INL/EXT-09-17015

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 7 Data Loading

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

March 2011

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

INL/EXT-09-17015
NUREG/CR-7039

Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 7 Data Loading

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

March 2011

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 N6423

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, 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

S. T. Wood, C. L. Smith, K. J. Kvarfordt, S. T. Beck, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 1 Summary Manual*, NUREG/CR-6952, August 2008.

C. L. Smith, S. T. Wood, W. J. Galyean, J. A. Schroeder, S. T. Beck, M. B. Sattison, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 2 Technical Reference*, NUREG/CR-6952, August 2008.

K. J. Kvarfordt, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 3 Code Reference Manual*, NUREG/CR-6952, August 2008.

S. T. Beck, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 4 Tutorial*, NUREG/CR-6952, August 2008.

C. L. Smith, J. Schroeder, S. T. Beck, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 5 GEM Manual*, NUREG/CR-6952, August 2008.

C. L. Smith, R. Nims, K. J. Kvarfordt, C. Wharton, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 6 Quality Assurance Manual*, NUREG/CR-6952, August 2008.

K. J. Kvarfordt, S. T. Wood, C. L. Smith, *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Vol. 7 Data Loading Manual*, NUREG/CR-6952, August 2008.

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 Version 8 is funded by the U.S. Nuclear Regulatory Commission 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 8. 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 (NRC) has developed the Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) software that is 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.

For nuclear power plant PRAs, 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 also can 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, 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 and users' needs. Starting with Version 5, SAPHIRE operated in the Microsoft Windows™ environment. Versions 6 and 7 included features and capabilities for developing and using larger, more complex models. SAPHIRE Version 8 includes significant new features and capabilities to meet user needs for NRC risk-informed programs. In general, these include:

- Improved user interfaces supporting NRC's Significance Determination Process, event and condition assessments, and more detailed types of PRA analyses.
- Development and use of NRC's Standardized Plant Analysis Risk models.
- New and improved solving algorithms.
- Support features for user-friendliness.

This NUREG-series report comprises seven volumes as outlined below and incorporates new features and capabilities of Version 8.

Volume 1, "Overview and Summary"

Volume 1 provides an overview of the functions and features available in SAPHIRE Version 8 and presents general instructions for using the software.

Volume 2, "Technical Reference"

Volume 2 summarizes the fundamental mathematical concepts of sets and logic, fault trees, and probability. It then describes the algorithms used to construct a fault tree and to obtain the minimal cut sets. This report presents the formulas used to obtain the probability of the top event from the minimal cut sets and the formulas for probabilities that apply for various assumptions concerning reparability and mission time. In addition, it defines the measures of basic event importance that SAPHIRE can calculate. This volume also gives an overview of uncertainty analysis using simple Monte Carlo sampling or Latin Hypercube sampling and states

the algorithms used by this program to generate random basic event probabilities from various distributions. Finally, this report discusses enhanced and new capabilities such as post-processing rules, integrated model solving using model types, and workspace analysis routines.

Volume 3, "Users' Guide"

Volume 3 provides a brief discussion of the purpose and history of the software as well as general information such as installation instructions, starting and stopping the program, and some pointers on how to get around inside the program. Next, it discusses database concepts and structure. The following nine sections (one for each of the menu options on the SAPHIRE main menu) furnish the purpose and general capabilities for each option. Finally, Volume 3 provides the capabilities and limitations of the software.

Volume 4, "Tutorial"

Volume 4 provides a series of lessons that guide the user through basic steps common to most analyses performed with SAPHIRE.

Volume 5, "Workspaces"

Volume 5 describes the functionality and process behind SAPHIRE Version 8 workspaces. Workspaces provide an area in which a PRA model can be analyzed to obtain risk insights for a given initiating event or condition. Workspaces replace the "Graphical Evaluation Module" in earlier SAPHIRE versions.

Volume 6, "Quality Assurance"

Volume 6 is designed to describe how the SAPHIRE software quality assurance (QA) is performed for Version 8, what constitutes its parts, and the limitations of those processes. In addition, this report describes the Independent Verification and Validation that was conducted for Version 8 as part of an overall QA process.

Volume 7, "Data Loading"

Volume 7 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. In addition, this manual covers loading data through the new Accident Sequence Matrix and discusses the Project Integrate interfaces with SAPHIRE.

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

CONTENTS

Section Page

PREVIOUS REPORTS ii
ABSTRACT iii
FOREWORD v
LIST OF FIGURES ix
LIST OF TABLES ix
EXECUTIVE SUMMARY xi
ACKNOWLEDGEMENTS xiii
ACRONYMS xv
1. INTRODUCTION 1
1.1 Background 1
1.2 Assumptions and Recommendations 3
2. OVERVIEW OF DATABASE CONCEPTS 5
2.1 SAPHIRE Database Unit - The Project 5
2.2 File Management 5
3. THE SAMPLE DATABASE 9
3.1 The Sample Database 9
3.2 The Sample Database Event Tree 9
3.3 The Sample Database Fault Trees 11
3.4 The Sample Database Basic Events 14
3.5 Sample Database Fault Tree Cut Sets 16
3.6 Sample Database Sequence Cut Sets 17
3.7 Sample Database Post-processing Actions 18
3.8 Sample Database Uncertainty 18
3.9 Sample Database Importance 19
4. LOADING THE SAMPLE DATABASE 21
4.1 Introduction 21
Flat File Data Importing/Exporting 21
4.2 Adding and Selecting the Database Project 22
4.2.1 Adding the Project 22
4.2.2 Selecting the Project 22

4.2.3 Entering Project Information, Description, and Text 22
4.2.4 Extracting and Verifying the Project Data 24
4.3 Loading the Event Tree Data 24
4.3.1 Entering the Event Tree Logic 26
4.3.2 Entering Sequence Names in Graphics 27
4.3.3 Entering Top Event Descriptions 28
4.3.4 Entering Link (Substitution) Rules 29
4.3.5 Generating and Verifying Event Tree Logic 30
4.4 Entering End State Data 32
4.4.1 Entering End State Names in Graphics 32
4.4.2 Entering End States for Analysis 33
4.4.3 Entering End State Description and Text 33
4.5 Loading the Fault Tree Data 33
4.5.1 Entering Fault Tree Logic 35
4.5.2 Entering Fault Tree Descriptions and Text 37
4.5.3 Entering Gate Descriptions and Attributes 38
4.5.4 Generating Fault Tree Cut Sets 39
4.5.5 Verifying the Fault Tree Data 40
4.6 Loading Basic Event Data 40
4.6.1 Adding/Modifying Basic Events 42
4.6.2 Basic Event Flat File Formats 42
4.6.3 Additional Basic Event Data Flat Files 42
4.7 Loading Sequence Data 47
4.7.1 Generating Sequence Cut Sets 47
4.7.2 Entering the Sequence Description and Text 48
4.8 Post-processing Actions 49
4.9 Analyzing Uncertainty 50
4.9.1 Generating Uncertainty for Fault Tree Cut Sets 50
4.9.2 Generating Uncertainty for Sequence Cut Sets 51
4.9.3 Generating Uncertainty for End States 51
4.9.4 Generating Uncertainty for Groups of Sequences or the Project 52
5. Other Data Loading Methods 53
5.1 Loading Data via an Accident Sequence Matrix 53
5.2 Integrate Project Utility 54
Appendix A - Procedures for Database Loading A-1
Appendix B - General MAR-D Data Interchange Formats B-1
Appendix C - MAR-D Files for Sample Database C-1

LIST OF FIGURES

Figure — Page

Figure 1. Going-to-work (WORK) event tree 10

Figure 2. Alarm clock failure fault tree 12

Figure 3. Personal problems fault tree 12

Figure 4. Transportation failure fault tree (normal time frame) 13

Figure 5. Transportation failure fault tree (late time frame) 13

Figure 6. Modify Project dialog 23

Figure 7. Going to work event tree graphic 25

Figure 8. Sequence generation logic report 31

Figure 9. Fault tree graphical editor 36

Figure 10. The modify basic event dialog 41

Figure 11. Information for an accident sequence matrix file 53

Figure 12. First step of the Project Integration option 55

LIST OF TABLES

Table — Page

Table 1. SAPHIRE database file names and descriptions. 7

Table 2. Basic event values for the sample problem. 14

Table 3. Basic event descriptions for the sample problem. 15

Table 4. Fault tree cut set results. 16

Table 5. Sequence cut set results. 17

Table 6. Fault Tree uncertainty values report. 18

Table 7. Sequence uncertainty values report. 18

Table 8. End state uncertainty values report. 19

Table 9. Results of sample database importance analysis. 19
Table 10. Extracted project flat files. 24
Table 11. Extracted project flat files for the sample project. 24
Table 12. Extracted event tree flat files (with logic and sequence names only). 27
Table 13. Event tree file (with logic, sequence, end state name, and top event descriptions)... 28
Table 14. Extracted event tree description and text flat files. 29
Table 15. Extracted event tree rules flat file. 30
Table 16. Extracted sequence logic flat files. 31
Table 17. Extracted end state flat files. 32
Table 18. Extracted fault tree logic and graphic flat files. 37
Table 19. Extracted fault tree descriptions and text flat files. 37
Table 20. Extracted fault tree gate flat files. 38
Table 21. Extracted fault tree cut sets flat files. 40
Table 22. Extracted basic event descriptions flat file. 44
Table 23. Extracted basic event data flat files. 45
Table 24. Extracted sequence cut sets flat files. 48
Table 25. Extracted sequence description and text flat files. 49
Table 26. Extracted fault tree attributes (uncertainty) flat file. 51

EXECUTIVE SUMMARY

The Data Loading report 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 Version 8 project is described in terms of the ASCII-formatted data structures available via the Load and Extract (aka, MAR-D) option. MAR-D is a mechanism in SAPHIRE to import or export probabilistic risk assessment data – via an open 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 are defined and discussed in this document include:

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
Basic event category
Basic event grade

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

ACKNOWLEDGEMENTS

We would like to specifically acknowledge Mr. Dan O'Neal of the U.S. Nuclear Regulatory Commission for his contribution to the development this report.

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) Version 8

Volume 7 Data Loading

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) Version 8.

Using SAPHIRE 8 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 figures 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. Version 6 was a Windows NT version that became a released code in 1998. Version 7 is also a Windows NT (or above) version that is currently the standard that is being used by the NRC.

Work began on a new version of SAPHIRE, Version 8, in 2004. Version 8 was designed to meet current NRC program needs such as those related to SPAR model development, the Significance Determination Process (SDP) program, the Risk Assessment Standardization Project (RASP), as well as the Accident Sequence Precursor (ASP) Program. The development of the SAPHIRE 8 version includes new features and capabilities. These features and capabilities are related to working with larger, more complex models and improving the user-friendliness of SAPHIRE's interfaces while retaining key functionality of Version 7.

Version 8 is being developed to support the SPAR models and to run them as an integrated model (e.g., Level 1 with external events). The graphical user interface has also improved from SAPHIRE 7. A tailored interface for the SDP and the ASP programs has been developed. The interfaces for the SDP, ASP, and general analysis introduce the concept of a "workspace" in which the analyst may run and save different analyses. The use of workspaces enables the user to separate the model construction from the model analysis.

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 8, 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 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 Saphire8 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 – this main window is called the Standard Analysis interface. These main menu options will be referred to frequently throughout this manual. The main menu functions used predominantly in data loading are

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

2.2 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 typically found in the SAPHIRE 8 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 type, the following relational files exist:

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

These file types should *never* be deleted unless the project is to be removed permanently from the user's hard drive.

In addition to the relation files, SAPHIRE can also produce external *flat* or ASCII files. These can be extracted or loaded to or from any Windows project subdirectory using SAPHIRE software. These flat files, grouped according to the type of data they contain, are listed in Table 1. In version 8, 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 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 8 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 1.

While MAR-D files may be used to import and export information, they also may be used to check information (e.g., by spell checking descriptions, by evaluating basic event probabilities). In addition, SAPHIRE 8 has a “project check” option that is found under **Tools → Check Project** that automates a variety of quality checks on the project information.

Table 1. SAPHIRE database file names and descriptions.

File name	Description	Applicable section in this manual
ProjectName.FAD	Project name and descriptions	4.2.3
ProjectName.FAA	Project attributes	4.2.3
ProjectName.FAT	Project text	4.2.3
ProjectName.FAY	Project event tree post-processing rules	-
ProjectName.FAS	Project fault tree post-processing rules	-
ProjectName.FAP	Project end state partition rules	-
ProjectName.BED	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	Basic event transformation information	- -
ProjectName.BEC	Basic event compound information	-
ProjectName.BEN	Basic event notes	-
ProjectName.BEG	Basic event grade	Appendix B
ProjectName.BECat	Basic event categories	Appendix B
ProjectName.FMD	Failure mode descriptions	(deprecated)
ProjectName.CTD	Component type descriptions	(deprecated)
ProjectName.STD	System type descriptions	(deprecated)
ProjectName.LCD	Location descriptions	(deprecated)
ProjectName.TTD	Train descriptions	(deprecated)
ProjectName.FTD	Fault tree names and descriptions	4.5.2
FaultTree.FTT	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 post-processing rules	-
FaultTree.PID	Fault tree P&ID	(deprecated)
EventTree.ETG	Event tree graphics	4.3.1
ProjectName.ETD	Event tree names and descriptions	4.3.5
ProjectName.ETT	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 post-processing rules	-
EventTree.ETP	Event tree end state partition rules	-
ProjectName.ESD	End state names and descriptions	4.4.3
ProjectName.EST	End state text	4.4.3
ProjectName.ESC	End state cut sets	-
ProjectName.SQD	Sequence names and descriptions	4.7.2
ProjectName.SQC	Sequence cut sets	4.7.1
ProjectName.SQA	Sequence attributes	Appendix B
ProjectName.SQT	Sequence text	4.7.2
ProjectName.SQL	Sequence logic	4.3.6
ProjectName.SQY	Sequence post-processing rules	-
ProjectName.SQP	Sequence end state partition rules	-
ProjectName.GTD	Gate description	4.5.3
ProjectName.GTA	Gate attributes	4.5.3
ProjectName.CSD	Change/flag set description	4.10.1/Appendix A
ProjectName.CSI	Change/flag set information	4.10.1/Appendix A
ProjectName.CSA	Change/flag set alternate names	-
ProjectName.HIA	Histogram attributes	-
ProjectName.HID	Histogram descriptions	-
ProjectName.HII	Histogram information	-
ProjectName.HIA	Histogram alternate name	
ProjectName.SLA	Slice alternate names	-
ProjectName.SLB	Slice basic event logic	-
ProjectName.SLD	Slice description	-
ProjectName.SLI	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 (Volume 4), a simple example shows the quantification for the frequency of an event tree. The tutorial describes (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. Sections 3.2 through 3.9 of this report 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 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.

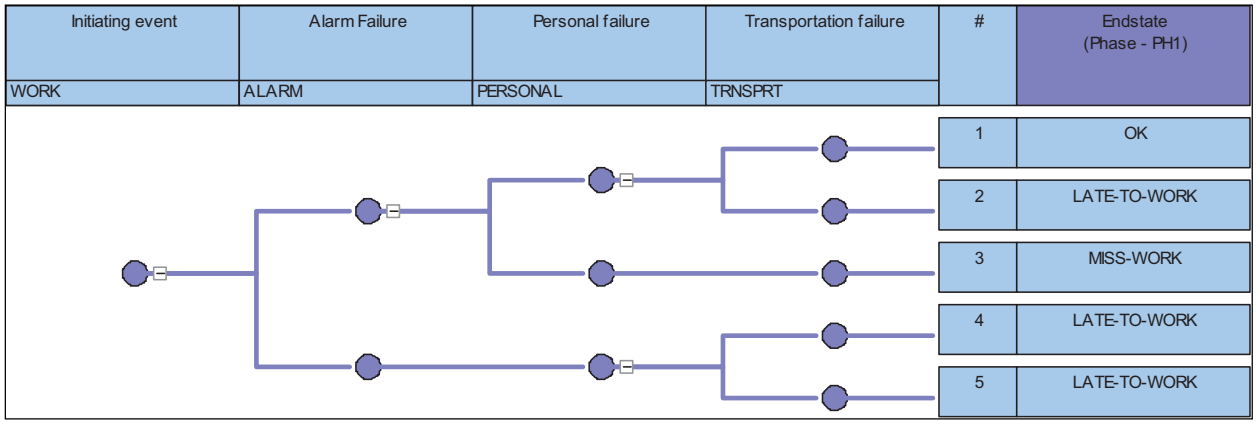


Figure 1. Going-to-work (WORK) event tree

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 TRNS-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 2 through 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 (Figure 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) 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.

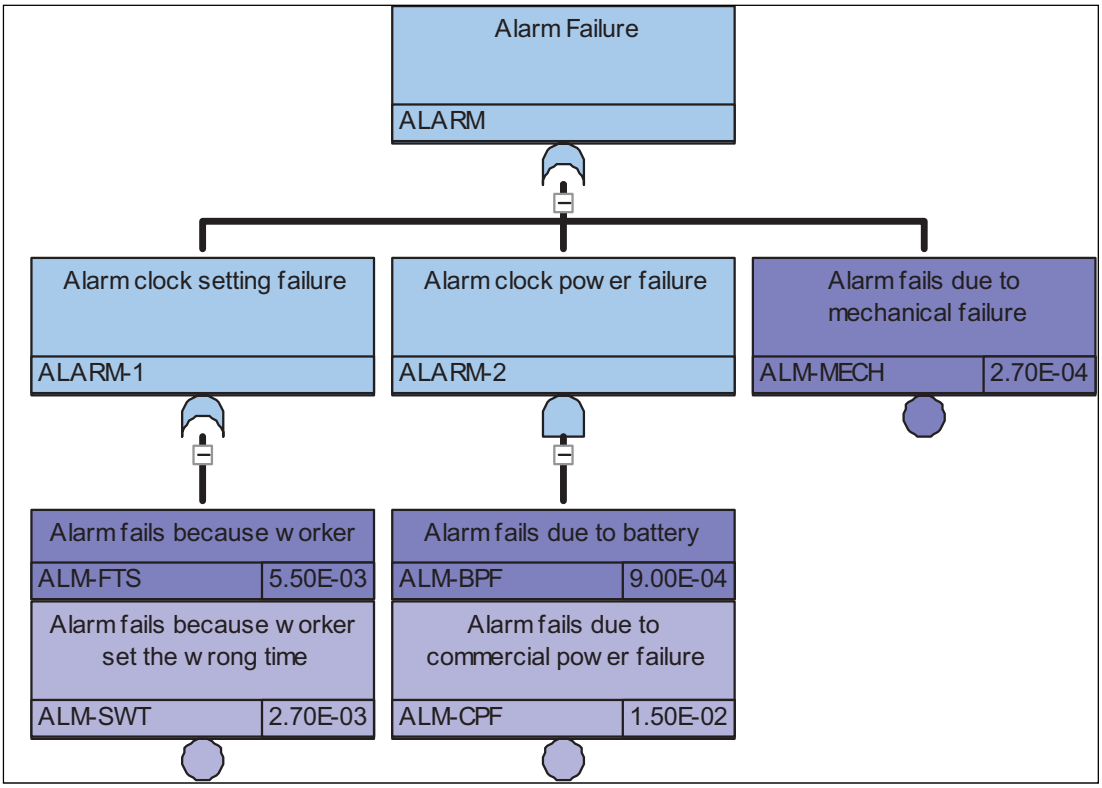


Figure 2. Alarm clock failure fault tree

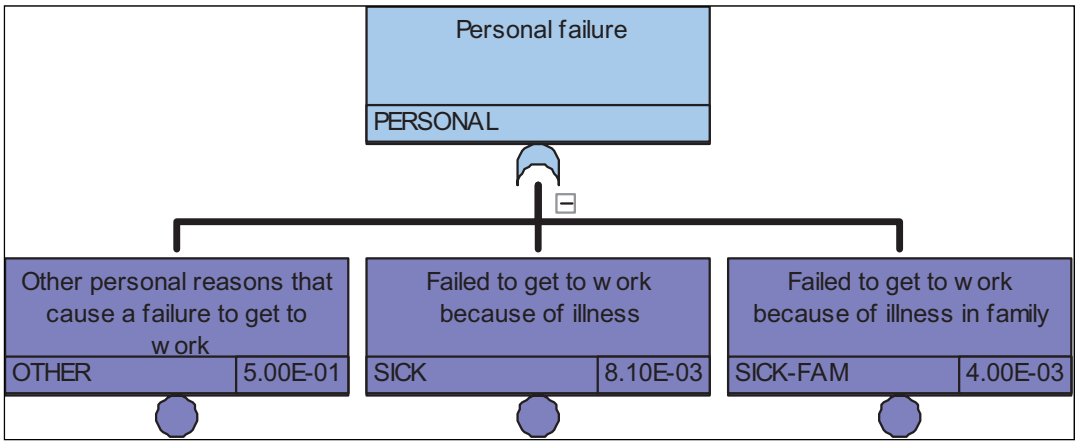


Figure 3. Personal problems fault tree

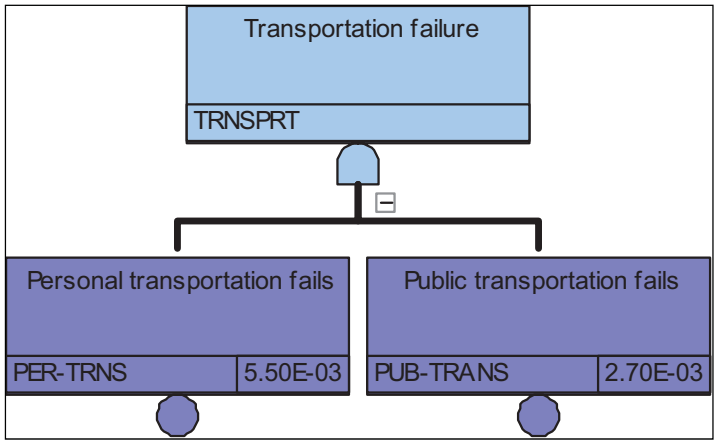


Figure 4. Transportation failure fault tree (normal time frame)

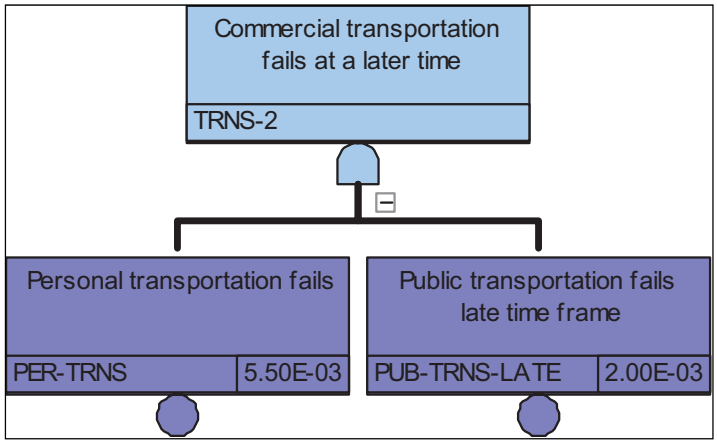


Figure 5. Transportation failure fault tree (late time frame)

The third fault tree TRNSPRT (Figure 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 TRNS-2 (Figure 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 (TRNS-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 2 and Table 3. 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 2 is the lognormal distribution error factor.

Table 2. Basic event values for the sample problem.

Basic event	Distribution type	Calculation type	Mean value	Uncertainty value (error factor)
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. 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 external event analysis features are covered. Consequently, fire, flood, and seismic analysis are not directly addressed by way of the sample database. However, additional model types simply result in additional objects (such as basic events) appearing in the MAR-D file, for example if a basic event DG-B were assigned to RANDOM, FIRE, and FLOOD model types, the in the basic event information (.BEI) file we would see (notionally):

```
* Name       ,FdT,UdC,UdT, UdValue , Prob      ,  … Analysis Type          , Phase Type
DG-B                , 1, 6                    , L, 1.000E+001, 2.000E-002, …  , RANDOM                ,
DG-B                , 1,                      ,  , 3.000E+000, 2.000E-002, … , FIRE                  ,
DG-B                , 1,                      ,  , 3.000E+000, 2.000E-002, …  , FLOOD                 ,
```

3.5 Sample Database Fault Tree Cut Sets

The fault tree cut sets and minimal cut set (mincut) upper bound for those fault trees contained in the sample database are shown in Table 4. The fault tree modeling of "personal failure due to sickness and other reasons" has the greatest probability of occurring.

Table 4. 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 5 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 5. 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 Post-processing Actions

Sequence 3 shown in the sequence cut set list in Table 5 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 post-processing or recovery action added *after* the sequence cut set generation.

3.8 Sample Database Uncertainty

The following tables (Table 6 to Table 8) 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 6 lists the uncertainty results for the fault trees, Table 7 lists the uncertainty results for each of the sequences, while Table 8 lists the uncertainty results for the end states.

Table 6. 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 7. Sequence uncertainty values report.

Event Tree Seq	**Mean**	**Min. Cut Upper Bound**	**Median**	**Std. Dev.**	**5th %**	**95th %**	**Minimum**	**Maximum**
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 8. End state uncertainty values report.

End State	Mean	Min. Cut Upper Bound	Median	Std. Dev.	5th %	95th %	Minimum	Maximum
LATE-TO-WORK	6.84E-01	6.75E-001	2.47E-01	1.49E+00	2.29E-02	2.56E+00	1.21E-3	2.21E+01
MISS-WORK	3.46E+00	3.99E+000	1.96E+00	4.88E+00	3.62E-01	1.10E+01	5.85E-2	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 8 shows the results of the importance analysis for the sample database.

Table 9. 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 Flat File Data Importing/Exporting

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 Actions

Section 4.9 Analyzing Uncertainty

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. Note that the MAR-D files associated with the sample database are fully listed in Appendix C.

Flat File Data Importing/Exporting

Most project data can be imported or exported using specific flat file formats, ether for the entire project or in sections. This may be useful to create template of common data for similar projects. Each of the following sections gives the flat file extension and formatting. To learn how to import, export or edit these sections, see Appendix A.

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 named Windows subfolder is also created in the specified project location. All files for this project will consequently be added to this folder.

To add a project while in an existing project, from the main menu select (**File → New → Project**). If there no previously opened project then the opening window has a Create 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 (**File → Open Existing Project**) from the menu option, navigating to the desired project folder, and selecting the project file name located in the project folder or a zipped project.

4.2.3 Entering Project Information, Description, and Text

Project description, information, and text can be entered into the database when the project is open. To add the project description (“This is a sample database”) or the project notes ("A simple example that models the probability of getting to work on time"), choose the (**Project → Modify**) option from the SAPHIRE main menu to open the Edit Project dialog. Project information that can be stored using this option includes name, description, creator, creation date, IE frequency units, facility location, version, notes. These fields and names of the fields are user definable (for example, “Company” may be renamed to “Vendor”).

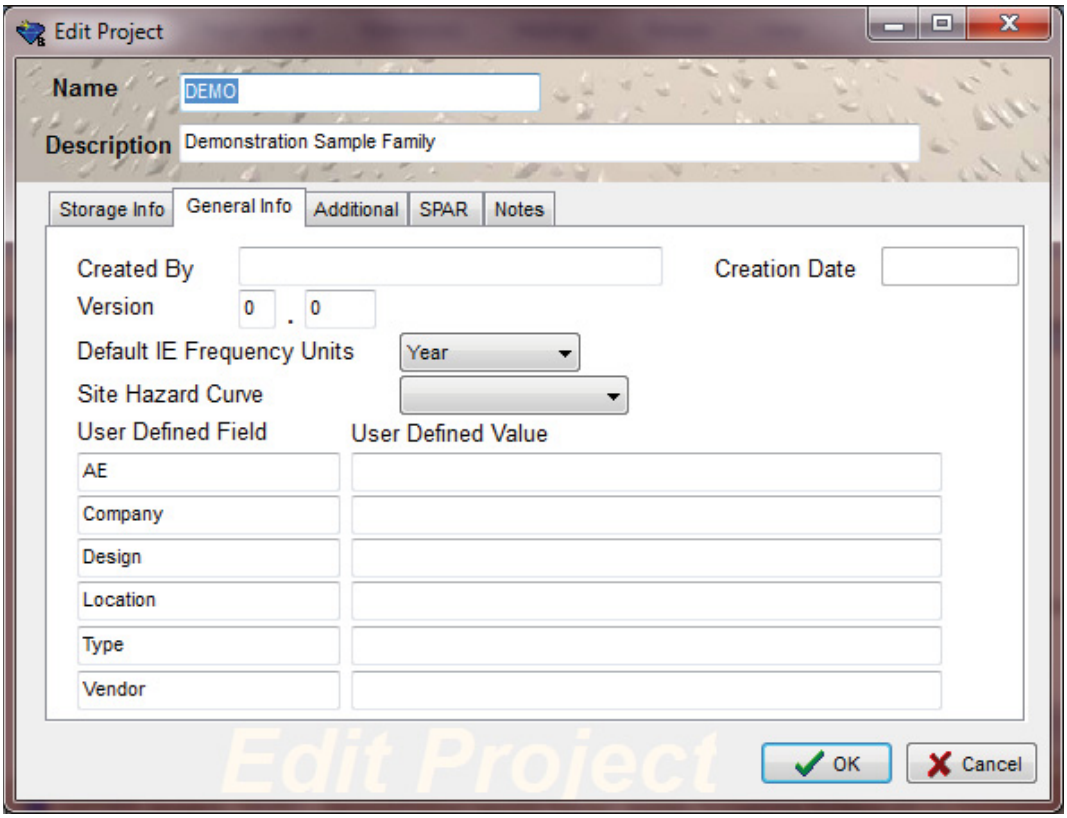


Figure 6. Modify Project dialog

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 project information can be extracted or imported .FAA, .FAD and .FAT. 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 11) 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 10. 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 11. Extracted project flat files for the sample project.

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 listing on the left side of the main window as long as the list filter is “Main Trees”. An event tree without an initiating event will be included in the event tree list only when the “Sub Trees” filter is selected. (See Appendix A for more info on showing/hiding item lists and item list filters.)

Top events can be found in the Fault Tree Listing with a filter of “Main Trees”. Top events are also listed in the Basic Event Listing when filtered by “Developed Event”, and initiating events with their corresponding “Initiator” filter. 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 PRAs. The sample database event tree to be loaded is shown in Figure 7.

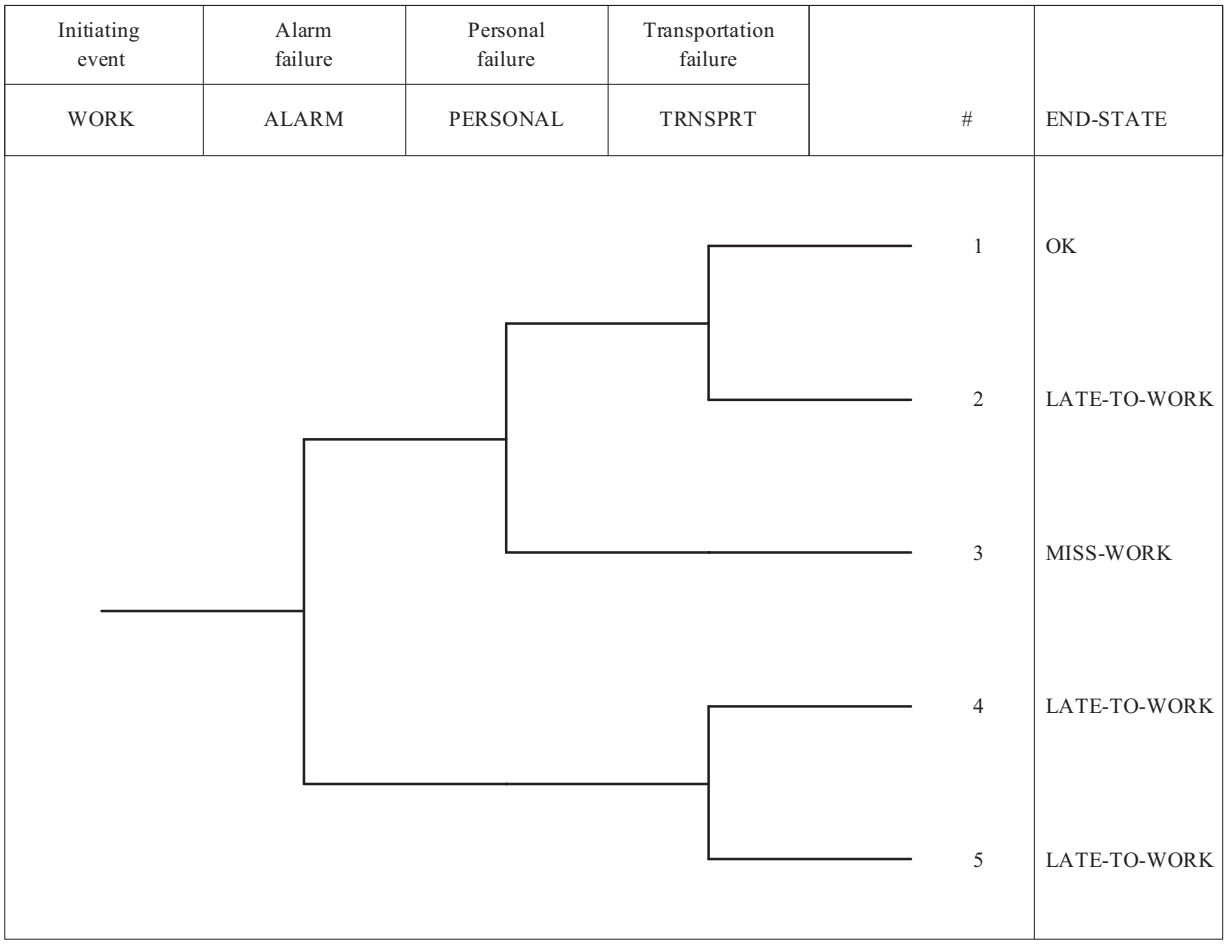


Figure 7. Going to work event tree graphic

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 event tree descriptions and text (Section 4.3.4)

5. Entering link (substitution) rules (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. Most items can be edited by both double clicking on the items or by having the quick edit on and just selecting the desired item to edit.

To create a new event tree you (a) double click "New event tree" in the Event Tree Listing (b) enter the event tree properties Name, Description, etc. (c) add the event tree structure, as shown in Figure 7, by selecting the desired button for adding branches or tops.

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. The flat files that will contain data are the event tree graphics (.ETG) and the event tree logic (.ETL) shown in Table 12. These two files are identical in SAPHIRE.

Table 12. Extracted event tree flat files (with logic and sequence names only).

Files	Extracted file information
.ETG and .ETL	SAMPLE, WORK, WORK = ^WINVER2.1 ^PHASES 1 5 1 //# phases defined, max count sequences, initial phase PHASE_1 16155777 Phase 1 ^TOPS ALARM, PERSONAL, TRNSPRT ^LOGIC 1 // initial phase, following are offset +1.0 +2.0 +3.0 -3.0 -2.0 3.0 -1.0 2.0 +3.0 -3.0 ^SEQUENCES 0 // offset from initial phase N, Endstate, N, Sequence Name, N, Frequency, N, Extra, Y, , Y, OK, Y, , Y, , , Y, , Y, LATE-TO-WORK, Y, , Y, , , Y, , Y, MISS-WORK, Y, , Y, , , Y, , Y, LATE-TO-WORK, Y, , Y, , , Y, , Y, LATE-TO-WORK, Y, , Y, , , ^ENDSEQUENCES //Now postprocess end names ^TOPDESC ^TEXT ^PARMS ^EOS

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 blank by default. 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. The sequence name is contained in the .ETG file. Table 13 shows the event tree graphics flat files that include named sequences (A, B, C, etc) under the flag "^SEQUENCES".

Sequences for an event tree are not created 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. Event tree sequences are shown in the Event Tree List under their parent event tree.

4.3.3 Entering Top Event Descriptions

Descriptions of top events are commonly found, as was shown in Figure 7. They normally appear above the top event designator. Top event descriptions can either be added in the graphics editor by double clicking the Top or using the quick edit. In depth procedures for adding top event descriptions using the graphics editor is provided in the SAPHIRE Tutorial.

The (.ETG or ETL) file also contains the Sequence names and can be modified before importing. Table 13 shows the event tree graphics flat files that include the top event descriptions under the flag "^TOPDESC".

Table 13. Event tree file (with logic, sequence, end state name, and top event descriptions).

File	Extracted file information
.ETG and .ETL	SAMPLE, WORK, WORK = ^WINVER2.1 ^PHASES 1 5 1 //# phases defined, max count sequences, initial phase PHASE_1 16155777 Phase 1 ^TOPS ALARM, PERSONAL, TRNSPRT ^LOGIC 1 // initial phase, following are offset +1.0 +2.0 +3.0 -3.0 -2.0 3.0 -1.0 2.0 +3.0 -3.0 ^SEQUENCES 0 // offset from initial phase N, Endstate, N, Sequence Name, N, Frequency, N, Extra, 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, , , ^ENDSEQUENCES //Now postprocess end names ^TOPDESC "Initiating event" ! "Alarm Failure" ! "Personal failure" ! "Transportation failure" ! ^TEXT ^PARMS ^EOS

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 for demonstration purposes. Table 14 shows the extracted files for the description and text.

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

Interactive Modify Event Trees Method
Event tree descriptions and name can be changed in the graphic editor by **Edit → Properties** 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.

Load from Event Tree Description Flat File Method
Using a text editor, the description data can be entered into the extracted event tree description flat file (.ETD). The event tree textual data can also entered into the event tree text flat file (.ETT) the same way. After modification or development, both files must be imported as described in Appendix A.

Table 14. 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.`

4.3.4 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 15 shows the ETR file.

Link Rule Editor Method
Link rules can be added or modified through a right click option in the Event Trees Listing (Edit Linkage Rules). This is the most straightforward and simplest way to edit the rules.

The procedure for entering the link rules is to (a) select the desired event tree, (b) right click and select the "Edit Linkage Rules" option, (c) once the rule editor is open, enter the desired rule text

(d) select compile, (e) if it compiles save and exit otherwise fix any problems repeat from step d. Note that the text for the rules may be imported and exported directly from the editor (via **File → Import** or **File → Export**, respectively).

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 and a compile check is not done while adding. 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 15. 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 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 16. 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), or bitmap (BMP) 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 tree graphical editor, (b) select (**File → Export Image**) option from the main menu. To print or save multiple event trees at once, (a) select (**Publish → Event Tree Report**) main menu option, (b) select the graphics option, then (c) select the desired format and (e) press the publish 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. From the main window, select the event tree(s) to link from the Event Trees list.

2. Right Click.

3. Choose the **Link** option to open the Event Tree Link Parameters dialog.

4. Select the “Create Report” checkbox to Send Report to Screen and choose **OK**.

The report will be similar to one shown in Table 16.

Table 16. 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 .

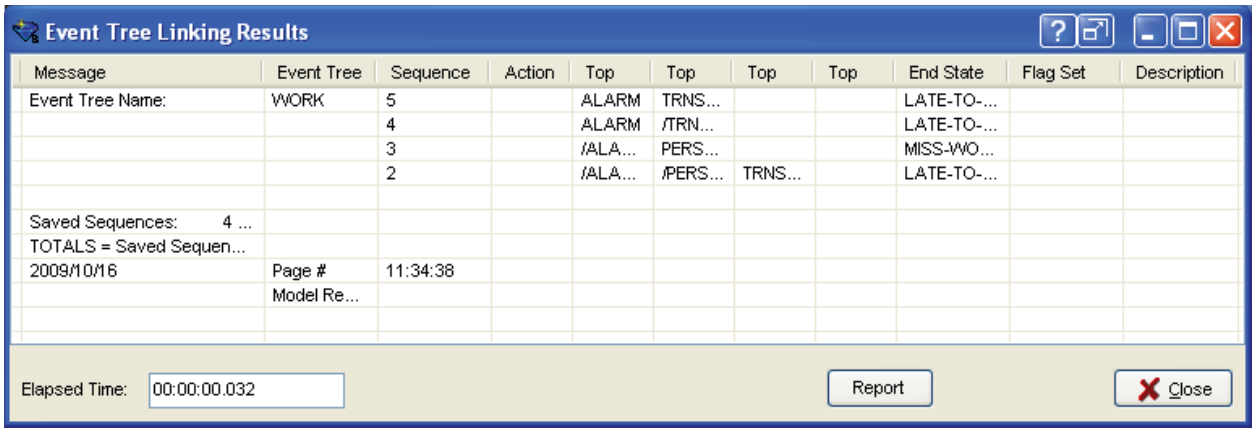


Figure 8. Sequence generation logic report

4.4 Entering End State Data

This section describes entering the end state data so that end state data is 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 17.

Table 17. 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 16 contain end state names. In depth procedures for adding end state and sequence names using the graphics editor is provided in the SAPHIRE Tutorial.

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

1. Opening the event tree graphical editor (double click the event tree from the list on the main window).

2. Double clicking sequence/end state to bring up the Edit Sequence dialog.

3. Typing in the end state name.

4. If the end state name column is not shown, select **View → End State Names** from the editor menu.

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 event tree's sequences are linked. 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.

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 can 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 process to export fault tree data is provided in Appendix A.

Note:
- Exporting logic from the fault tree editors "Export Logic" menu option does not export any basic event info. To get all the information needed for importing use the load/extract option described in Appendix A.

- 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 from the appropriate file.

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, use the SAPHIRE fault tree graphical editor to create the logic given in the hard copy. (see SAPHIRE the help file for more details)

    a. Use the SAPHIRE fault tree graphical editor to add the basic event and gate names and descriptions as discussed in the SAPHIRE help file.

    b. Extract the necessary flat files to enter the basic event descriptions (.BED), fault tree descriptions (.FTD -Section 4.5.2), and gate descriptions (.GTD - Section 4.5.3). Load these modified files using the load 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), fault tree descriptions (.FTD - Section 4.5.2), and gate descriptions (.GTD - Section 4.5.3) in the correct formats.

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

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

4. If electronic data contains 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). 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: 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 (.FTL) if available and compatible. The next best method is to enter the fault-tree structure information into SAPHIRE in the graphical editor.

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 logic information for the sample database are shown in Table 18. 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 in the SAPHIRE graphical editor.

The fault tree creation procedure requires you to (a) double click “New Fault Tree” from the Fault Trees left side list, (b) enter the fault tree info (name, description, etc), (c) 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 9.

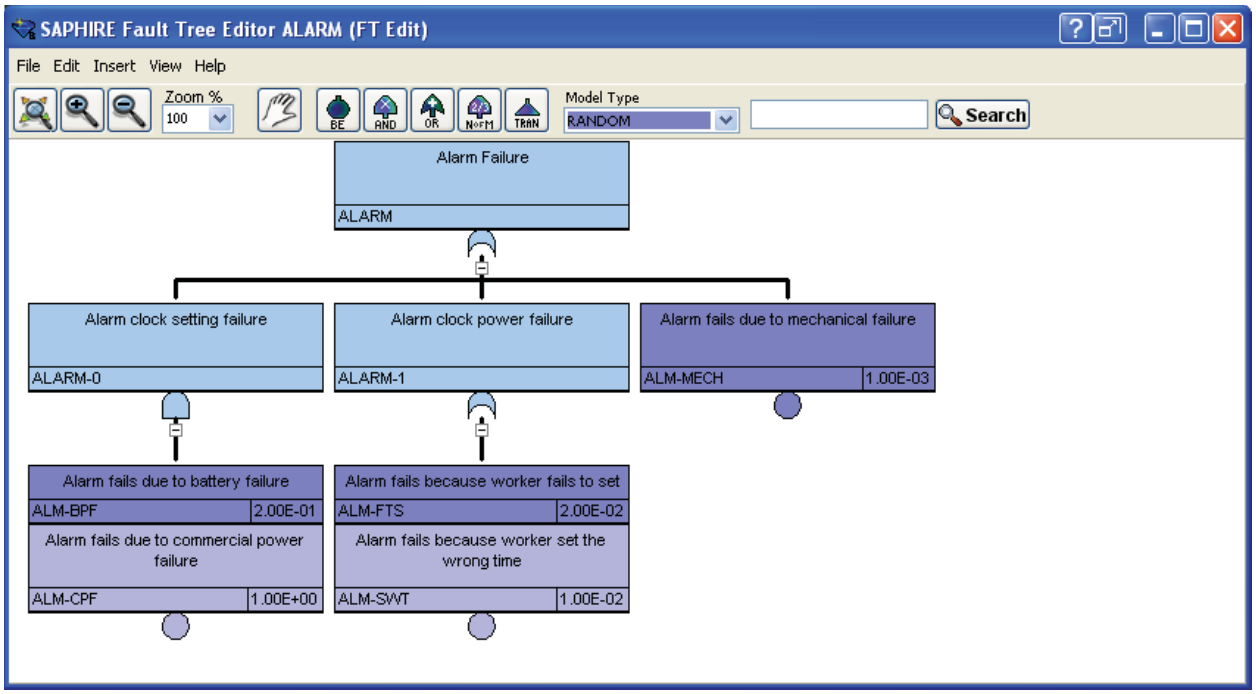


Figure 9. Fault tree graphical editor

Note:

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

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 fault tree as long as it doesn't produce a circular reference.

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 either when initially creating the fault tree or from the menu (Edit → Properties) option. This does not affect the analysis except that the fault tree list can then be successfully filtered via "Sub Trees", "Main Trees" options.

X SAPHIRE 8 **does not** have a Text Logic Editor as in previous versions of SAPHIRE.

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 Table 18. This method is not discussed further.

Table 18. Extracted fault tree logic and graphic flat files.

File	Extracted file information
.FTL	SAMPLE, ALARM = ALARM-0 AND ALM-BPF ALM-CPF ALARM OR ALARM-0 ALARM-1 ALM-MECH ALARM-1 OR ALM-FTS ALM-SWT ^EOS SAMPLE, PERSONAL = PERSONAL OR OTHER SICK SICK-FAM ^EOS SAMPLE, TRNSPRT = TRNSPRT OR PER-TRNS PUB-TRNS

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 19 contains the fault tree descriptions and text extracted. Below are the available methods for entering the fault tree descriptions and text.

Table 19. 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

The fault tree descriptions and name can be modified by using the graphical editor (double click a fault tree from the main window) then select the menu option (**Edit → Properties**). The fault tree text can be changed in the fault tree graphical editor by changing the text in the "Fault Tree Notes" section below the graphic (to view, select **View → Show Notes** from the graphical editor). 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 fault tree graphics editor is described in the SAPHIRE User's Guide.

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). Table 20 shows the fault tree gate files extracted.

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

Table 20. 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 similar to how basic even data is entered. 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, double click the desired gate; select and right click on the desired gate and choose the Edit popup menu option; or turn on quick edit and just select the desired gate.

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 21 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:

1. Open the solve form by select the desired fault tree, right click, and select solve.

2. Set the probability cutoff 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.

3. Pressing solve after selecting the appropriate cutoff values.

4. Results are then calculated and cut sets can then be viewed from there or after closing the results window by selecting the solved fault tree, right clicking, and selecting the "View Cut Sets" option.

5. Solving also make data such as “Summary Results”, “Importance Measures”, and “Uncertainty” available.

Table 21. 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 listing on the left side of the main form. (If the Basic Event list is not visible select **View → Basic Events** main menu option. However, these items 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.

Basic events can be added and modified when creating Fault Trees; by double clicking a basic event (see Figure 10) or "New basic event" in the Basic Event listing; and by using a flat file to load text based files through the (**File → Load/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.

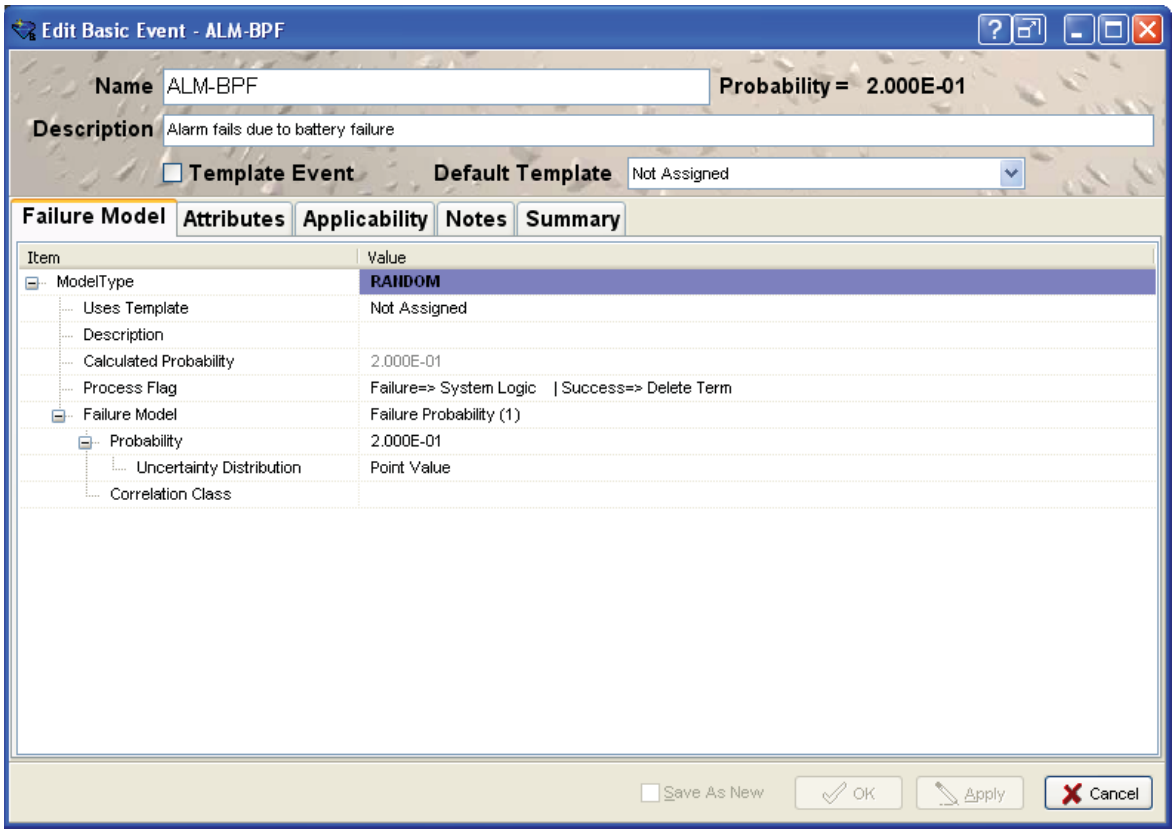


Figure 10. The modify basic event dialog

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

1. Adding/Modifying Basic Events (Section 4.6.1)

2. Basic Event Flat File Formats (Section 4.6.2)

4.6.1 Adding/Modifying 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.

Interactive Modify Basic Events Method
Basic events can be added by double clicking the "New basic event" option in the Basic Events listing. To edit existing basic event info, double click the event or right click it and select the "Edit Basic Event" menu item. This opens the Basic event Editor and displays all editable fields in the various tab section. (as seen in Figure 10) 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.

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. Other basic event info can be added with other files such as the (.BEA and .BEI) files. 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 Basic Event Flat File Formats

Basic Event Description Flat File
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 (as shown in Table 22). 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.

4.6.3 Additional Basic Event Data Flat Files

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 23 shows the basic event data flat files extracted.

Load from Basic Event Information Flat File Method.
Both the (.BEA and .BEI) files contain other basic event information that can be imported and 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.

Table 22. Extracted basic event descriptions flat file.

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

Table 23. Extracted basic event data flat files.

File	Extracted file information
.BEA	SAMPLE =
	* Name ,AltName
	,Typ,Sys,Fail,Loc,CompId,Train,Attributes ,Is Template, Template
	Name ,Use TemplateFlags
	<FALSE> , <FALSE> , , , , , , , 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, F,
	<INIT> , <INIT> , , , , , , , 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, F,
	<PASS> , <PASS> , , , , , , , 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, F,
	<TRUE> , <TRUE> , , , , , , , 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, F,
	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, F,
	ALARM3 , ALARM3 , , , , , , , 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, F, B
	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, F, B
	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, F, B
	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, F, B
	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, F, B
	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, F, B
	INIT-EV , INIT-EV , , , , , , , 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, 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, F, B
	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, 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, F,
	PUB-TRANS-LATE , PUB-TRANS-LATE , , , , , , , 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, F, B
	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, 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, F, B
SICK-FAM                , SICK-FAM                ,    , , , , ,        , , 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, F, B
TRNS-2                  , TRNS-2                  , DE , , , ,        , , 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, 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, F, F,
WORK                    , WORK                    ,    , , , , ,        , , 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, F, F,
.BEI    SAMPLE                  =
        * Name          ,FdT,UdC,UdT, UdValue , Prob      , Lambda    , Tau       , Mission
        ,Init,PF, UdValue2, Calc. Prob, Freq, Analysis Type           , Phase Type
        ALARM                   , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        ALARM3                  , 1, ,  , 3.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E+000, Y, RANDOM                 ,
        ALM-BPF                 , 1, ,  , 3.000E+000, 2.000E-001, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 2.000E-001, Y, RANDOM                 ,
        ALM-CPF                 , 1,,  , 3.000E+000, 1.020E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E+000, Y, RANDOM                 ,
        ALM-FTS                 , 1, ,  , 3.000E+000, 2.000E-002, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 2.000E-002, Y, RANDOM                 ,
        ALM-MECH                , 1, ,  , 3.000E+000, 1.000E-003, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E-003, Y, RANDOM                 ,
        ALM-SWT                 , 1, ,  , 3.000E+000, 1.000E-002, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E-002, Y, RANDOM                 ,
        BD                      , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        BE                      , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        BF                      , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        BG                      , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        EJ                      , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        INIT-EV                 , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000, I,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        OTHER                   , 1, ,  , 3.000E+000, 5.000E-002, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 5.000E-002, Y, RANDOM                 ,
        PER-TRNS                , 1, ,  , 3.000E+000, 1.007E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E+000, Y, RANDOM                 ,
        PERSONAL                , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        PUB-TRANS-LATE          , 1, ,  , 3.000E+000, 3.000E-001, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 3.000E-001, Y, RANDOM                 ,
        PUB-TRNS                , 1, ,  , 3.000E+000, 1.006E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E+000, Y, RANDOM                 ,
        SICK                    , 1, ,  , 3.000E+000, 1.030E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 1.000E+000, Y, RANDOM                 ,
        SICK-FAM                , 1, ,  , 3.000E+000, 5.000E-002, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 3.000E+000, 5.000E-002, Y, RANDOM                 ,
        TRNS-2                  , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        TRNSPRT                 , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000,  ,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
        WORK                    , 1, ,  , 0.000E+000, 1.000E+000, 0.000E+000, 0.000E+000,
        0.000E+000, I,  , 0.000E+000, 1.000E+000,  , RANDOM                 ,
```

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) 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 24 shows the sample database sequence cut sets.

Below are the available methods for generating sequence cut sets.

Solve Sequence Logic Method
Use the Solve form to generate sequence cut sets. 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(s) to solve.

2. Right click and choose the Solve option from the popup menu to bring up Solve Cut Sets 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.

3. Click the Cut Sets button to view the cut sets and, if desired, create a report.

Table 24. 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 25 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 when editing the event tree. (Refer to section 4.3.2) 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.

Table 25. 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 Post-processing Actions

This section discusses the addition of post-processing or recovery actions to sequence cut sets. PRAs often have post-processing actions applied to a specific scenario of events that may occur in a sequence or fault tree cut set. These 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 post-processing action that will be applied to one sequence cut set. Post-processing actions or rules can be applied to fault tree cut sets using Fault Tree post processing and Project Fault Tree post processing rules. 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 an event tree from the Event Tree listing.

2. Right click to invoke a pop up menu and choose the Edit Post-processing Rules option. (Depending on the desired applicable scope of the rule, the project Event Tree Post Processing Rules could also be edited.)

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

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

4.9 Analyzing Uncertainty

Uncertainty of the cut set and end state results are 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 26). 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. Select a fault tree from the list.

2. Right click and choose “View Uncertainty” option from the popup menu.

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

4. Wait for the calculation to complete a graph is then displayed and results can be seen by pressing the Result Table button.

Table 26. 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 5, 1.032E-004, 1.018E-003, 2.577E-003, 9.309E-003, 4.912E-006, 1.228E-001, 5.489E-003, 7.906E+000, 1.032E+002, , , ,----, , 0,ALARM PERSONAL ,0, 2.400E+001, 2.007E-002,------E----,------E----, 5000, 4321,--, , 3 3, 2.759E-003, 1.198E-002, 1.950E-002, 5.977E-002, 4.544E-004, 6.530E-001, 2.731E-002, 7.855E+000, 1.219E+002, , , M,----, , 0,PERSONAL TRNS-2 ,0, 2.400E+001, 1.100E-005,------E----,------E----, 5000, 4321,--, , 1 2, 7.801E-007, 5.441E-006, 1.039E-005, 3.676E-005, 8.671E-008, 3.369E-004, 1.549E-005, 5.348E+000, 6.211E+001, , , M,----, , 0,TRNS-2 TRNSPRT ,0, 2.400E+001, 1.485E-005,------E----,------E----, 5000, 4321,--, , 1 2, 1.053E-006, 7.345E-006, 1.403E-005, 4.963E-005, 1.171E-007, 4.548E-004, 2.092E-005, 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 is the same as 4.9.1 except select sequence(s).

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 is the same as the steps in 4.9.1 except have the desired end state(s) selected.

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.

To generate sequence group or project, just select the desired sequences or all the event trees to solve the entire project and follow the steps in 4.9.1

5. OTHER DATA LOADING METHODS

While MAR-D (e.g., Load/Extract) is the primary mechanism for loading data files into SAPHIRE 8, there are two other methods available:

- Loading an “accident sequence matrix” file
- Using the “project integrate” module

5.1 Loading Data via an Accident Sequence Matrix

Development of models which make use of similar event tree structures, such as external events models, was made user-friendly by providing a method to import event tree logic and sequence flag sets. This method allows a model developer to specify the accident sequence information in a spreadsheet. An example input deck is shown in Figure 11.

```
* Model Type Name,Model Type ID,IE,IE Freq.,IE Desc.,Event Tree Name,Event Tree Desc.,X-fer to,End State Substitution,Flag Set Name,Flag Set Desc.,Flag Set Setting(s),,,,

FIRE,FIR,IE-FRI-1,4.84E-05,Fire Scenario 1,FIRE1,"Demo Fire scenario 1",LOSP,,FIRE_FS_1,Flag Set for Fire Scenario 1,E-CV-A

FIRE,FIR,IE-FRI-2,2.67E-04,Fire Scenario 2,FIRE2,"Demo Fire scenario 2",LOSP,,FIRE_FS_2,Flag Set for Fire Scenario 2,E-CV-B

FIRE,FIR,IE-FRI-3,2.58E-04,Fire Scenario 3,FIRE3,"Demo Fire scenario 3",LOSP,,FIRE_FS_3,Flag Set for Fire Scenario 3,E-MOV-1

FLOOD,FLI,IE-FLOOD-1,4.84E-05,Flood Scenario 1,FLOOD1,"Demo Flood scenario 1",LOSP,,FLOOD_FS_1,Flag Set for Flood Scenario 1,E-CV-A

FLOOD,FLI,IE-FLOOD-2,2.67E-04,Flood Scenario 2,FLOOD2,"Demo Flood scenario 2",LOSP,,FLOOD_FS_2,Flag Set for Flood Scenario 2,E-CV-B

FLOOD,FLI,IE-FLOOD-3,2.58E-04,Flood Scenario 3,FLOOD3,"Demo Flood scenario 3",LOSP,,FLOOD_FS_3,Flag Set for Flood Scenario 3,E-MOV-1

SEISMIC1,EQ1,IE-EQ-BIN-1,1.036E-03,Seismic Scenario 1,SEISMIC1,"Demo Seismic scenario 1",LOSP,,SEISMIC_FS_1,Flag Set for Seismic Scenario 1,E-CV-A

SEISMIC2,EQ2,IE-EQ-BIN-2,2.560E-05,Seismic Scenario 2,SEISMIC2,"Demo Seismic scenario 2",LOSP,,SEISMIC_FS_2,Flag Set for Seismic Scenario 2,E-CV-B

SEISMIC3,EQ3,IE-EQ-BIN-3,8.740E-06,Seismic Scenario 3,SEISMIC3,"Demo Seismic scenario 3",LOSP,,SEISMIC_FS_3,Flag Set for Seismic Scenario 3,E-MOV-1
```

Figure 11. Information for an accident sequence matrix file

In SAPHIRE, the accident sequence matrix file is loaded via **Project → Tools → Add Accident Matrix** from the main menu. Click the open button and locate the accident matrix file desired. Click the Add button and the process begins. Information will appear in a text box as the accident matrix file is being processed. Any errors or warnings will also be displayed in this box. When done, click the **OK** button. The project will now be populated with the event trees and basic events described in the accident sequence matrix file.

This capability builds upon an existing model – it does not build a new project from scratch. In the above example an internal initiating events model is extended to include external initiating events.

SAPHIRE will create the following from the above line items:

Model Type: (Model Type Name, Model Type ID)

Initiating Event: (IE, IE Freq., IE Desc.)

Event Tree: (Event Tree Name, Event Tree Desc.)

End State Substitution (created in the linkage rules)

Flag Set: (Flag Set Name, Flag Set Desc., Flag Set Setting(s)), where the "Flag Set Settings" is the basic event name which will be set to True (the ASM does not currently set the basic event to False or Ignore).

The following must exist:

The transfer tree: (X-fer to)

5.2 Integrate Project Utility

Integrate Project provides comparison of the current project with another project and transfer of selected items from the compared project to the current one.

To integrate one project into another:

- Open main menu **File → Integrate Project**
- Choose the project either from a list of recently opened projects or browse for a file using the Browse button (see Figure 12).

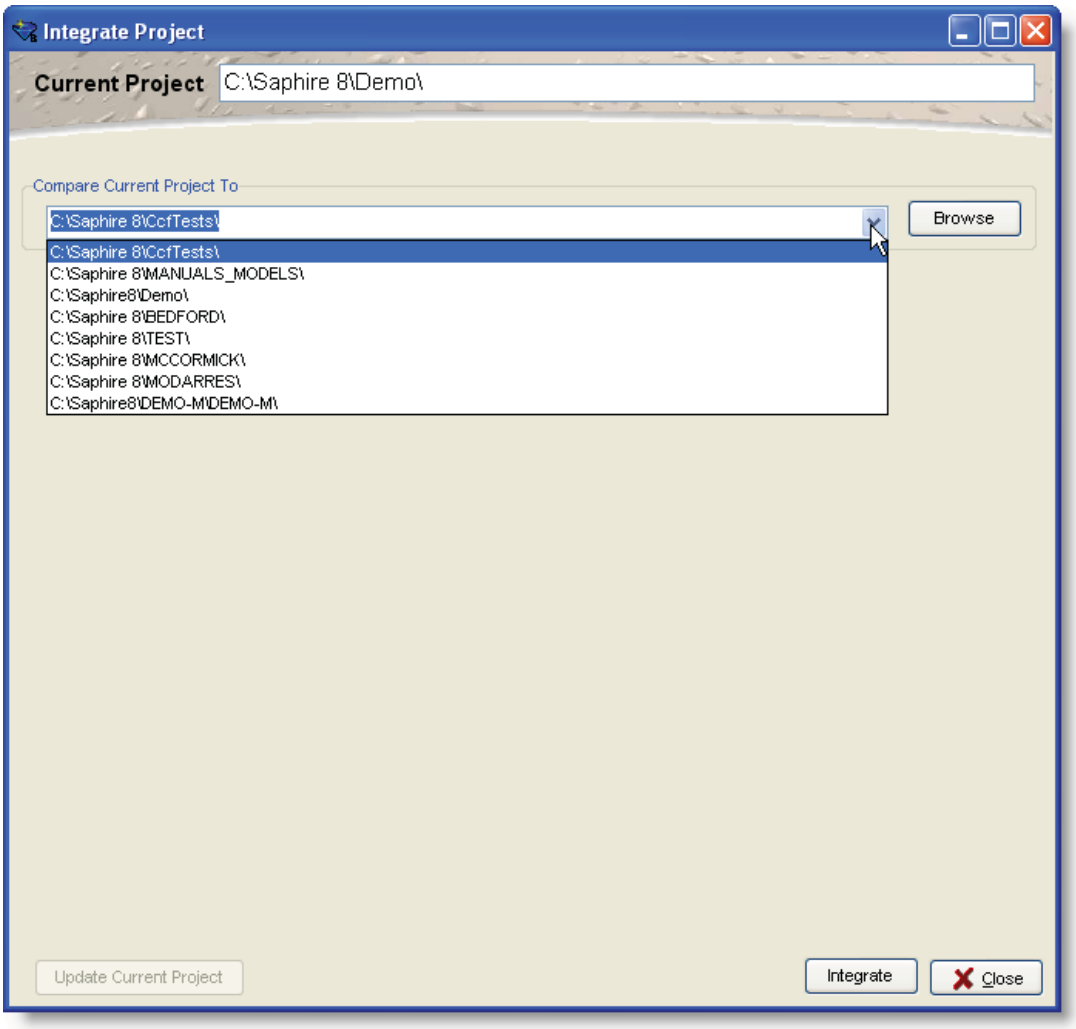
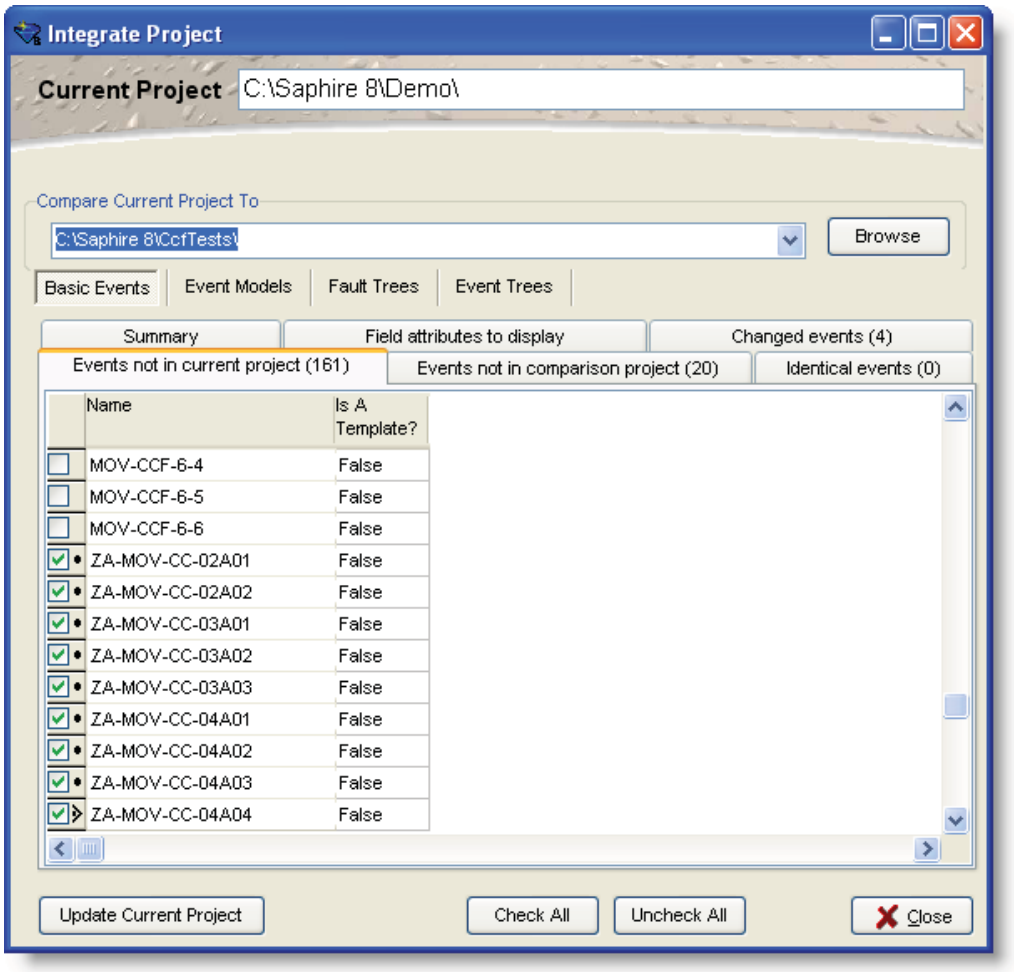


Figure 12. First step of the Project Integration option

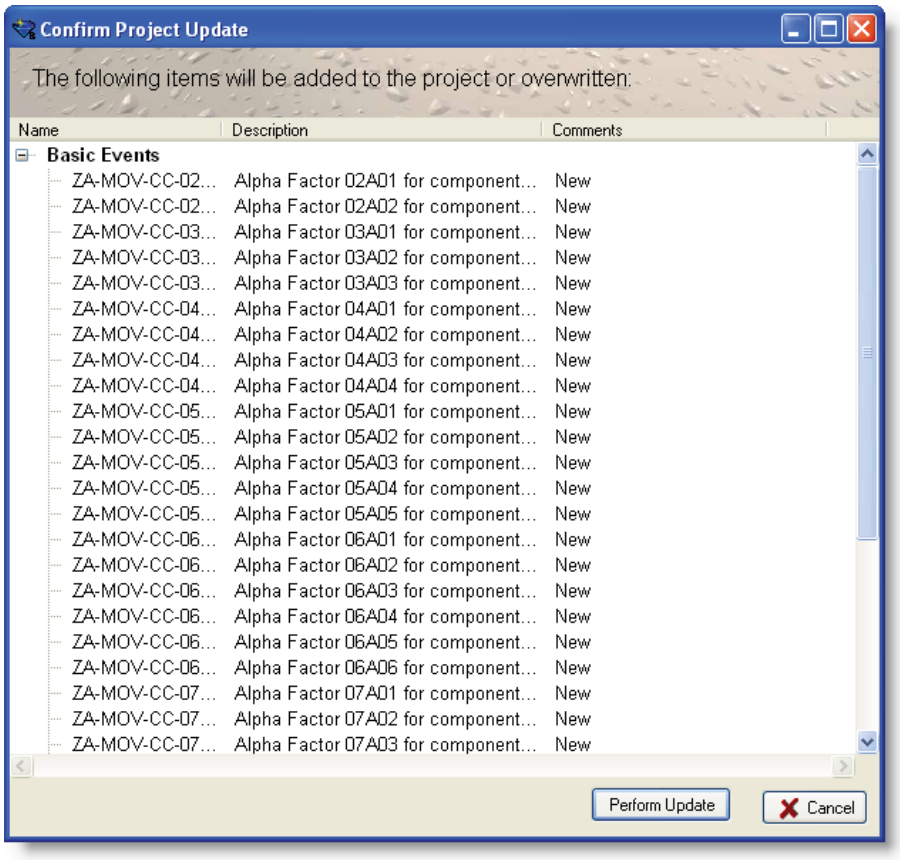
Click on the Integrate button to compare the files between the project just selected and the project that is already open. Depending on the amount of information stored inside the two projects, this step may take several seconds.



The report will show the differences between the two projects. To transfer items from the compared project to the current project:

- As shown above, place check marks in the boxes next to Basic Events, Event Models, Fault Trees, and Event Trees desired for import into the current project.

- Click on **Update Current Project** to start the import and a verification window will open.



After reviewing the items to be added either click on **Perform Update** to add them to the current project or **Cancel** to exit.

Appendix A

Procedures for Database Loading

A. Procedures for Loading or Extracting Data

Extracting Files

The user may extract flat files from the interactive database by using the load/extract form. A .MARD file and a corresponding sub-folder in the user specified directory is created/needed. Associated files needed for load/extract are contained in the sub-folder.

To **extract** a flat file:

1. Select the items desired for exporting from the left side lists. (If exporting section of the entire project and not individual items, skip this step.)

2. Select the (**File → Load/Extract**) main menu option, as shown in Figure A-1. The Load and Extract Data dialog will appear, as shown in Figure A-2.

3. Select the Extract tab at the top left of the dialog.

4. If you want to export sections of the entire project, for example the logic of all the fault trees, select the radio button option “All Items”.

5. Check the sections of the project you want to export. For example, if you want to export the fault tree logic and all the basic event information, under “Fault Tree” check the sub category “Logic” and check the main category “Basic Events”. (See figure A-1) Else if exporting the entire project, click the button on the bottom of the form “Mark”, to check all the items.

6. Enter the name and location to save the export files or click the “Save as” button to browse for a location.

7. Click the Process Button.

8. If any errors occurred it will be stated and you can view those errors by pressing the Errors button on the bottom of the form.

9. To accept the default file name, click the OK button. The user may first rename the file, but the extension should not be changed.

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

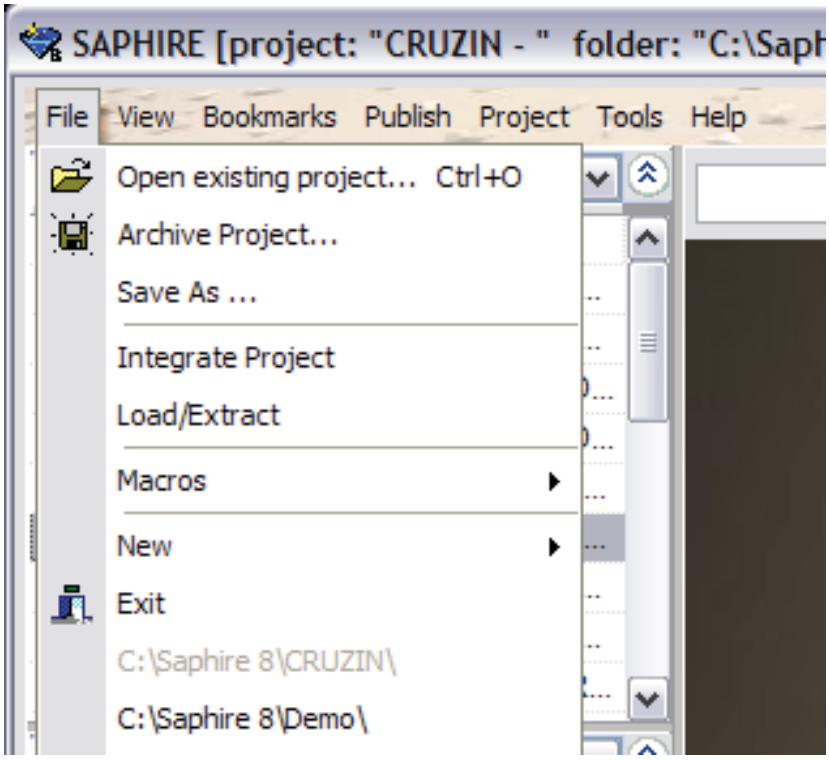


Figure A-1 Load and Extract menu option

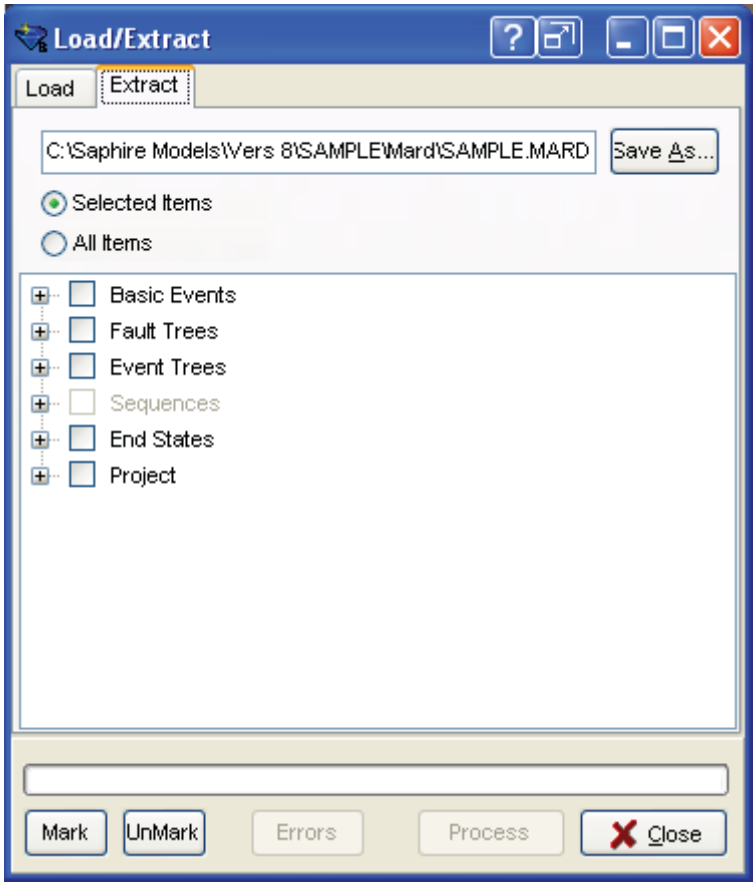


Figure A-2 Extract menu option

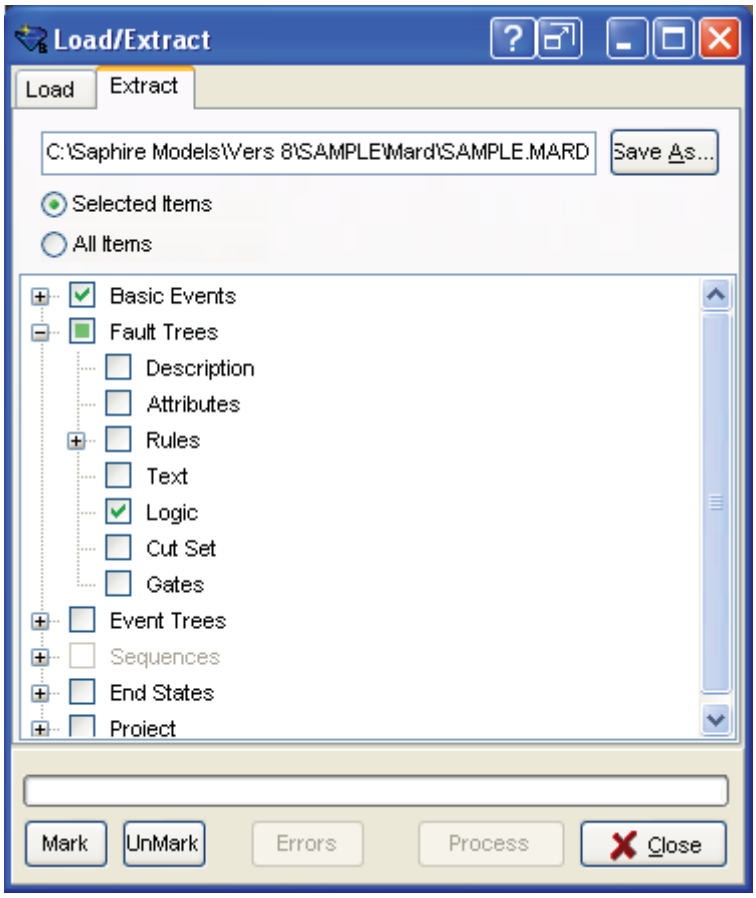


Figure A-3 Example of a Basic Events extraction dialog

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 (note that information loaded this way goes into the General Analysis interface). Most extracted files from previous versions of SAPHIRE will be able to be loaded into SAPHIRE 8 version, however possible incorrect default values could be used if old version files are missing needed data.

To load a flat file from a previous version of SAPHIRE:

1. Select the (**File → Load/Extract**) main menu option, as shown in Figure A-1. The Load and Extract Data dialog will appear, as shown in Figure A-2.

2. Check “Allow old formats option.

3. Select “Open” and browse for the desired data file, as shown in figure A-5. (Change “Files of type” to “All files” in order to see all possible files.)

4. Be default only the correct import option will be allowed so just click the “Process” button.

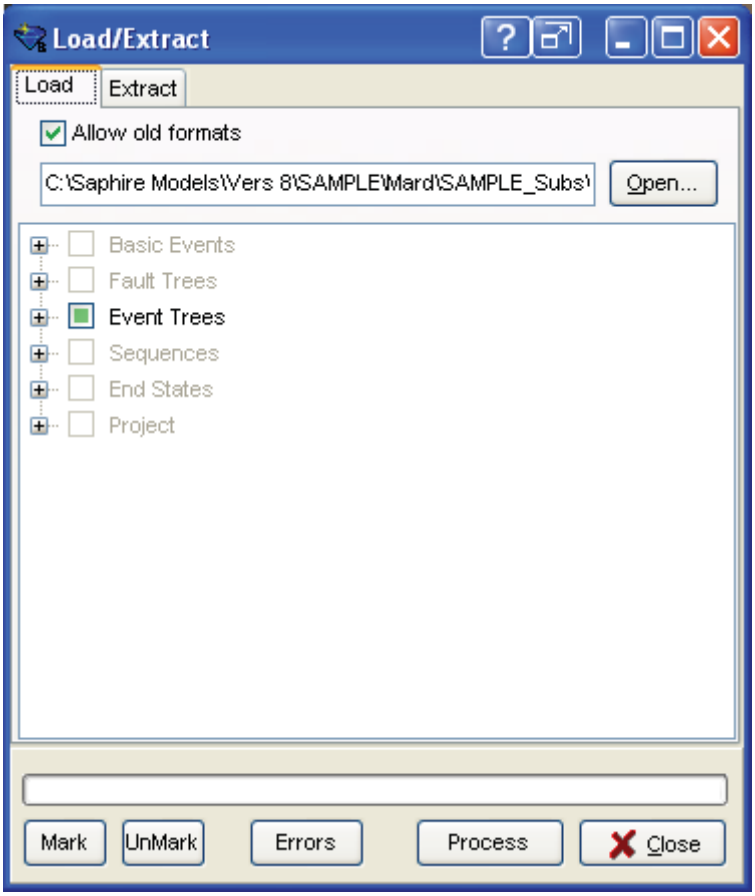


Figure A-4 Load menu option

To load a flat files from a SAPHIRE 8 export:

1. Select the (File → Load/Extract) main menu option, as shown in Figure A-1. The Load and Extract Data dialog will appear, as shown in Figure A-2.

2. Select “Open” and browse for the (.MARD) file.

3. All available data for import from the selected (.MARD) file is distinguished by having enabled check boxes. Select the data desired for importing (see Figure A-7).

4. Click the Process button. Any sections that had problems with importing can be viewed by pressing the Errors button.

Note – Even if an entire project is exported, when importing it, all items may not be available. This is because the project that was exported had no data in that area.

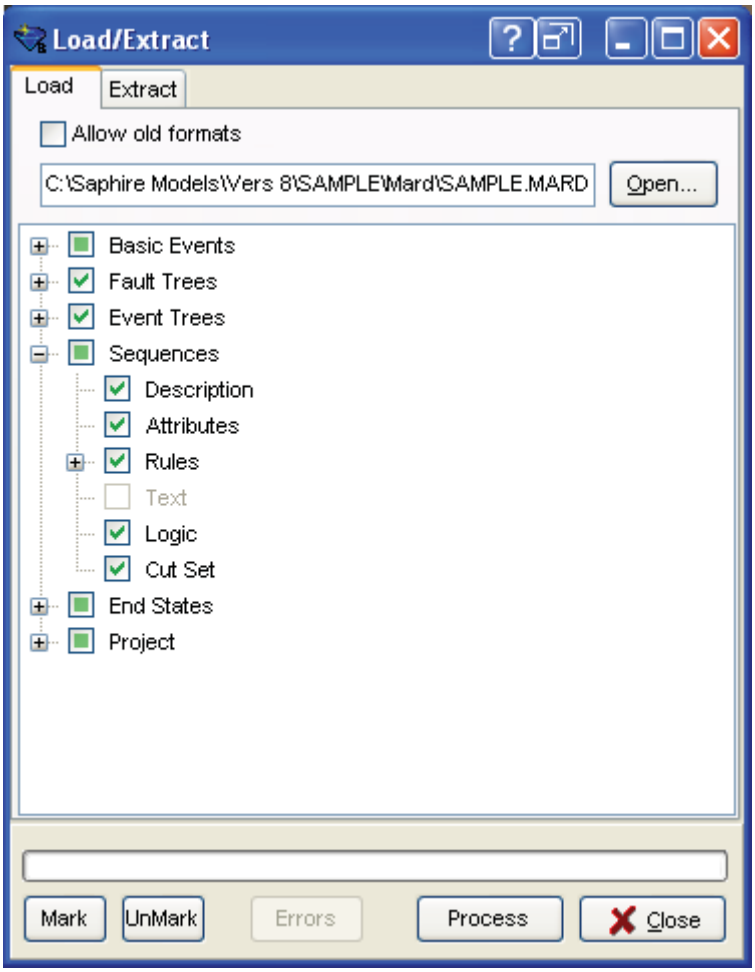


Figure A-7 Version 6 Load data prompt

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 October 2010. SAPHIRE 8 uses a slightly different method than previous versions, for loading/extracting MAR-D files. To make it easier to load/extract groups of data a (.MARD) file is used. This (.MARD) file contains a reference to all other necessary files in a subdirectory created in the extract or needed for the load. This makes loading easier by just having to open the one (.MARD) file.

Except where noted, file formats are the same for SAPHIRE versions 7 and 8. The primary version differences occur for the event tree logic and basic event information. These changes were made to allow basic events to have various Model and Phase types and event trees to have different phases.

The file formats are backward compatible: version 7 files can be successfully loaded into version 8. It is not valid to try and load version 8 files into version 7, due to the presence of subtle format and content changes some files will fail.

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.

Contents list for Appendix B

Project Information B-4

Basic Event Information B-6

Fault Tree Information B-12

Event Tree Information B-16

End State Information B-21

Sequence Information B-23

Gate Information B-27

Change Set Information B-27

Histogram Information B-30

Project Information

Project Names and Descriptions

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

Project-Level Fault Tree (system) 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

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

File Name:

xxxxxx.BED

File Format:

project =

name,description[,A],analysis type, phase type

. . . , . . .

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
analysis type	24 character	Analysis Type for this basic event (blank if phase type is present)
phase type	24 character	Phase Type for this basic event (blank if analysis type is present)

Note : The (name, analysis type) and (name, phase type) combination must be unique. For example you can have several events called ALARM, as long as each has a different analysis or phase.

Basic Event Rate Information

File Name:

xxxxxx.BEI

File Format:

project =

name, Fdt, udC, udT, udValue, prob, lambda, tau, mission, Init, Flag, udV2, Calc. Prob, Freq, Analysis Type, Phase Type

. . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . . , . . .

where

Project	24 character	Project name
Name	24 character	Basic event name
Fdt	1 character	Failure 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)

UdValue	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
Calc. Prob		
Freq		
Analysis Type	24 character	Analysis Type for this basic event (blank if phase type is present)
Phase Type	24 character	Phase Type for this basic event (blank if analysis type is present)

General Rules:

1. The name field is mandatory.
2. Note that the “analysis type” field is now referred to in SAPHIRE 8 as Model Type.

Basic Event Attribute Codes

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
train	3 character	Train identifier
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.
Is Template	1character	Flag 'Y' if item is a template blank if not
Template Name	24 character	Name of template if it uses a template blank if it doesn't use a template.
Use Template Flags,…,	32- 1 character flags	32 comma separated values indicating either values to use from the template, or what values available if the event is a template. (Currently only 24 of them are in use.)

1	Failure Model
2	Probability
3	Lambda
4	Tau
5	Mission Time
6	Prob – uncert distribution type
7	Prob – uncert value1
8	Prob – unsert vaule2
9	Lambda – uncert distribution type
10	Lambda – uncert value1
11	Lambda – uncert value2
12	Tau – uncert distribution type
13	Tau – uncert value1
14	Tau – uncert value2
15	Mission Time – uncert distribution type
16	Mission Time – uncert value1
17	Mission Time – uncert value2
18	Correlation class
19	Process Flag
20	Frequency Units
21	Transform Level
22	Transform Type
23	Transform Events
24	Model Type

General Rules:

1. The name field is mandatory.

Basic Event Transformations

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

In SAPHIRE version 8, 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

Basic Event Category

File Name:

xxxxxx.BECat

File Format:

project =

BE-name1, Category Name, Category Level (1-9), BE1 Category Lable

BE-name2, Category Name, Category Level (1-9), BE2 Category Lable

BE-name3, Category Name, Category Level (1-9), BE3 Category Lable

…

Basic Event Grade

File Name:

xxxxxx.BEG

File Format:

BE-name1, Grade

BE-name2, Grade

BE-name3, Grade

…

where

BE-namei	24 character	Basic event name
Grade	1 character	Type of basic event “blank” = regular basic event S = system generated event V = “virtual” event

Fault Tree Information

Fault Tree Names and Descriptions

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

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

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

Event Tree Information

Event Tree Names and Descriptions

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.

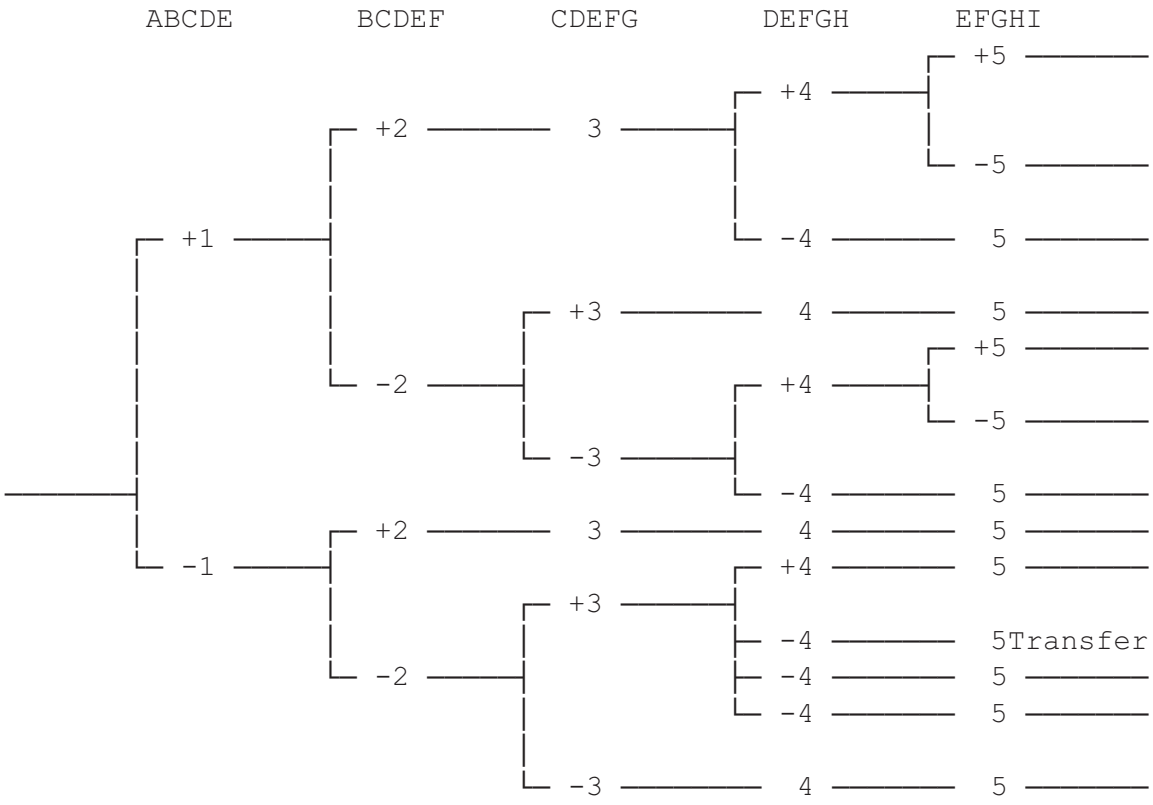
File Name:

xxxxxx.ETG

File Format:

See file format for the Event Tree Logic

SAMPLE GRAPHICAL EVENT TREE



File Name:

xxxxxx.ETL

File Format:

project, event tree, init event [,T] =

*Phases Marker___# phases defined___max count sequences__initial phase

^PHASES 1 13 1

^TOPS

*Top names space seperated

ABCDE BCDEF CDEFG DEFGH EFGHI

^LOGIC 1 // initial phase, following are offset

+1.0 +2.0 3.0 +4.0 +5.0
-5.0
-4.0 5.0
-2.0 +3.0 4.0 5.0
-3.0 +4.0 +5.0
-5.0
-4.0 5.0
-1.0 +2.0 3.0 4.0 5.0
-2.0 +3.0 +4.0 5.0
-4.0 5.0
-4.0 5.0
-4.0 5.0
-3.0 4.0 5.0

^SEQUENCES 0 // offset from initial phase

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

^ENDSEQUENCES     //Now postprocess end names
^TOPDESC
 ""
!
 ""
!
 ""
!
 ""
!
 ""
!
 ""
!
^TEXT
^PARMS
^EOS
```

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

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

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 =

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

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

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	5th 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

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

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

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

File Name:

xxxxxx.CSI

File Format:

project, change=

^PROBABILITY

eventname,calc,udT,prob,lambda,tau,udV,udC,mission,init,udV2

^CLASS

eventname,group,compType,compId,system,location,failMode,train,init,att1
,..att16

* CLASS PROBABILITY HEADER

calcType,udT,prob,lambda,tau,udV,udC,mission,init,udV2

^EOS

project,change2=

where

change	24 character	change set name
name	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
udV2	Floating point	Uncertainty distribution value 2
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

File Name:

xxxxxx.CSA

File Format:

project=

name,altName, type

…,…
where

project	24 character	Project name
name	24 character	Change set primary name
altName	24 character	Change set alternate name
type	1 character	C = change set F = flag set
subtype	1 character	Blank – default (no sub type) S = sensitivity P = permanent (always used during a Workspace analysis) – not currently used.

Histograms

Histogram Description

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.HII

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

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

Appendix C

MAR-D Files for Sample Database

C. MAR-D Files for Sample Database

SAPHIRE Version 8 MAR-D formats for the Sample Database are presented.

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.

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

Basic Event Rate Information File (.BEI)

```
SAMPLE                  =
*  Name         ,FdT,UdC,UdT, UdValue , Prob      , Lambda    , Tau       ,
      Mission, Init, PF, UdValue2, Calc. Prob, Freq, Analysis Type
      , Phase Type
ALARM          ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
ALM-BPF        ,1,    ,L, 3.000E+000, 9.000E-008,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
ALM-CPF        ,1,    ,L, 3.000E+000, 1.500E-002,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
ALM-FTS        ,1,    ,L, 1.000E+001, 5.500E-006,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
```

```
ALM-MECH      ,1,    ,L, 3.000E+000, 2.700E-008,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
ALM-SWT       ,1,    ,L, 1.000E+001, 2.700E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
MEDICINE      ,1,    ,L, 5.000E+000, 5.000E-001,+0.000E+000,
      +0.000E+000,+0.000E+000,R, ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
OTHER         ,1,    ,L, 1.000E+001, 8.100E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
PER-TRNS      ,1,    ,L, 5.000E+000, 5.500E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
PERSONAL      ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
PUB-TRNS      ,1,    ,L, 3.000E+000, 2.700E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
PUB-TRNS-LATE ,1,    ,L, 3.000E+000, 2.000E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
SICK          ,1,    ,L, 1.000E+001, 8.100E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
SICK-FAM      ,1,    ,L, 1.000E+001, 4.000E-003,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
TRNS-2        ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
TRNSPRT       ,1,    ,L, 1.000E+000, 1.000E+000,+0.000E+000,
      +0.000E+000,+0.000E+000, , ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
WORK          ,1,    ,L, 2.000E+000, 2.480E+002,+0.000E+000,
      +0.000E+000,+0.000E+000,I, ,+0.000E+000, 1.000E+000,  , RANDOM
      ,
```

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,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,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,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,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,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,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,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,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,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,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,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,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,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,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,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,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,
```

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 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 =
^WINVER2.1
^PHASES 1 5 1  //# phases defined, max count sequences, initial phase
PHASE_1 16155777 "Phase 1"
^TOPS
ALARM, PERSONAL, TRNSPRT
^LOGIC 1                  // initial phase, following are offset
 +1.0 +2.0 +3.0
           -3.0
      -2.0  3.0
 -1.0  2.0 +3.0
           -3.0
^SEQUENCES 0                          // offset from initial phase
N, Endstate, N, Sequence Name, N, Frequency, N, Extra,
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, , ,
^ENDSEQUENCES      //Now postprocess end names
^TOPDESC
 "Initiating event"
!
 "Alarm Failure"
!
 "Personal problems"
!
 "Personal and commercial"
 "transportat ion failure"
!
^TEXT
^PARMSDESPITCH 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 , Alternate Name
ALARM                   , OR   , ALARM
ALARM-0                 , AND  , ALARM-0
ALARM-1                 , OR   , ALARM-1
ALARM-2                 , OR   , ALARM-2
BD                      , TRAN , BD
BE                      , TRAN , BE
BF                      , TRAN , BF
BG                      , TRAN , BG
EJ                      , OR   , EJ
PERSONAL                , OR   , PERSONAL
TRNS-2                  , OR   , TRNS-2
TRNSPRT                 , OR   , TRNSPRT
```

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-7039, Vol. 7
INL/EXT-09-17015

2. TITLE AND SUBTITLE
Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8

Volume 7 Data Loading

3. DATE REPORT PUBLISHED

MONTH	YEAR
MARCH	2011

4. FIN OR GRANT NUMBER
N6423

5. AUTHOR(S)
K. J. Kvarfordt, S. T. Wood, C. L. Smith, S. R. Prescott

6. TYPE OF REPORT
Technical

7. PERIOD COVERED (Inclusive Dates)

8. PERFORMING ORGANIZATION - 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. 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 8. 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 8, software, data loading, load, extract, MAR-D

13. AVAILABILITY STATEMENT
Unlimited

14. SECURITY CLASSIFICATION
(This page)
Unclassified
(This report)
Unclassified

15. NUMBER OF PAGES

16. PRICE