Project name	
name	24 character	Histogram primary name	
type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range

H			Hazard
Description	120 character	Histogram description	
A	1 character	If included indicates alternate description	

Histogram Information

File Name:
xxxxxxxx.Hll
File Format:
project, name1=
type, subtype
bin1 value1, bin1 value2
bin2 value1, bin2 value2
...
bin20 value1, bin20 value2
^EOS
project, name2 =
where

Project	24 character	Project name	
NameN	24 character	Histogram primary name	
Type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
Subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range
H			Hazard
bin value1	Exponential	first value for bin	
bin value2	Exponential	second value for bin	

Histogram Attributes (Version 7 only)

File Name:
xxxxxxxx.HII
File Format:
project =
name, type, subtype, altName
where

project	24 character	Project name	
name	24 character	Histogram primary name	
type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range
H			Hazard
altName	24 character	Histogram alternate name	

Slices

Slice Descriptions (Version 6)

File Name:
xxxxxxxx.SLD
File Format:
project =
name, description
where

project	24 character	Project name
name	24 character	Slice name
description	60 character	Slice description

Slice Descriptions (Version 7)

File Name:
xxxxxxxx.SLD
File Format:
project =
name, description[, A]
where

project	24 character	Project name

name	24 character	Slice name
description	120 character	Slice description
A	1 character	If included indicates alternate description

Slice Basic Events

File Name:
xxxxxxxx.SLB
File Format:
project, slice =
eventname + eventname + eventname + .
^EOS
project, slice2 =
where

project	24 character	Project name
slice	24 character	Slice name
eventname	24 character	Event names in the slice
+ or *	1 character	Slice logic: +=or , *= and

General Rules:

1. A plus symbol (+) or asterisk (*) between event names represent the logic in a slice. Spaces are ignored. All logic must be the same in a slice.

2. A period (.) denotes the end of the slice.

3. A slash (/) precedes complemented events.

4. Event names have a maximum of 24 characters including the "/" character for complemented events.

5. A line beginning with an asterisk (*) is a comment.

Slice Basic Information

File Name:
xxxxxxxx.SLI
File Format:
project, slice =
eventname , delta, factor
...,...
^EOS
project, slice2 =
where

project	24 character	Project name
slice	24 character	Slice name
eventname	24 character	Event names in the slice

delta	Floating point	Delta value that is factored
factor	1 character	Factor flag: F=multiply, Blank=add

Slice Basic Attributes (Version 7 only)

File Name:
xxxxxx.SLA
File Format:
project=
name,altName
...,...
where

project	24 character	Project name
name	24 character	Slice primary name
altName	24 character	Slice alternate name

SETS FORMAT

Sequences (SETS)

Sequence Cut sets

File Name:
xxxxxx.DNF.

The format of the SETS output cut sets file (.DNF) is dependent upon the command issued within SETS. The factored form is

A * (B + C)

The disjunctive normal form is

A * B + A * C.

ONLY the disjunctive normal form is accepted by SAPHIRE at this time.

File Format:
sequence-name =
eventName * eventName +
eventName * eventName.
where
. . . . .

General Rules:

1. An asterisk (*) separates event names. Spaces are ignored.
2. A plus sign (+) separates cut sets.
3. A period (.) denotes the end of a sequence.
4. An asterisk (*) in the first column denotes a comment.

Fault Trees (SETS)

Fault Tree Logic

File Name:
xxxxxx.SET.
File Format:
FAULT TREE$ fault tree name.
COMMENT$ descriptive material $
gate type $ gate name. IN$ input 1, input 2, . . . , input n.
OUT$ output 1, output 2, . . . , output n.
event type $event name. OUT$ output 1, . . . , output n.
where

fault tree name	The name of the fault tree.
gate type	The type of gate being defined.
AG	= AND gate
OG	= OR gate
EOR	= Exclusive OR gate (converted to SG)
EAG	= Exclusive AND gate (converted to SG)
SG	= Special Gate
gate name	The name of the gate being defined (16 characters) input n
	The names of the gates or primary events that are the immediate inputs to the gate being defined (16 characters)
output n	The names of the gates that are the immediate outputs of the gate or primary event being defined (16 characters).
event type	The type of primary event being defined.
BE	= Basic Event
CE	= Conditional Event
UE	= Undeveloped Event
DE	= Developed Event
EE	= External Event
COMMENT$	Defines a comment. Must follow a "." delimiter.

Fault Tree Cut sets

The fault tree cutsets are stored in the System relation in the block data file. The format of the cutset file (.DNF) is given above.

Basic Events (SETS)

Basic Event Descriptions

File Name:
xxxxxxx.DES.
File Format:
name $ description $
name $ description $
where

name	event name
name list	description of event

Basic Event Failure Rates

File Name:
xxxxxxx.VBK.
File Format:

VALUE BLOCK$ value-block-name

prob $ name-list$

prob $ name-list$

where

prob	point value probability estimate
name list	list of event names separated by commas

Appendix C

MAR-D Files for Sample Database

C. MAR-D Files for Sample Database

SAPHIRE Version 6 MAR-D formats for the Sample Database are presented. Version 6 results were selected for presentation since they can be loaded into both versions 6 and version 7.

Note that these examples are shown in a document created by a word processor. Actual MAR-D files should be edited in a text editor, such as Notepad, so that formatting codes are not embedded into the text. SAPHIRE handles only ASCII text characters.

In this document, some line wrapping occurs so that entire lines can be displayed. Where this occurs in this document, the wrapped line will appear indented.

PROJECT FILES

These are examples of files (or partial files) in MAR-D formats for the Sample database. These formats are as of August 2005.

Project Names and Description File (.FAD)

```
SAMPLE          ,This is a sample data base
```

Project Attribute File (.FAA)

```
SAMPLE                  , 0001 =
*  Name          , Mission   , NewSum    , Company  , Location       ,Typ,
      Design   ,Vendr, Arch Eng , OpDate   , QualDate
SAMPLE                  , 2.400E+001,+0.000E+000,STANDARD  ,HOMETOWN        ,
      ,          ,     ,          ,----/--/--,----/--/--
```

Project Text File (.FAT)

```
SAMPLE           =
      A simple example that models the probability of getting to work on time.
SAMPLE           =
 A simple example that models the probability of getting to work on time.
```

BASIC EVENT FILES

Basic Event Names and Description File (.BED)

```
SAMPLE                =
ALARM                ,ALARM CLOCK FAILURE
ALM-BPF              ,Alarm fails due to battery failure
ALM-CPF              ,Alarm fails due to commercial power failure
ALM-FTS              ,Alarm fails because worker fails to set
ALM-MECH             ,Alarm fails due to mechanical failure
ALM-SWT              ,Alarm fails because worker set wrong time
MEDICINE             ,Recovery for sick failure preventing attending work
OTHER                ,Other personal reasons that cause a failure to get to work
PER-TRNS             ,Personal transportation
PERSONAL             ,PERSONAL PROBLEMS
PUB-TRNS             ,Public transportation fails
PUB-TRNS-LATE        ,Public transportation fails late time frame
SICK                 ,Failed to get to work because of illness
SICK-FAM             ,Failed to get to work because of illness in project
TRNS-2               ,COMMERCIAL TRANSPORTATION FAILS AT A LATER TIME
TRNSPRT              ,PERSONAL AND COMMERCIAL TRANSPORTATION FAIL
WORK                 ,Event tree (WORK) initiating event
```

Basic Event Rate Information File (.BEI)

```
SAMPLE                    =
*  Name          ,FdT,UdC,UdT, UdValue , Prob       , Lambda     , Tau        ,
      Mission ,Cat,PF, UdValue2
ALARM            ,1,     ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
ALM-BPF          ,1,     ,L, 3.000E+000, 9.000E-008,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
ALM-CPF          ,1,     ,L, 3.000E+000, 1.500E-002,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
ALM-FTS          ,1,     ,L, 1.000E+001, 5.500E-006,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
ALM-MECH         ,1,     ,L, 3.000E+000, 2.700E-008,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
ALM-SWT          ,1,     ,L, 1.000E+001, 2.700E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
MEDICINE         ,1,     ,L, 5.000E+000, 5.000E-001,+0.000E+000,
      +0.000E+000,+0.000E+000,R, ,+0.000E+000
OTHER            ,1,     ,L, 1.000E+001, 8.100E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
PER-TRNS         ,1,     ,L, 5.000E+000, 5.500E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
PERSONAL         ,1,     ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
PUB-TRNS         ,1,     ,L, 3.000E+000, 2.700E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
PUB-TRNS-LATE ,1,     ,L, 3.000E+000, 2.000E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
SICK             ,1,     ,L, 1.000E+001, 8.100E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
SICK-FAM         ,1,     ,L, 1.000E+001, 4.000E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
TRNS-2           ,1,     ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
```

```
TRNSPRT       ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000
WORK          ,1,    ,L, 2.000E+000, 2.480E+002,+0.000E+000,
      +0.000E+000,+0.000E+000,I, ,+0.000E+000
```

Basic Event Attribute File (.BEA)

```
SAMPLE         =
*  Name        ,AltName                ,Typ,Sys,Fail,Loc,CompId,Train,
      Attributes                      ,Template Name           ,Use
      TemplateFlags                                                ,Shape
ALARM         ,ALARM                    ,DE ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
ALM-BPF       ,ALM-BPF                  ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
ALM-CPF       ,ALM-CPF                  ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
ALM-FTS       ,ALM-FTS                  ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
ALM-MECH      ,ALM-MECH                 ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
ALM-SWT       ,ALM-SWT                  ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
MEDICINE      ,MEDICINE                 ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
OTHER         ,OTHER                    ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
PER-TRNS      ,PER-TRNS                 ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,B
PERSONAL      ,PERSONAL                 ,DE ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
PUB-TRNS      ,PUB-TRNS                 ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
PUB-TRNS-LATE ,PUB-TRNS-LATE            ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N, ,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,B
SICK          ,SICK                     ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
SICK-FAM      ,SICK-FAM                 ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
TRNS-2        ,TRNS-2                   ,DE ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
TRNSPRT       ,TRNSPRT                  ,DE ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
```

```
WORK          ,WORK                      ,   ,   ,   ,   ,       ,
      ,Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,
      ,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,
```

FAULT TREE FILES

Fault Tree Names and Description File (.FTD)

```
SAMPLE                    =
ALARM        ,ALARM CLOCK FAILURE
PERSONAL     ,PERSONAL PROBLEMS
TRNS-2       ,COMMERCIAL TRANSPORTATION FAILS AT A LATER TIME
TRNSPRT      ,PERSONAL AND COMMERCIAL TRANSPORTATION FAIL
```

Fault Tree Logic File (.FTL)

```
SAMPLE, ALARM =
ALARM                     OR   ALARM-1 ALARM-2 ALM-MECH
ALARM-1                   OR   ALM-FTS ALM-SWT
ALARM-2                   AND  ALM-BPF ALM-CPF
^EOS
SAMPLE, PERSONAL =
PERSONAL                  OR   OTHER SICK SICK-FAM
^EOS
SAMPLE, TRNS-2 =
TRNS-2                    AND  PER-TRNS PUB-TRNS-LATE
^EOS
SAMPLE, TRNSPRT =
TRNSPRT                   AND  PER-TRNS PUB-TRNS
```

Fault Tree Graphics File (.DLS)

```
NOT IN ASCII FORMAT
```

Fault Tree Cut Sets File (.FTC)

```
SAMPLE, ALARM, 0001=
ALM-BPF * ALM-CPF +
ALM-FTS +
ALM-MECH +
ALM-SWT .
^EOS
SAMPLE, PERSONAL, 0001=
OTHER +
SICK +
SICK-FAM .
^EOS
SAMPLE, TRNS-2, 0001=
PER-TRNS * PUB-TRNS-LATE .
^EOS
SAMPLE, TRNSPRT, 0001=
PER-TRNS * PUB-TRNS .
```

Fault Tree Attribute File (.FTA)

```
SAMPLE, 0001 =
* Name             , Level, Mission   , MinCut    , Def ProCut,Used
      ProCut,Sample,Seed,Siz,Sys,Cuts,Events, UdValues,
      Def Flags,              Used Flags,      S QMethod, S QPasses, R
      QMethod, R QPasses
ALARM      ,0, 2.400E+001, 2.706E-003,------E----,------E----, -----,-----,--,
      ,------,-----,------E----,------E----,------E----,------E----,------E---
      -,------E----,------E----,------E----,------E----,
      ,                        , ,----,M,   0
PERSONAL   ,0, 2.400E+001, 2.007E-002,------E----,------E----, -----,-----,--,
      ,------,-----,------E----,------E----,------E----,------E----,------E---
      -,------E----,------E----,------E----,------E----,
      ,                        , ,----,M,   0
TRNS-2     ,0, 2.400E+001, 1.100E-005,------E----,------E----, -----,-----,--,
      ,------,-----,------E----,------E----,------E----,------E----,------E---
      -,------E----,------E----,------E----,------E----,
      ,                        , ,----,M,   0
TRNSPRT    ,0, 2.400E+001, 1.485E-005,------E----,------E----, -----,-----,--,
      ,------,-----,------E----,------E----,------E----,------E----,------E---
      -,------E----,------E----,------E----,------E----,
      ,                        , ,----,M,   0
```

Fault Tree Text File (.FTT)

```
SAMPLE, ALARM=
The ALARM fault tree is a simple representation modeling alarm clock failure.
Some common reasons for alarm clock failure include setting the wrong time,
mechanical failure, or power failure (either battery or commercial).
```

EVENT TREE FILES

Event Tree Names and Descriptions File (.ETD)

```
SAMPLE           =
WORK            ,WORK EVENT TREE
```

Event Tree Graphics File (.ETG)

```
SAMPLE, WORK, WORK =
^WINVER1.0
^TOPS
ALARM, PERSONAL, TRNSPRT
^LOGIC
 +1 +2 +3
       -3
    -2  3
 -1  2 +3
       -3
^SEQUENCES
Y, SEQUENCE, N, Y, N, STATE, N, N,
N, N, N, OK, Y, A, Y, OK,
Y, B, Y, LATE-TO-WORK, Y, , Y, ,
Y, C, Y, MISS-WORK, Y, , Y, ,
Y, D, Y, LATE-TO-WORK, Y, , Y, ,
Y, E, Y, LATE-TO-WORK, Y, , Y, ,
^TOPDESC
 "INITIATING EVENT"
!
 "ALARM FAILURE"
!
 "PERSONAL FAILURE"
!
 "TRANSPORTATION",
 "FAILURE"
!
^PARMS
START 52.00, 809.60 WINDOW 132.00, 363.50, 1043.00, 1274.50
ASPECTRATIO 0.74
HEADER 972.00, 1224.00, 1476.00, 1728.00
STRING E
DEFFONT 5
TOPWIDTH 16
TOPSIZE -15.00
TOPFONT 1
TOPFACE Times_New_Roman
TOPPITCH 2
TOPCOLOR 15
DESHITE 3
DESSIZE -10.00
DESFONT 5
DESFACE Times_New_Roman
DESCOLOR 15
DESPITCH 2
NODEHITE 20.00
ENDSIZE -15.00
ENDFONT 1
ENDFACE Times_New_Roman
ENDPITCH 2
ENDCOLOR 15
```

```
BACKCOLOR 1
TOPBACKCOLOR 1
LINECOLOR 15
HILITECOLOR 1
LOCALE 1033
MODDATE 2003/09/23
```

Event Tree Logic File (.ETL)

```
SAME AS THE .ETG FILE SECTION C.5.2
```

Event Tree Attribute File (.ETA)

```
SAMPLE           =
*  Name         , Init Event
WORK            , WORK
```

Event Tree Rules File (.ETR)

```
SAMPLE, WORK=
| rule to substitute TRNS-2 for TRNSPRT
if ALARM then
  TRNSPRT = TRNS-2;
endif
```

Event Tree Recovery Rules (.ETY)

```
SAMPLE, WORK=
| rule to add recovery potential to the cut sets
if SICK then
  recovery = MEDICINE;
endif
```

Event Tree Text File (.ETT)

```
SAMPLE, WORK=
A FAIL-SUCCESS LOGIC WAS USED TO DEVELOP AN EVENT TREE TO CALCULATE THE
FREQUENCY THAT THE AVERAGE PERSON WILL ARRIVE ON TIME, BE LATE, OR MISS A DAY
OF WORK.
```

END STATE FILES

End State Names and Description File (.ESD)

```
SAMPLE           =
LATE-TO-WORK      , This end state represents being late to work
MISS-WORK         , This end state represents missing work
```

End State Text File (.EST)

```
SAMPLE, LATE-TO-WORK=
THIS IS THE LATE TO WORK END STATE.
```

SEQUENCE FILES

Sequence Names and Description File (.SQD)

```
SAMPLE, WORK=
2               ,LATE TO WORK
3               ,MISS WORK
4               ,LATE TO WORK
5               ,LATE TO WORK
```

Sequence Cut Set File (.SQC)

```
SAMPLE, WORK, 2, 0001=
PER-TRNS * PUB-TRNS .
^EOS
SAMPLE, WORK, 3, 0001=
OTHER +
SICK * MEDICINE +
SICK-FAM .
^EOS
SAMPLE, WORK, 4, 0001=
ALM-BPF * ALM-CPF +
ALM-FTS +
ALM-MECH +
ALM-SWT .
^EOS
SAMPLE, WORK, 5, 0001=
ALM-BPF * ALM-CPF * PER-TRNS * PUB-TRNS-LATE +
ALM-FTS * PER-TRNS * PUB-TRNS-LATE +
ALM-MECH * PER-TRNS * PUB-TRNS-LATE +
ALM-SWT * PER-TRNS * PUB-TRNS-LATE .
```

Sequence Cut Set Attribute File (.SQA)

```
SAMPLE, WORK, 0001=
* Name         , End State              , MinCut    , Mission   , ProCut
      ,Sample,Seed,Siz,Cuts,Events, UdValues, Def Flags, Used FlagsS QMethod,
      S QPasses, R QMethod, R QPasses
2    ,LATE-TO-WORK    , 3.683E-003, 2.400E+001,------E----, 1000,40777,--,
      1,    3,------E----,------E----,------E----,------E----, ------E----,---
      ---E----,------E----,------E----,------E----,     ,     , ,----,M,   0
3    ,MISS-WORK       , 3.985E+000, 2.400E+001,------E----, 1000,46267,--,
      3,    5,------E----,------E----,------E----,------E----, ------E----,---
      ---E----,------E----,------E----,------E----,     ,     , ,----,M,   0
4    ,LATE-TO-WORK    , 6.710E-001, 2.400E+001,------E----, 1000,52257,--,
      4,    6,------E----,------E----,------E----,------E----, ------E----,---
      ---E----,------E----,------E----,------E----,     ,     , ,----,M,   0
5    ,LATE-TO-WORK    , 7.381E-006, 2.400E+001,------E----, 1000,58407,--,
      4,    8,------E----,------E----,------E----,------E----, ------E----,---
      ---E----,------E----,------E----,------E----,     ,     , ,----,M,   0
```

Sequence Logic File (.SQL)

```
SAMPLE, WORK, 2=
/ALARM /PERSONAL TRNSPRT .
^EOS
SAMPLE, WORK, 3=
/ALARM PERSONAL .
^EOS
SAMPLE, WORK, 4=
ALARM /TRNSPRT .
^EOS
SAMPLE, WORK, 5=
ALARM TRNS-2 .
```

Sequence Text File (.SQT)

```
SAMPLE, WORK, 3=

Sequence 3 is the event tree sequence that is used to demonstrate the use of
recovery rules or recovery actions.
```

GATE FILES

Gate Description File (.GTD)

```
SAMPLE           =
ALARM          , ALARM CLOCK FAILURE
ALARM-1        , ALARM CLOCK SETTING FAILURE
ALARM-2        , ALARM CLOCK POWER FAILURE
PERSONAL       , PERSONAL PROBLEMS
TRNS-2         , COMMERCIAL TRANSPORTATION FAILS AT A LATER TIME
TRNSPRT        , PERSONAL AND COMMERCIAL TRANSPORTATION FAILURE
```

Gate Attributes File (.GTA)

```
SAMPLE                    =
*  Name           , Type
ALARM                     , OR
ALARM-1                   , OR
ALARM-2                   , AND
PERSONAL                  , OR
TRNS-2                    , AND
TRNSPRT                   , AND
```

Appendix D

Seismic Data Loading

D. Seismic Data Loading

INTRODUCTION

This appendix discusses the features and basic data loading processes of the seismic module in SAPHIRE 5.0. The seismic data loading process assumes the availability of internal-events PRA or database (i.e. a SAPHIRE data base implementing analysis with random failures within a particular system). The procedures necessary for seismic data loading using the SAPHIRE code are described in the following subsections.

SAPHIRE SEISMIC CAPABILITIES

The SAPHIRE seismic analysis capabilities are designed to function directly from the internal-events PRA. Thus, internal basic events, system fault tree models, accident sequences, and initiating events have all been defined and developed for the system of interest. The SAPHIRE seismic analysis consists of taking the internal basic events (having random failures) and converting them into seismic basic events that represent seismic-induced failures. SAPHIRE performs transformations in the form of Boolean identities that allows the user to build on an internal-events analysis when developing a seismic model. After seismic vulnerabilities have been identified, they are incorporated into an existing internal-events analysis using a set of basic event transformations that substitute in seismic-induced failures that are used to generate seismic sequence or system cut sets.

BUILDING AND LOADING THE SEISMIC SAPHIRE MODEL

Hazard Curves

The hazard curve represents a range of possible earthquake magnitudes. The curve is usually found in the form of a probability of exceedence curve, with the earthquake ground acceleration on the horizontal axis and the probability of exceeding that acceleration on the vertical axis. (Sources of hazard curve data and information include NUREG-1488 and NUREG-4550.) SAPHIRE uses this information in the form of a histogram or a discreet probability density distribution. For a more detailed description of hazard curves and the methodology on their use during seismic analysis, see the SAPHIRE Technical Reference Manual.

The hazard curve (or histogram) that will be used in the seismic analysis is developed or modified by selecting the desired seismic hazard curve in the SAPHIRE program. This is done by selecting Modify → Project main menu option. Under the heading "Site Hazard Curves", there are three fields: "Low", "Medium", and "High". The histogram listed in the "Medium" field will be the one used during analysis. If a seismic hazard curve is not available, then one must be added in order to generate quantified cut sets. A seismic hazard curve (or histogram) can be added (or loaded) into the SAPHIRE database using two methods. The histogram can be added and the discrete data points input from the Modify→ Histograms main menu option or it can be loaded from a histogram flat file (.HII) through the Utility → Load and Extract main menu option. The procedures for both methods are discussed below.

Loading the Seismic Histogram through the Modify main menu option

To add a seismic histogram, the following steps are required:

1. Select Modify → Histogram main menu option.

2. Right click to invoke a popup menu, and from it select Add.

3. Choose the Hazard histogram format.

4. Enter the name and description of the seismic histogram.

5. Enter the acceleration rates and frequencies. The acceleration rate is the peak ground acceleration (i.e., magnitude of the earthquake). The frequency is the probability that an earthquake that exceeds the ground acceleration will occur.

6. Press the OK button to save the new histogram.

Next, assign the histogram to the project's site hazard curve:

1. Select Modify → Project main menu option to bring up the Edit Project dialog.

2. Under the heading "Site Hazard Curves", type in the name of the seismic histogram for the "Medium" field.

Loading the Seismic Histogram Through the MAR-D Interface

The hazard curve (or histogram) may also be loaded into the SAPHIRE database using the Utility → Load and Extract main menu option (also known as the MAR-D interface). The histogram is represented in an ASCII text file and loaded into the SAPHIRE database as discussed in Appendix A. The two flat file types that are required to load the histogram using MAR-D are discussed below.

Histogram Description File (.HID)

The MAR-D flat file format for the SAPHIRE version 6 histogram description file (.HID) is shown below. (The version 7 format is the same, but can accommodate up to 24 character names, and 120 character descriptions.)

File Name:

xxxxxxxx.HID

File Format:

project =

name, type, subtype, description[, A]

where

project	16 character	Project name	
name	16 character	Histogram primary name	
type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range
H			Hazard
Description	60 character	Histogram description	
A	1 character	If included indicates alternate description	

An example of a histogram description file in MAR-D format is as follows:

```
SAMPLE          =
SEISMIC         , H, H, Histogram for Seismic Analysis
```

Histogram Information File (.HII)

The MAR-D data format for the SAPHIRE version 6 histogram information file (.HII) is shown below. (The version 7 format is the same, but can accommodate up to 24 character names.)

File Name:

xxxxxxxx.HII

File Format:

project, name1=
type, subtype
bin1 value1, bin1 value2
bin2 value1, bin2 value2
...
bin20 value1, bin20 value2
^EOS
project, name2 =

where

Project	16 character	Project name	
NameN	16 character	Histogram primary name	
Type	1 character	Histogram type	
H			Hazard
U			Uncertainty
F			Fragility
Subtype	1 character	Histogram subtype	
P			Percent
A			Area
R			Range
H			Hazard
bin value1	Exponential	first value for bin	
bin value2	Exponential	second value for bin	

An example of a histogram information file in MAR-D format is shown below. For this example, the flat file will load seven bins with seismic hazard histogram data. For all .HII files containing seismic data, "bin1 value1" or column 1 is the earthquake frequency (per yr) and "bin1 value2" or column 2 is the mean failure acceleration of the earthquake.

SAMPLE, SEISMIC =
H, H
3.680E-003, 1.000E-001
2.980E-004, 2.000E-001
7.200E-005, 3.000E-001
2.620E-005, 4.000E-001
1.170E-005, 5.000E-001
6.000E-006, 6.000E-001
3.360E-006, 7.000E-001

Event Trees

The creation of a seismic analysis model in SAPHIRE requires the development of a seismic event tree. The seismic event tree can be designed to incorporate the seismic analysis by two methods. The first method utilizes the internal basic events and fault trees assumed already present in the database. This method prioritizes and links the seismic-induced internal events and fault trees and will generate seismic sequence cut sets from the internal basic events. The second method utilizes separated seismic fault tree logic that may incorporate internal events or separate seismic events to generate the seismic cut sets. For both

methods, the seismic event tree begins with a generic seismic-initiating event set to a value of 1.0 (True Event). The actual magnitude and frequency of the earthquake of interest are identified by the user and factored into the analysis when the cut sets are generated and quantified.

The top events for the seismic event tree are those events or systems that have the potential to be induced by an earthquake. They are listed in order of severity, with the more severe-induced initiators listed first. This also addresses the potential pitfall of over-counting core damage sequences where, for example, a single earthquake induces both a large LOCA and a small LOCA at the same time. During the seismic analysis, the event tree top events are treated as seismic events with the associated seismic fragility data.

The procedure for loading or adding event trees to SAPHIRE database was discussed in Section 4.3. Identical procedures are required for the loading of the seismic event tree and any sub trees.

Fault Trees

The seismic system models (i.e., fault trees) can be created in SAPHIRE either as independent, stand-alone seismic fault trees, or they can also be integrated with the internal events analysis. To integrate seismic analysis into the internal events analysis, transformations need to be defined that convert random failures to seismic-induced failures.

Because the internal fault trees do not include several seismic related basic events, they must be added to the internal fault trees or independent seismic fault trees must be created. The procedures for loading or adding system fault trees were discussed in Section 4.5.

Basic Event Data

In most instances, seismic basic events are transformed internal basic events where the seismic considerations are implemented after the transformations. Seismic failure data are usually characterized by a median fragility and two uncertainty terms representing the random uncertainty and confidence uncertainty (Beta-R and Beta-U, respectively). See the SAPHIRE Technical Reference Manual for a more in depth discussion of seismic fragility and component failure probabilities.

The necessary steps in loading seismic basic events into the SAPHIRE program are:

1. Add the seismic event to the database including any basic event attribute data.

2. Enter the seismic failure acceleration data.

3. Enter the seismic uncertainty data.

4. Modify any internal basic events that are determined to have seismic vulnerabilities to include a seismic susceptibility "flag". This will allow for the internal basic event to be transformed into a new seismic event.

5. Enter the transformation definition to the internal basic event that is seismic susceptible.

These steps are further discussed in the following sections.

Adding Seismic Basic Events

Before the internal basic event transformation can be created, the seismic basic events must be defined. In most cases, the newly created seismic event has a different name than the internal basic event name that it is transformed from originally. For example, if the internal basic event HPI-MOV-FO-108A is determined to be seismic susceptible, then it must be transformed into a seismic event. The new seismic event could be named S-HPI-MOV-FO-108A and must be added to the database.

The procedure for adding seismic basic events and their descriptions is identical to that of internal basic events and is discussed in Section 4.6.

Loading the Seismic Failure Acceleration Data

Loading of the seismic failure data is similar to the procedures discussed for loading failure data discussed in Section 4.6. Two methods can be used to load seismic failure acceleration data. The data can be entered in the Modify → Basic Event main menu option or from basic event flat file (.BEI) and loaded through the Utility → Load and Extract main menu option as described in Appendix A. Differences between loading seismic data and the procedures discussed in Section 4.6 are outlined below.

Loading Through the Modify → Basic Event main menu option.

To enter seismic data into a seismic basic event record, go to the "Failure Data Calculation Type". Enter a "G" or an "H", which defines the basic event as a seismic basic event. Entering a "G" allows you to input an assumed g-level (earthquake strength) for use in initially generating cut sets. The "H" tells SAPHIRE to use the hazard curve identified in the “Medium” hazard curve in the Modify → Project option.

Loading Through the MAR-D Interface.

The loading of seismic failure data through the MAR-D interface is similar to the procedures described in Section 4.6 for load internal basic event failure rates. The seismic basic event flat file (.BEI) data format is similar to that in Appendix B except for the following:

1. Set the calculation type (calc) to "G" or "H" to define the basic event as a seismic event.

2. Place the Seismic Failure value in the .BEI "prob" position.

3. If a calculation type of "G" is used, specify an earthquake "G-Level". Place it in the .BEI "Lambda" position.

Loading the Seismic Uncertainty Data

Loading of the seismic uncertainty data is similar to the procedures discussed in Section 4.6. Two methods can be used to load seismic uncertainty data. The data can be entered in the Modify → Basic Events main menu option or from a basic event flat file (.BEI) and loaded through MAR-D as described in Appendix A. Differences between loading seismic data and the procedures discussed in Section 4.6 are outlined below.

Loading Through the Modify→ Basic Events main menu option

To enter seismic uncertainty data into a seismic basic event record, go to the "Uncertainty Data Calculation Type". Enter an "S", which defines the basic event as a seismic basic event. Enter the Beta-R and the Beta-U in their respected blocks.

Loading Through the MAR-D Interface.

The loading of seismic uncertainty data through the MAR-D interface is similar to the procedures described in Section 4.6 for loading internal basic event uncertainties. The seismic basic event flat file (.BEI) data format is similar to that described in Appendix B except for the following:

1. Set the uncertainty type (UdT) to "S" to allow for the implementation of seismic uncertainties.

2. Specify the seismic uncertainty term representing the random uncertainty, Beta-R. Place this value in the .BEI UdValue position. Specify the confidence uncertainty term, Beta-U, and place it in the .BEI UdValue2 position.

Defining Internal Event Susceptibility to Seismic Activity

In order to integrate the internal event analysis with a seismic analysis, the internal basic event must be transformed into the new seismic event. This process first involves defining the internal basic event as seismically susceptible. Basic event susceptibility can be entered into the SAPHIRE database through either the Modify → Basic Events main menu option or by way of a basic event attribute flat file (.BEA) loaded through the Utility → Load and Extract main menu option. Both methods are discussed below.

Defining Susceptibility Through the Modify→ Basic Events main menu option.

An internal event that is determined to be seismically vulnerable is defined in SAPHIRE as seismically susceptible. This is done under the Modify → Basic Events main menu option. Highlight the desired internal event and chose Modify from the popup menu. Select the Attributes tab and check the Seismic box in the Susceptibilities area. This will identify the basic event as susceptible to seismic initiators.

Defining Susceptibility Through MAR-D.

An internal basic event flat file (.BEA) can be generated from MAR-D as is described in Appendix A. The file format of the .BEA is described in Appendix B. To define a basic event as seismic susceptible, attribute 4 (att4) must be changed from "N" to "Y". Reloading this .BEA file with the seismic susceptible attribute is described in Appendix A.

Defining the Internal Basic Event Transformations

A transformation is a replacement or addition inside the fault tree logic. An internal event that is determined to be seismically vulnerable needs to be transformed into a new seismic event in SAPHIRE. During the transformation process, the internal basic event is replaced with a seismic basic event or a series of seismic events.

SAPHIRE utilizes three types of transformations: (1) AND, (2) OR, and (3) ZOR. An "AND" type transformation replaces the event being transformed with an AND gate having any transformed events as inputs. An "OR" type transformation replaces the event being transformed with an OR gate having any transformed events as inputs. A "ZOR" type transformation implies that if any transformed events from the original transformed event fail, then all events fail. Since for seismic analysis, an internal random basic event is transformed into one new seismic basic event, the transformation type should be "OR". This will prevent the random event and the seismic event from being "ANDed" together during the seismic analysis.

Basic event transformation also requires a "transformation level" that indicates the level of substitution for the transformation. The transformation is an integer between 0 and 255. For seismic analysis, the transformation level is generally either 0 or 1.

Transformation data can be entered into the SAPHIRE database using either the Modify → Basic Events main menu option or from a basic event transformation flat file (.BET) loaded through the Utility → Load and Extract main menu option. Both methods are discussed below.

Loading Seismic Transformations Using the Modify → Basic Event main menu

Basic event transformation is accomplished in SAPHIRE through the "Modify → Basic Events main menu option. This is done by with the following steps:

1. Highlight the desired internal event and choose “Modify” from the popup menu.
2. Select the Transformations tab.
3. Choose the transformation type (usually “OR”) and enter the transformation level (usually 0 or 1).
4. From the “All Events” list located on the left side of the dialog, highlight one or more seismic events you wish to transform the original event, and click the Add button. The selected transformation events will appear on the right side of the dialog in the “Selected Event” area. Repeat this process until all desired seismic events have been included.
5. Choose the OK button to save the changes

Loading Seismic Transformations with the MAR-D Utility

Basic event transformation may also be loaded into SAPHIRE through a MAR-D file (.BET). Below is the MAR-D file format for the SAPHIRE version 6 basic event transformation file (.BET). (The version 7 format is the same, but can accommodate up to 24 character names.)

File Name:

xxxxxx.BET

File Format:

project =
name1,level,type
bename1, bename2, . . . ,
. . . , benameN
^EOS
name2,level,type
bename1, bename2, . . . ,
. . . , benameN
^EOS

Where

Project	16 character	Project name
Name	16 character	Event name
Type	4 character	Transformation type
Level	3 character	Transformation level
bename1..N	16 character	Event name

The loading of a MAR-D flat file into SAPHIRE is described in detail in Appendix A.

GENERATING AND QUANTIFYING SEISMIC CUT SETS

Generating and quantifying seismic cut sets at both the fault tree level and the sequence level is similar to that for internal (random) analysis described in Sections 4.5.4 and 4.5.7, respectively. The few minor differences are noted below.

Generating Seismic Cut sets

When generating seismic cut sets during both fault tree and sequence analysis, you must specify that seismic analysis is desired. This is accomplished in both the "Fault Trees" and "Sequences" main menu options of SAPHIRE. To change from "Random" analysis to "Seismic" analysis, you have two options:

1. Open the Define Constants dialog found under the Utility → Define Constants main menu option. On the "General" tab, select "Seismic" from the analysis type combo box.

2. Open the "Fault Trees" or "Sequences" dialogs found under the corresponding main menu options. Select "Seismic" from the analysis type combo box located in the dialog.

Quantifying Seismic Cut sets

When quantifying seismic cut sets during both fault tree and sequence analysis, you should confirm that the "Analysis type" is set to "Seismic". In addition, after selecting "Quantify" from the popup menu option, you must choose the "G-Level" for which quantification is to be performed. The options available for "G-level" quantification include:

1. Selecting one of the g-level bins that contain a non-zero value obtained from the hazard histogram identified for use with the current project.

2. Selecting "ALL COMBINED". This gives an overall value obtained by adding the data using all bins in the histogram.

3. Selecting "ALL SEPARATE". This quantifies the cut sets at each g-level bin that contains a non-zero value obtained from the hazard histogram used with the current Family. It should be noted that after quantification using the "ALL SEPARATE" option, the cut set list for each g-level is not maintained. When quantification is completed, only the last quantification performed (at that specific g-level) is available. However, numerical results are stored and are available for each individual g-level that was calculated. These individual results are generally used during uncertainty analysis.

NRC FORM 335 (2-89)
NRCM 1102, 3201. 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

(See Instructions on the reverse)

1. REPORT NUMBER (Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)
NUREG/CR-6952
INL/EXT-05-00643

2. TITLE AND SUBTITLE
Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 7 Data Loading Manual

3. DATE REPORT PUBLISHED
MONTH: September
YEAR: 2008

4. FIN OR GRANT NUMBER
N6203

5. AUTHOR(S)
K. J. Kvarfordt, S. T. Wood, C. L. Smith

6. TYPE OF REPORT
Technical

7. PERIOD COVERED (Inclusive Dates)

8. PERFORMING ORGANIZAITON - NAME AND ADDRESS (If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)
Idaho National Laboratory
Battelle Energy Alliance
P.O. Box 1625
Idaho Falls, ID 83415-3850

9. SPONSORING ORGANIZATION - NAME AND ADDRESS (If NRC, type "Same as above"; If contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)
Division of Risk Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

10. SUPPLEMENTARY NOTES
D. O'Neal, NRC Project Manager

11. ABSTRACT (200 words or less)
The Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) is a software application developed for performing a complete probabilistic risk assessment (PRA) using a personal computer. SAPHIRE is primarily funded by the U.S. Nuclear Regulatory Commission (NRC) and developed by the Idaho National Laboratory. This report is intended to assist the user to enter PRA data into the SAPHIRE program using the built-in MAR-D ASCII-text file data transfer process. Towards this end, a small sample database is constructed and utilized for demonstration. Where applicable, the discussion includes how the data processes for loading the sample database relate to the actual processes used to load a larger PRA models. The procedures described herein were developed for use with SAPHIRE Version 6.0 and Version 7.0. In general, the data transfer procedures for version 6 and 7 are the same, but where deviations exist, the differences are noted. The guidance specified in this document will allow a user to have sufficient knowledge to both understand the data format used by SAPHIRE and to carry out the transfer of data between different PRA projects.

12. KEY WORDS/DESCRIPTORS (List words or phrases that will assist researchers in locating the report.)
SAPHIRE, software, reliability, risk, safety, PRA, data, MAR-D

13. AVAILABILITY STATEMENT
Unlimited

14. SECURITY CLASSIFICATION
(This page)
Unclassified
(This report)
Unclassified

15. NUMBER OF PAGES

16. PRICE

NRC FORM 335 (2-89)