DOCUMENTATION FOR THE FRAMES-HWIR TECHNOLOGY SOFTWARE SYSTEM, VOLUME 8: SPECIFICATIONS

Project Officer and Technical Direction:	Mr. Gerard F. Laniak U.S. Environmental Protection Agency Office of Research and Development National Environmental Research Laboratory Athens, Georgia 30605
Prepared by:	Pacific Northwest National Laboratory Battelle Boulevard, P.O. Box 999 Richland, Washington 99352 Under EPA Reference Number DW89937333-01-0

U.S. Environmental Protection Agency
Office of Research and Development
Athens, Georgia 30605

October 1999

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

This document was printed on recycled paper.
(9/97)

ACKNOWLEDGMENTS

A number of individuals have been involved with this effort. Mr. Gerard F. Laniak of the U.S. Environmental Protection Agency (EPA), Office of Research and Development, National Environmental Research Laboratory, Athens, Georgia, provided the overall technical direction and review throughout this work. This report was prepared by the Pacific Northwest National Laboratory[1] (PNNL) staff of K. J. Castleton, M. A. Pelton, B. L. Hoopes, J. W. Buck, G. M. Gelston, R. E. Lundgren, G. Whelan, R. Y. Taira. Additional PNNL staff supporting this effort include Wayne Cosby, Nancy Foote, Kristin Manke, Jill Pospical, Debbie Schulz, and Barbara Wilson. Useful inputs were provided by many U.S. EPA individuals working on the Hazardous Waste Identification Rule, including Messrs. Barnes Johnson, Stephen Kroner, and David Cozzie, and Drs. David Brown, Robert Ambrose, Zubair Saleem, Donna Schwede, and Sharon LeDuc, among many others.

[1]Operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

Summary

The U.S. Environmental Protection Agency (EPA) is developing a comprehensive environmental exposure and risk analysis software system to assess exposures and risks relevant to the Hazardous Waste Identification Rule (HWIR). The software system adapted to automate this assessment is the Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES), developed by the Pacific Northwest National Laboratory (PNNL). The FRAMES-HWIR Technology Software System comprises a number of components, including six processors and a number of databases. This report documents specifications for each of these components and is intended to assist those who are developing system components.

A number of organizations teamed in the development of the modified system. To facilitate system operations, the team agreed to follow a common set of specifications, which are described in this document. The first draft of this document was sent to EPA on February 27, 1998. A draft addressing EPA's comments was provided to EPA on April 3, 1998. Shortly thereafter, EPA provided the document to all team members for their use in developing their respective system components. Decisions made between April and early October necessitated a revision in October 1998. In addition, the October version reorganized the material to aid developers in quickly identifying which portions applied to their components. Figure 1.1 in the Introduction shows the location and flow of data through these components. This reorganization grouped the material as follows:

If You Are Developing the	**Read**
Chemical properties database	Chapter 6.0
Chemical Properties Processor	Chapter 6.0
Computational Optimization Processor	Chapter 7.0
Distribution Statistics Processor	Chapter 3.0
Exit Level Processor I	Section 7.1
Exit Level Processor II	Section 7.2
Model error statistics database	Section 5.2
Module within the Multimedia Multipathway Simulation Processor	Chapter 8.0
Multimedia Multipathway Simulation Processor	Chapter 8.0
National distribution statistics database	Section 3.2
National statistics database	Section 4.2
Regional distribution statistics database	Section 3.2
Regional statistics database	Section 4.2
Risk Visualization Processor	Chapter 10.0
Site delineation database	Chapter 2.0

If You Are Developing the	Read
Site-based database	Section 4.2
Site Definition Processor	Chapter 5.0
Site Layout Processor	Chapter 4.0
Site survey database	Section 5.2
Static national database	Section 4.2
Static regional database	Section 4.2
System User Interface	Chapter 11.0

Acronyms and Abbreviations

ASCII	American Standard Code for Information Interchange
COP	Computational Optimization Processor
CPP	Chemical Properties Processor
C_w	concentration of chemical in waste stream
DLL	Dynamic Link Library
DOS	disc operating system
DSP	Distribution Statistics Processor
ELP I	Exit Level Processor I
ELP II	Exit Level Processor II
EOF	End of File
EPA	U.S. Environmental Protection Agency
FRAMES	Framework for Risk Analysis in Multimedia Environmental Systems
GRF	Global Results Files
HQ	Hazard Quotient
HWIR	Hazardous Waste Identification Rule
EPA-CMTP	Environmental Protection Agency's Composite Model for Leachage Migration with Transformation Products
EXAMS	Exposure Analysis Modeling System
HWIR.DLL	HWIR Input/Output Dynamic Link Library
HWIRMC.DLL	HWIR Monte Carlo Dynamic Link Library
ISC-ST	Industrial Source Complex-Short Term
MET	Meteorological
MMSP	Multimedia Multipathway Simulation Processor
OCRWM	Office of Civilian Radioactive Waste Management
PNNL	Pacific Northwest National Laboratory
PSOF	Protective Summary Output File
RAM	random access memory
RSOF	Risk Summary Output File
RVP	Risk Visualization Processor
SDP	Site Definition Processor
SLP	Site Layout Processor
SSF	Site Simulation Files
SUI	System User Interface
UTM	Universal Trans-Mercator

Contents

Figures

Tables

1.0 Introduction

The U.S. Environmental Protection Agency (EPA) is developing a comprehensive environmental exposure and risk analysis software system for agency-wide application. The software system is being prototyped (i.e., initial design and implementation) for application to the technical assessment of exposures and risks relevant to the Hazardous Waste Identification Rule (HWIR). The HWIR is designed to determine quantitative criteria for allowing a specific class of industrial waste streams to no longer require disposal as a hazardous waste (i.e., “exit” Subtitle C) and to instead be disposed of in Industrial Subtitle D facilities. Hazardous waste constituents with values less than these exit criteria levels would be reclassified as nonhazardous wastes under the Resource Conservation and Recovery Act.

The software system adapted to automate this assessment is the Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES), developed by the Pacific Northwest National Laboratory (PNNL). The FRAMES-HWIR Technology Software System consists of a series of components within a system framework (Figure 1.1). The process being used to develop the FRAMES-HWIR Technology Software System includes steps for requirements analysis, design, specification, and development, with testing constituting a critical portion of each step. This document covers the specification step of the process.

This document is designed to assist those who are developing components for the FRAMES-HWIR Technology Software System. Module developers are defined as those who are developing modules for the Multimedia Multipathway Simulation Processor (MMSP), recognizing that those modules themselves constitute processors. These modules include source modules (aerated tank, land application unit, landfill, surface impoundment, and waste pile), environmental fate and transport modules (air, watershed, waterbody network, aquifer, and vadose zone), foodchain modules (farm, terrestrial, and aquatic), and exposure and risk modules (human and ecological). Processor developers, on the other hand, are those developing one of the processors of the FRAMES-HWIR Technology Software System: the Distribution Statistics Processor (DSP), the Site Layout Processor (SLP), the Site Definition Processor (SDP), the Computational Optimization Processor (COP), the MMSP, the Exit Level Processors (ELP), or the Risk Visualization Processor (RVP). All modules and processors interact with a System User Interface (SUI). In addition, the system accesses a number of databases.

As shown in Figure 1.1, these components must work together and with the SUI for the FRAMES-HWIR Technology Software System to operate effectively. Key to this synergy is the passing of data among databases, processors, modules, and the SUI. To ensure that data are passed appropriately, all processors, modules, and interfaces must meet the specifications outlined in this document. Each of the PNNL-developed components are described in companion documents as listed in the reference list; the system itself is documented in a summary report entitled, *Overview of the FRAMES-HWIR Technology Software System*. The section immediately following this paragraph provides a brief overview of the system, as adapted from the summary report.

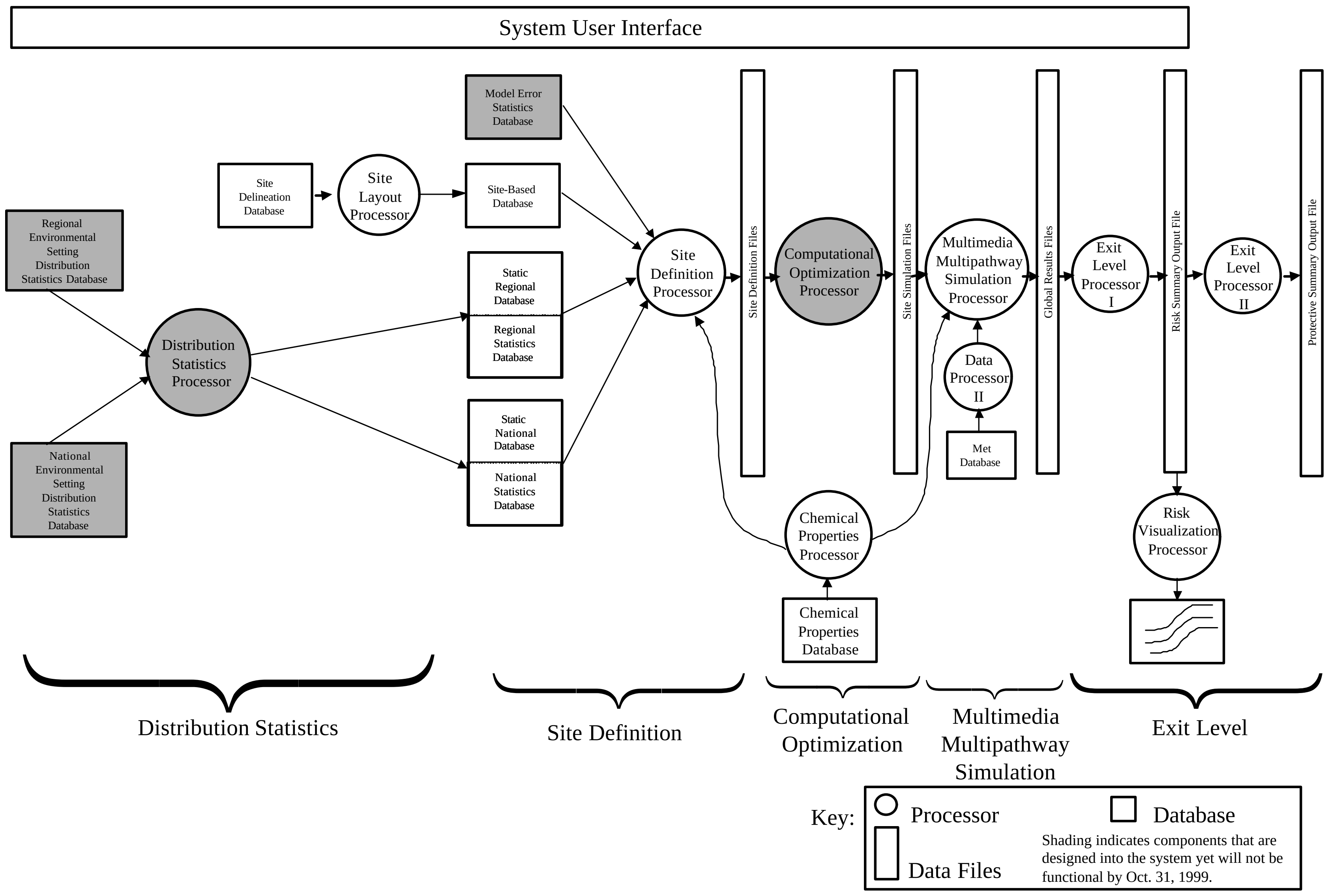
System User Interface
Model Error Statistics Database
Site Delineation Database
Site Layout Processor
Site-Based Database
Regional Environmental Setting Distribution Statistics Database
Distribution Statistics Processor
Static Regional Database
Regional Statistics Database
National Environmental Setting Distribution Statistics Database
Static National Database
National Statistics Database
Site Definition Processor
Site Definition Files
Computational Optimization Processor
Site Simulation Files
Multimedia Multipathway Simulation Processor
Data Processor II
Met Database
Global Results Files
Exit Level Processor I
Risk Summary Output File
Exit Level Processor II
Protective Summary Output File
Chemical Properties Processor
Chemical Properties Database
Risk Visualization Processor
Distribution Statistics
Site Definition
Computational Optimization
Multimedia Multipathway Simulation
Exit Level
Key:
Processor
Database
Data Files
Shading indicates components that are designed into the system yet will not be functional by Oct. 31, 1999.

Figure 1.1 Overview of the FRAMES-HWIR Technology Software System

1.1 Overview of the FRAMES-HWIR Technology Software System

The FRAMES-HWIR Technology Software System consists of a System User Interface and a series of processors within a system framework. The interface and processors work together to answer the question, “At what concentration level before disposal would a particular chemical be nonhazardous to humans and the environment near a particular facility?” A variety of factors affect how this question is answered, for example:

- The number of concentration levels for a particular chemical. The HWIR Assessment Strategy considers a range of concentration levels for each chemical based on a number of factors including the process that generated that chemical, the chemical properties, and the level of known toxicity.
- The number of chemicals to be evaluated. The Assessment Strategy evaluates all chemicals currently listed as hazardous, including 160 organic chemicals and 20 metals.
- The definition of nonhazardous. The EPA will define nonhazardous once risks have been assessed to human and ecological receptors.
- The varying environmental conditions among sites. The Assessment Strategy will evaluate typical environmental conditions found across the nation, ranging from desert environments to mountainous environments to swamp-like environments.
- The different types of waste management systems. The Assessment Strategy will evaluate contamination that could emanate from aerated tanks, land application units, landfills, surface impoundments, and waste piles.
- The types of receptors. The Assessment Strategy will consider exposures and risks to community residents and their children, home gardeners and their children, farmers and their children, and recreational fishers and their children within a 2-kilometer radius of the facility as well as plants and animals typically found within that same radius of a particular facility. Endangered species will be included.

The user provides input through the SUI to select the site to be evaluated, which types of sources to consider, which chemicals to consider, waste concentration levels, and the location of files containing key information. This information is used as input to characterize a site and its potential impacts to surrounding receptors. In this context, site refers to a waste management facility, which might contain one or more waste management units, such as aerated tanks, landfills, land application units, surface impoundments, or waste piles. Whenever possible, data collected at actual waste management facilities are used to fill the databases from which site information is defined. However, when such data are not available, the SDP selects appropriate information from either regional or national data of the same type using, in part, statistical information developed by the Distribution Statistics processor. For example, if a site in Georgia were missing data related to rainfall, the SDP would select a distribution of regional rainfall data or, if that were not available, a distribution of national rainfall data to provide the necessary information.

The system collects data from databases specified by the user to create a full definition of a particular site. For example, site delineation data collected on a gridded system is processed through the SLP to provide input to the site-based database. Chemical properties data are processed by the Chemical Properties Processor (CPP) to provide a consistent set of data to computational models. All the gathered and processed information is used by a group of modules within the MMSP to simulate source releases, fate and transport of contamination through the environment, and exposure and risk impacts to specified receptors. Each module in the system requires specific parameters, which are provided through a series of Site Simulation Files (SSF). In trying to provide all the data modules needed, these files may contain duplicate information. The data may also indicate that a particular site is not appropriate to run (for example, when data conflicts or when a similar site has recently been assessed). Such conditions could result in a large amount of unnecessary data being passed through the system, slowing performance. Therefore, before risks are assessed, the COP reviews the data to optimize the data set and ensure smooth performance.

The MMSP then assesses risks through a complex modeling protocol that looks at release of contaminants in a variety of ways; transport of those contaminants through the environment; exposure of humans, animals, and plants; and the resulting risks or hazards posed by such exposures. For each particular site, the appropriate models are chosen for implementation. The ELP I then tabulates the calculated risks for visualization in the RVP and ultimate processing by the ELP II to allow EPA to determine what chemical levels will protect human and ecological health.

1.2 Overview of this Report

The following sections of this document describe how the data are gathered and processed across the system:

- Section 2.0 describes the input and output of the HWIR Input/Output Dynamic Link Library (HWIRIO.DLL), which coordinates the flow of information to and from all of the data groups and modules in the system. This section also describes error and warning file formats.
- Section 3.0 describes the DSP and its inputs: the national and regional distribution statistics databases.
- Section 4.0 describes the SLP and the spatial delineation database.
- Section 5.0 describes the SDP and its inputs: the model error statistics database, the site-based database, the static regional database, the regional statistics database, the static national database, the national statistics database, and the site survey database.
- Section 6.0 describes the CPP and its input, the chemical properties database.
- Section 7.0 describes the COP and its inputs, the Site Simulation Files.
- Section 8.0 describes the MMSP and the behavior of the media modules within it. This chapter gives instructions on how to implement a module within the MMSP.
- Section 9.0 describes the ELP I and II,and the risk visualization processor.
- Section 10.0 describes the SUI, including the Message.ALL file format.
- Appendix A provides detailed descriptions of the data groups used to categorize data in the MMSP.
- Appendix B provides examples of coding.
- Appendix C provides site layout data group details.
- Appendix D provides techniques for connecting between modules.

2.0 Specifications for Facilitating Dynamic Link Libraries

The FRAMES-HWIR Technology Software System uses three dynamic link libraries that are described in this chapter: the HWIRIO.DLL, the HWIR Spatial and Temporal Integrator Dynamic Link Library (HWIRST.DLL), and the HWIR Monte Carlo Dynamic Link Library (HWIRMC.DLL). A dynamic link library (DLL) is a modular set of routines that comes with or can be added to a software system. Each DLL file has a ".dll" file name extension. DLL files are dynamically linked with the program that uses them during program execution instead of being compiled with the main program. The set of such files is somewhat comparable to the library routines provided with programming languages such as FORTRAN, C, and C++.

2.1 HWIR Input/Output Dynamic Link Library (HWIRIO.DLL)

The HWIRIO.DLL supplies the interface between the data files and the program codes, providing a storage format that is both flexible and capable of expanding to meet system needs. Most of the processors of the FRAMES-HWIR Technology Software System, as well as the modules within the MMSP, use the HWIRIO.DLL.

The design of the HWIRIO.DLL encompasses several subroutines, including those for initialization, input, output, and finalization of HWIR data. The initialization routines are expected to be called before any input or output routines. Finalization routines are expected to be called after all input and output calls are complete. These subroutines are described in the following sections. Two prototypes (interfaces) are presented for each subroutine or function: a FORTRAN interface and a C++ prototype. The C++ prototypes are also compatible with versions of C that support prototypes.

Many function prototypes (interfaces) describe one or more functions. For example:

```
Function ReadReal[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6]) Real
```

is a condensed method for specifying seven separate functions. The seven functions are:

```
Function ReadReal(Datagroup,Variable,Units) Real
Function ReadReal1(Datagroup,Variable,Units,Index 1) Real
Function ReadReal2(Datagroup,Variable,Units,Index 1,Index 2) Real
Function ReadReal3(Datagroup,Variable,Units,Index 1,Index 2,Index 3) Real
Function ReadReal4(Datagroup,Variable,Units,Index 1,Index 2,Index 3,Index 4) Real
Function ReadReal5(Datagroup,Variable,Units,Index 1,Index 2,Index 3,Index 4, Index 5) Real
Function ReadReal6(Datagroup,Variable,Units,Index 1,Index 2,Index 3,Index 4, Index 5, Index 6) Real
```

A similar type of expression could be employed for subroutines. This notation method is used throughout this section. The FORTRAN and C++ types are described for each of the arguments as well.

Across all subroutines, the HWIRIO.DLL will have a data table that contains information on the name, type, and units for each data point. The name, type, and units in the HWIRIO.DLL are compared with the name, type, and units of the data point passed into the HWIRIO.DLL subroutine. If the names, type, and units do not match, an error occurs. The data table that is used by the HWIRIO.DLL will consist of a subset of the information provided from each module. The fields of this data table will be:

Data Group Name (String)
Variable Name (String)
Dimensionality (Integer)
Type (String)"Logical","Integer","Real","String"
Units (String)
Minimum (Double)
Maximum (Double)

The HWIRIO.DLL will test for consistency between the specifications and the calling program. For example, if a FORTRAN program module had the following line of code:

```
MyInts(i,j,k)=ReadInt3('Source_SSF','XYZ','cm/s',i,j,k)
```

the HWIRIO.DLL can make the following tests for consistency with the specified information in the data table:

1) Does the Data Group "Source_SSF" exist? If not, an error occurs.
2) Does the variable "XYZ" exist in "Source_SSF"? If not, an error occurs.
3) Are any of the indices negative or equal to zero? If so, an error occurs.
4) Do the specified units for "XYZ" in "Source_SSF" match the units passed by the calling program? If not, an error occurs.
5) Does the specified dimensionality of "XYZ" match the calling program's assumptions of dimensionality? If not, an error occurs.
6) Is any one of the indices larger than the maximum indices stored in the file? If so, a warning occurs and returns a 0.
7) Is the value stored in the data file outside the specified bounds for that variable? If so, an error occurs.
8) Does the specified type of the variable match the assumed type from the calling program? If not, an error occurs.

The data dictionary file for SSF and Global Result Files (GRFs) consists of three types of information: 1) the number of variable lines, 2) a field header line, and 3) the variable data description lines. A data dictionary file is a text file that contains the data about the data (sometimes referred to as meta data) that is contained in a SSF or GRF file. The first line of the dictionary file contains only the number of variables listed within the dictionary. The number of variables includes the one-line field header, along with the number of actual variables in the file (i.e., 35 variables would have 36 lines). The second line of the file contains the field header of the file, which is comprised of the following fields: "Code," "Dimensions," "Data Type," "Min," "Max," and "Units." Following the field head line are the N number of description lines, where N is defined by the number of varibles minus one specified in the first of the file. A description line consists of the "Code," "Dimensions," "Data Type," "Min," "Max," and "Units" specific for the variable. After the "Units" field are the "Description," "Dim1," "Dim2," "Dim3," "Dim4," "Dim5," and "Dim6" fields. These fields are optional. Following the field header is the information for the variable corresponding to the fields header for each variable. Each line contains all of the description data for one variable.

The “Code” field contains the code or name for each variable in the file. “Code” field entries are character strings limited to a maximum of twenty characters per field. The “Dimensions” field contains an integer denoting the number of dimensions associated with the variable. The number of dimensions of a variable must be within the range of zero to six. The “Data Type” field is a set of character strings that are limited to “float,” “integer,” “string,” or “logic.” Both the “Min” and “Max” fields contain floating-point numbers that are used to set an upper and lower limit upon the values assigned to the variable. The “Units” field of the dictionary contains a string of characters denoting the units of measure for the variable. While the units may be of any type, a maximum length of twenty characters limits the character string representing the units. The “Description” field is used to hold a textual description of the variable and is optional. The description field is represented by a string of characters that may be of any length including zero. The fields “Dim1” through “Dim6” are character strings used to hold a textual description of each dimension of the variable and are optional. The character strings used by the dimension description fields may be of any length including zero. The following is an example of a data dictionary file.

```
6,
"Code","Dimensions","DataType","Min","Max","Units",”Description”, “Dim 1", “Dim 2", “Dim 3", “Dim 4", “Dim 5", “Dim 6"
"VapDDep",2,"FLOAT",0,100000,"g/m2-s",
"VapWDep",2,"FLOAT",0,100000,"g/m2-s",
"PM10",2,"FLOAT",0,100000,"g/m2-s",
"ParDDep",2,"FLOAT",0,100000,"g/m2-s",
"ParWDep",2,"FLOAT",0,100000,"g/m2-s",
```

2.1.1 OpenGroups and CloseGroups Subroutines

The OpenGroups and CloseGroups subroutines will open and close all groups appropriate for a module or processor, including error and warning files. A group in the FRAMES-HWIR Technology Software System is a collection of related parameters that is used or produced by the system or its components. For example, for the CPP, the OpenGroups Subroutine opens the file CP.SSF, which will hold chemical information produced by the processor, as well as an error file and a warning file. The CloseGroups Subroutine would close these same groups after the CPP finished its activities. These routines make it possible for the component to detect errors that cause the program to terminate abnormally. A program terminates abnormally when an error occurs that was not detected and/or managed by the program. The error and warning files are opened before the component starts operating because the system assumes a component will terminate abnormally or its operation will result in a warning. Only when the component continues successfully to completion are the warning and error files deleted by the CloseGroups program. Additional information on warning and error files can be found in Section 2.1.4.

2.1.1.1 Subroutine OpenGroups

The OpenGroups subroutine is expected to be the first call in a program. This subroutine first retrieves the call arguments and creates warning and error files if they don’t exist or opens the existing warning and error files. If the warning and error files are opened successfully, then all appropriate Data Groups are opened with the correct read/write mode.

FORTRAN Interface	Subroutine OpenGroups()
C++ Prototype	void OpenGroups();

2.1.1.2 Subroutine CloseGroups

The CloseGroups subroutine is expected to be the last call in a program. This function closes all Data Groups, warning, and error files. If the program terminated normally (i.e., no errors occurred) and no warnings occurred, the warning and error files are deleted and the MMSP is called. If the program terminated normally (i.e., no errors occurred), the error file is deleted and control is returned to the MMSP. The SUI can then access the warning file (even if empty) to be displayed in the System Status Screen message box. If an error occurs, the CloseGroups subroutine is not called and the module terminates abnormally. The error file remains and control is returned to the MMSP. The SUI can then access the error and warning files to be displayed in the System Status Screen message box.

FORTRAN Interface	Subroutine CloseGroups()
C++ Prototype	void CloseGroups();

2.1.2 Input Functions and Subroutines

These subroutines provide the capability to read the SSFs, the GRFs, and the call arguments. All processors and the modules within the MMSP make use of these functions and subroutines.

2.1.2.1 Function ReadInt

This function returns the integer value for the given Data Group, variable name, and set of indices.

FORTRAN Interface	Function ReadInt[1-6](Datagroup,Variable,Units [,Index 1,...,Index 6]) Integer
C++ Prototype	int ReadInt[1-6](Datagroup,Variable,Units [,Index 1,...Index 6]);

Arguments	**FORTRAN Type**	**C++ Type**
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int

2.1.2.2 Function ReadReal

This function returns the real value for the given Data Group, variable name, and set of indices.

FORTRAN Interface	Function ReadReal[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6]) Real
C++ Prototype	float ReadReal[1-6](Datagroup,Variable,Units [, Index 1,...Index6]);

Arguments	**FORTRAN Type**	**C++ Type**
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int

2.1.2.3 Function ReadLogical

This function returns the logical value for the given Data Group, variable name, and set of indices.

FORTRAN Interface	Function ReadLog[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6]) Logical
C++ Prototype	int ReadLog[1-6](Datagroup,Variable,Units [, Index 1,...Index6]);

Arguments	FORTRAN Type	C++ Type
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int

2.1.2.4 Subroutine ReadString

This subroutine returns for the variable "String," the string value for the given Data Group, variable name, and set of indices. Due to difference between the storage of character arrays in C++ and FORTRAN this routine is not a function that returns a character array. Instead this routine expects the calling program to pass a reference to a character array to the routine. The variable "String" must be 80 characters.

FORTRAN Interface	Subroutine ReadString[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6], String)
C++ Prototype	void ReadString[1-6](Datagroup,Variable,Units [, Index 1,...Index6], String)

Arguments	FORTRAN Type	C++ Type	Intent
Datagroup	Character*20	char *	Input
Variable	Character*20	char *	Input
Units	Character*20	char *	Input
Index 1 - Index 6	Integer	int	Input
String	Character*80	char *	Output

2.1.2.5 Function NumArgs

This function returns the number of call arguments passed to a module.

FORTRAN Interface	Function NumArgs() Integer
C++ Prototype	int NumArgs()

2.1.2.6 Function GetArgInt

This function returns the integer value for the given call argument index.

FORTRAN Interface	Function GetArgInt(ArgIndex) Integer
C++ Prototype	int GetArgInt(ArgIndex);

Arguments	**FORTRAN Type**	**C++ Type**
ArgIndex	Integer	int

2.1.2.7 Subroutine GetArgString

This subroutine returns to the variable "String," the string value for the call argument index. Due to the difference between the storage of character arrays in C++ and FORTRAN this routine is not a function that returns a character array. Instead this routine expects the calling program to pass a reference to a character array to the routine. The variable "String" must be 80 characters.

FORTRAN Interface	Subroutine GetArgString(AgrIndex,String)
C++ Prototype	void GetArgString(AgrIndex,String)

Arguments	**FORTRAN Type**	**C++ Type**	**Intent**
ArgIndex	Integer	int	Input
String	Character*80	char *	Output

2.1.3 Output Subroutines

These subroutines provide the capability to write information to theGRFs. The modules within the MMSP use these subroutines.

2.1.3.1 Subroutine WriteInt

This subroutine stores the given integer value in the given Data Group, variable name, and set of indices.

FORTRAN Interface	Subroutine WriteInt[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6],Value)
C++ Prototype	void WriteInt[1-6](Datagroup,Variable,Units [, Index 1,..., Index6],Value);

Arguments	FORTRAN Type	C++ Type
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int
Value	Integer	int

2.1.3.2 Subroutine WriteReal

This subroutine stores the given real value in the given Data Group, variable name, and set of indices.

FORTRAN Interface	Subroutine WriteReal[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6],Value)
C++ Prototype	void WriteReal[1-6](Datagroup,Variable,Units [, Index 1,...Index 6],Value);

Arguments	FORTRAN Type	C++ Type
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int
Value	Real	float

2.1.3.3 Subroutine WriteLog

This subroutine stores the given logical value in the given Data Group, variable name, and set of indices.

FORTRAN Interface	Subroutine WriteLog[1-6](Datagroup,Variable,Units [, Index 1,...,Index 6],Value)
C++ Prototype	void WriteLog[1-6](Datagroup,Variable,Units [, Index 1,...Index 6],Value);

Arguments	**FORTRAN Type**	**C++ Type**
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int
Value	Logical	int

2.1.3.4 Subroutine WriteString

This subroutine stores a string of up to 80 character values in the given Data Group, variable name, and set of indices.

FORTRAN Interface	Subroutine WriteString(Datagroup,Variable,Units [, Index 1,...,Index 6],Value)
C++ Prototype	void WriteString[1-6](Datagroup,Variable,Units [, Index 1,...Index 6],Value)

Arguments	**FORTRAN Type**	**C++ Type**
Datagroup	Character*20	char *
Variable	Character*20	char *
Units	Character*20	char *
Index 1 - Index 6	Integer	int
Value	Character*80	char *

2.1.4 Error and Warning Subroutines

The error and warning files are free-format American Standard Code for Information Interchange (ASCII). All data written to these files will be echoed to the user of the FRAMES-HWIR Technology Software System through the SUI.

Errors will be written to the error file. Error files are expected to be populated only when a critical error occurs (i.e., abnormal termination of the module). Critical errors are errors that require the computation to be terminated. Examples include when a file or variable called for is not found in the location specified or the variable does not match that requested. The string description of the error that is written to this file should be detailed enough to allow the component developer to diagnose the problem but at a minimum should include component name and subroutine name, if possible.

Warnings will be written to the warning file. Warnings are descriptions of suspect calculations. The description of the warnings written to the file should be detailed enough to allow the component developer to diagnose whether there is an error.

The error and warning subroutines provide access to the error and warning files. Errors will be limited to 80 characters in FORTRAN. If a warning in a FORTRAN-programmed component must be longer than 80 characters, then multiple calls to the subroutine will be required to include the entire description.

2.1.4.1 Subroutine Error

This subroutine will add the string value passed by the calling program to the error file and then terminate the calling program.

FORTRAN Interface	Subroutine Error(Description)
C++ Prototype	void Error(Description);

Arguments	**FORTRAN Type**	**C++ Type**
Description	Character*80	char *

2.1.4.2 Subroutine Warning

This subroutine will add the string value passed by the calling program to the warning file and then return control to the calling program.

FORTRAN Declaration	Subroutine Warning(Description)
C++ Prototype	void Warning(Description);

Arguments	**FORTRAN Type**	**C++ Type**
Description	Character*80	char *

2.2 HWIR Monte Carlo Dynamic Link Library (HWIRMC.DLL)

This DLL contains the core set of Monte Carlo functions and subroutines for the FRAMES-HWIR Technology Software System. Its intended to be used to perform all sampling, from parameterized distributions, for the DSP and SDP. Typically the calling program will make use of the NumDist and NumCor subroutines to inform the HWIRMC.DLL as to the number of distributions and correlations. Then the calling program will make use of the various distribution definition functions to set the parameters for each distribution. The Cor subroutine is then used to inform the HWIRMC.DLL as to the correlations between the specified distributions. Then the sample subroutine is called to perform the actual sampling.

2.2.1 Initialization and Shutdown Subroutines

This set of routines are typically called before any sample definition subroutines or the correlation subroutines. Clear is called to return memory resources to the system (see Section 2.2.1.5).

2.2.1.1 Subroutine DebugOn

This subroutine will turn on debug messages in the HWIRMC.DLL. The default is to have messages off.

FORTRAN Declaration	Subroutine DebugOn()
C++ Prototype	void StatDebugOn();

2.2.1.2 Subroutine Seed

This subroutine sets the seed to be used by the HWIRMC.DLL. This allows for a reproducible set of samples from the given set of distributions.

FORTRAN Declaration	Subroutine Seed(newSeed)
C++ Prototype	void StatSeed(newSeed);

Arguments	**FORTRAN Type**	**C++ Type**
newSeed	Integer*4	int

2.2.1.3 Subroutine NumDist

This subroutine sets the number of distributions that will have values sampled from them.

FORTRAN Declaration	Subroutine NumDist(ND)
C++ Prototype	void StatNumDist(ND);

Arguments	**FORTRAN Type**	**C++ Type**
ND (number of distributions)	Integer*4	int

2.2.1.4 Subroutine NumCor

This subroutine sets the number of correlations between the given distributions that will have values sampled from them.

FORTRAN Declaration	Subroutine NumCor(NC)
C++ Prototype	void StatNumCor(NC);

Arguments	**FORTRAN Type**	**C++ Type**
NC (number of correlations)	Integer*4	int

2.2.1.5 Subroutine Clear

This subroutine deallocates any memory resources used by the HWIRMC.dll.

FORTRAN Declaration	Subroutine Clear()
C++ Prototype	void StatClear();

2.2.1.6 Subroutine Cor

This subroutine sets a correlation between two distributions. The subroutine NumCor is expected to be called before any call to Cor(). Index1 and Index2 need to be the indices returned by the Distribution Definition Functions.

FORTRAN Declaration	Subroutine Cor(Index1,Index2,newCor)
C++ Prototype	void StatCor(Index1,Index2,newCor);

Arguments	FORTRAN Type	C++ Type
Index1	Integer*4	int
Index2	Integer*4	int
newCor	Real*8	double

2.2.1.7 Subroutine Sample

This subroutine performs the sampling from the given set of distributios. Subroutine NumDist and NumCor are required to be called before Sample(). The Cor and Distribution Definition Functions are also expected to be called before Sample().

FORTRAN Declaration	Subroutine Sample(Num,Values)
C++ Prototype	void StatSample(Num,Values);

Arguments	FORTRAN Type	C++ Type
Num (number of distributions being sampled from)	Integer*4	int
Values	Real*8(Num)	double *

2.2.2 Distribution Definition Functions

In general the Distribution Definition Functions provide the HWIRMC.DLL with the parameters for a specific distribution type. For example, the Normal() function sets the mean, standard deviation, minimum, and maximum for a normal distribution. The number of distributions to be specified is done using the NumDist subroutine. The number of correlations between the distributions is set by the NumCor subroutine. Each of the Distribution Definition Function returns the index of the specified distribution in the internal HWIRMC.DLL distribution array.

2.2.2.1 Function Normal, LogNormal, JohnsonSB, and TrnLogNormal

These functions have four parameters that represent the mean, standard deviation, minimum, and maximum. For the logarithmic triangular distribution (TrnLogNormal), the mean and standard deviation are translated values.

FORTRAN Declaration	function Normal(mean,sd,min,max) Integer*4 function LogNormal(mean,sd,min,max) Integer*4 function JohnsonSB(mean,sd,min,max) Integer*4 function TrnLogNormal(mean,sd,min,max) Integer*4

C++ Prototype	int StatNormal(mean,sd,min,max);
	int StatLogNormal(mean,sd,min,max);
	int StatJohnsonSB(mean,sd,min,max);
	int StatTrnLogNormal(mean,sd,min,max);

Arguments	**FORTRAN Type**	**C++ Type**
mean	Real*8	double
sd (standard deviation)	Real*8	double
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double

2.2.2.2 Function Exponential and Triangular

These two functions take three parameters. For the Triangular distribution, the ct argument is the mode of the distribution, not the arithmetic mean.

FORTRAN Declaration	function Exponential(ct,min,max) Integer*4 function Triangular(ct,min,max) Integer*4

C++ Prototype	int StatExponential(ct,min,max);
	int StatTriangular(ct,min,max);

Arguments	**FORTRAN Type**	**C++ Type**
ct (central tendancy)	Real*8	double
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double

2.2.2.3 Function Uniform and IntUniform

These two functions take two parameters. The IntUniform distribution returns a whole number from min to max inclusive.

FORTRAN Declaration	function Uniform(min,max) Integer*4 function IntUniform(min,max) Integer*4

C++ Prototype	int StatUniform(min,max);
	int StatIntUniform(min,max);

Arguments	**FORTRAN Type**	**C++ Type**
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double

2.2.2.4 Function SB and SU

These functions have four parameters that represent the central tendency, square root of variance, minimum and maximum.

FORTRAN Declaration	function SB(ct,var,min,max) Integer*4 function SU(ct,var,min,max) Integer*4

C++ Prototype	int StatNormal(ct,var,min,max);
	int StatLogNormal(ct,var,min,max);

Arguments	**FORTRAN Type**	**C++ Type**
ct	Real*8	double
var (value similar to standard deviation)	Real*8	double
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double

2.2.2.5 Function Gamma and Weibull

These functions have four parameters that represent the shape, scale, minimum, and maximum.

FORTRAN Declaration	function Gamma(shape,scale,min,max) Integer*4 function Weibull(shape,scale,min,max) Integer*4

C++ Prototype	int StatGamma(shape,scale,min,max);
	int StatWeibull(shape,scale,min,max);

Arguments	**FORTRAN Type**	**C++ Type**
shape	Real*8	double
scale	Real*8	double
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double

2.2.2.6 Function Empirical(min,max,Num,Values,CumProb)

This function has five parameters that represent the minimum, maximum, Number of value probability pairs (Num), Values, and the associated Cumulative Probabilities (CumProb)**.**

FORTRAN Declaration	function Empirical(min,max,Num,Values,CumProb) Integer*4
C++ Prototype	int StatEmpirical(min,max,Num,Values,CumProb);

Arguments	**FORTRAN Type**	**C++ Type**
min (minimum value)	Real*8	double
max (maximum value)	Real*8	double
Num	Integer*4	int
Values	Real*8 Values(Num)	double *
CumProb	Real*8 CumProb(Num)	double *

3.0 Specifications for the Distribution Statistics Processor

The DSP randomly samples from statistical distributions representing measurement and sampling error related to the statistics of parameters required by the modules in the MMSP (e.g., distribution, mean, standard deviation, and range). The DSP receives input from the Regional and national environmental setting distribution statistics databases. Output from the DSP populates the statistical portion of the regional statistics and national statistics databases.

3.1 Read and Write Requirements of the Distribution Statistics Processor

The DSP is expected to read the header SSF to know which site and realization is currently being executed. Like all processors and modules, any errors that occur in the DSP are expected to be written to the GRF directory. The variable definitions for the header SSF can be found in Appendix A, Table A.1.1. The DSP is required to read and write in the following manner:

Call Arguments	**Description**
SSF_Dir	Call for Site Simlation Files Directory
GRF_Dir	Call for Global Results Files Directory
Header File Name	Header file contains all information needed to run entire simulation

Read expectations	National and Regional Distribution Statistics Databases (format discussed in Section 3.2)
Write expectations	National and Regional Statistics Databases (format discussed in Section 3.2

## 3.2	National and Regional Environmental Setting Distribution Statistics Database Formats

This section describes the file formats for the different databases used as input to the DSP and files created by the DSP. The national environmental setting distribution statistics database will be named NDistStat.mdb. The regional version of that same file will be RDistStat.mdb. The NDistStat.mdb and RDistStat.mdb file format will be a Microsoft® Access database with the four tables listed below:

1)	Variable distribution data (see Table 3.1) and example of variable distribution data (see Table 3.2)
2)	Cross-Correlation Data (see Table 3.3) and example of cross-correlation data (see Table 3.4)

3) User-defined distribution data (see Table 3.5) and example of User-Defined Distribution Data (see Table 3.6)
4) Reference data (see Table 3.7) and example of reference data (see Table 3.8).

Table 3.1 Variable Distribution Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Description	Type	Number of Characters
Setting_ID	text string that is a unique identifier for a variable	Text	20
Data_Group_Name	text string that defines which data group this variable is in	Text	20
Variable_Name	text string that defines this variable's name	Text	20
Units	units of variable	Text	20
Index_1	define the indices for this variable	Number (Integer)	
Index_2	define the indices for this variable	Number (Integer)	
Index_3	define the indices for this variable	Number (Integer)	
Index_4	define the indices for this variable	Number (Integer)	
Index_5	define the indices for this variable	Number (Integer)	
Index_6	define the indices for this variable	Number (Integer)	
Data_Type	type of information this variable stores. String, Logical, Real, and Integer are the possibilities	Text	10
Reference_Index	defines the reference number for this variable	Text	20
Minimum	minimum value for this variable	Number (Double)	
Maximum	maximum value for this variable	Number (Double)	
Distribution_Type	distribution type for this variable. Constant, Uniform, Normal, Log Uniform, Log Normal, User Defined, etc. are the possibilities	Text	12
CTD_Distribution_Typ e	Distribution type of the Central Tendency Distribution (CTD); Constant, Uniform, Normal, Log Uniform, Log Normal, User Defined, etc. are the possibilities	Text	10
CTD_Central_Tenden cy	CTD's Central Tendency value. Mean for normal distribution for example. It is also the value that will be used in a constant distribution.	Number (Double)	
CTD_Variance	CTD's Variance parameter. Standard Deviation for normal distribution for example.	Number (Double)	
CTD_Minimum	CTD's minimum value	Number (Double)	

Field Name	Description	Type	Number of Characters
CTD_Maximum	CTD's maximum value	Number (Double)	
CTD_User_Def_Index	If the distribution type is user defined this value defines the index of the user defined distribution values in the User Defined Distribution Data table.	Text	20
VD_Distribution_Type	Variance Distribution; Constant, Uniform, Normal, Log Uniform, Log Normal, User Defined, etc. are the possibilities	Text	12
VD_Central_Tendency	VD's Central Tendency value. Mean for normal distribution for example. It is also the value that will be used in a constant distribution.	Number (Double)	
VD_Variance	VD's Variance parameter. Standard Deviation for normal distribution for example.	Number (Double)	
VD_Minimum	VD minimum value	Number (Double)	
VD_Maximum	VD maximum value	Number (Double)	
VD_User_Def_Index	If the distribution type is user defined this value defines the index of the user defined distribution values in the User Defined Distribution Data table.	Text	20
Cross_Correlation	Cross correlation between the CTD and VD	Number (Single)	

Table 3.2 Example of Variable Distribution Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Record 1	Record 2	Record 3	Record 4	Record 5	Record 6
Setting_ ID	EPA_1	EPA_1	EPA_1	EPA_1	EPA_1	EPA_1
Data_Group_Name	SRC_1	SRC_1	SRC_1	SRC_1	SRC_1	SRC_1
Variable_Name	sw_var46	sw_var46	sw_var46	sw_var46	ATArea	veg
Units	cm/yr	cm/yr	cm/yr	cm/yr	cm2	fraction
Index_1	1	1	1	1	0	0
Index_2	1	2	3	4	0	0
Index_3	0	0	0	0	0	0
Index_4	0	0	0	0	0	0
Index_5	0	0	0	0	0	0
Index_6	0	0	0	0	0	0
Data_Type	float	float	float	float	float	float
Reference_Index	Example1	Example1	Example1	Example1	Example1	Example1
Minimum	20	20	40	40	30	10
Maximum	80	60	60	80	40	80
Distribution_Type	Normal	Normal	Normal	Normal	User Defined	Normal
CTD_Distribution_Type	Normal	Normal	Normal	Normal	Normal	Normal
CTD Central_Tendency	15	15	15	15	0	0.5

Field Name	Record 1	Record 2	Record 3	Record 4	Record 5	Record 6
CTD_Variance	5	5	5	5	0	0.15
CTD_Minimum	0	0	0	0	25	0
CTD_Maximum	30	30	30	30	45	1
CTD_User_Def_Index					AT Area	
VD_Distribution_Type	Normal	Normal	Normal	Normal	User Defined	Normal
VD_Central_Tendency	15	10	0	20	10	0.5
VD_Variance	5	5	5	10	0	0.5
VD_Minimum	0	0	0	25	10	5
VD_Maximum	30	30	20	40	50	20
VD_User_Def_Index					AT Area	
Cross_Correlation	1	-1	1	0	1	1

Table 3.3 Cross-Correlation Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Type	Number of Characters
V1_Setting_ID	Text	20
V1_Data_Group_Name	Text	20
V1_Variable_Name	Text	20
Units	Text	20
V1_Index_1	Number (Integer)	
V1_Index_2	Number (Integer)	
V1_Index_3	Number (Integer)	
V1_Index_4	Number (Integer)	
V1_Index_5	Number (Integer)	
V1_Index_6	Number (Integer)	
V2_Setting_ID	Text	20
V2_Data_Group_Name	Text	20
V2_Variable_Name	Text	20
V2_Index_1	Number (Integer)	
V2_Index_2	Number (Integer)	
V2_Index_3	Number (Integer)	
V2_Index_4	Number (Integer)	
V2_Index_5	Number (Integer)	

Field Name	Type	Number of Characters
V2_Index_6	Number (Integer)	
Cross_Correlation	Number (Single)	

Table 3.4. Example of Cross-Correlation Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Record 1	Record 2	Record 3	Record 4
V1 Setting ID	EPA_1	EPA_1	EPA_1	EPA_1
V1 Data Group Name	SRC_1	SRC_1	SRC_1	SRC_1
V1 Variable Name	sw_var46	sw_var46	sw_var46	sw_var46
Units	cm/yr	cm/yr	cm/yr	cm/yr
V1_Index_1	1	1	1	1
V1_Index_2	1	2	3	4
V1_Index_3	0	0	0	0
V1_Index_4	0	0	0	0
V1_Index_5	0	0	0	0
V1_Index_6	0	0	0	0
V2_Setting_ID	EPA_1	EPA_1	EPA_1	EPA_1
V2_Data_Group_Name	SRC_1	SRC_1	SRC_1	SRC_1
V2_Variable_Name	veg	veg	veg	veg
V2_Index_1	0	0	0	0
V2_Index_2	0	0	0	0
V2_Index_3	0	0	0	0
V2_Index_4	0	0	0	0
V2_Index_5	0	0	0	0
V2_Index_6	0	0	0	0
Cross_Correlation	0.85	0.85	0.85	0.85

Table 3.5. User-Defined Distribution Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Type	Number of Characters
User Defined Dist. Index	Text	20
Value	Number (Double)	
Cumulative Probability	Number (Double)	

Table 3.6 Example of User-Defined Distribution Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Record 1	Record 2	Record 3	Record 4
User Defined Dist. Index	ATArea	ATArea	ATArea	ATArea
Value	25	35	40	45
CumulativeProbability	10	60	90	100

Table 3.7 Reference Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Type	Number of Characters
Reference Index	Text	20
Reference Description	Memo	Unlimited

Table 3.8 Example Reference Data for the Regional and National Environmental Setting Statistics Databases

Field Name	Record 1
Reference Index	Example1
Reference Description	This is a reference for example purposes only!

4.0 Specifications for the Site Layout Processor

The SLP calculates fractions that relate one component of site layout information to another. The SLP receives input from the site delineation database. Output from the SLP populates a portion of the site-based database.

4.1 Read and Write Requirements of the Site Layout Processor

The SLP is required to read and write in the following manner:

Call Arguments	**Description**
Spatial Delineation Database	Fully qualified path to the Spatial Delineation Database
Site-Based Database	Fully qualified path to the Site-Based Database
Percent of coverage	The percent used to determine the number of points to represent a given area

Read expectations	Spatial Delineation Database (format described in Section 4.2) Site-Based Database (format described in Section 5.2)
Write expectations	Fractional site layout information for the Site-Based Database (format described in Section 5.2). Also the air layout for each site in the Site-Based Database.

4.2 Spatial Delineation Database Format

The database format for the spatial delineation database will contain five tables. The five tables all have the same format. While the Universal Trans-Mercator (UTM) coordinates do not define a measurement system for elevations in Table 4.1 and 4.2, the column name UTM_Z is used to identify elevations that are associated with the UTM_X and UTM_Y columns. The five tables are as follows:

1) Farm
2) Human
3) Ecological
4) Lakes
5) Watershed.

Table 4.1 shows the format for these data tables, and Table 4.2 provides an example table. In the example table assume that there are three farms at a site. Table 4.2 shows that cells (345.45,1233.0) and (345.45,1234.0) are associated with the first farm. Cell (346.45,1233.0) is associated with the second farm. A separate table is used for each of the following areas: farm, human receptor, ecological, lakes, and watersheds. If Table 4.2 was a watershed table, then it would state that cells (345.45,1233.0) and (345.45,1234.0) are associated with the first watershed.

Table 4.1 Format for Data Table in the Spatial Delineation Database

Field Name	Type	Number of Characters
SETING_ID	Text	20
UTM_X	Double	
UTM_Y	Double	
UTM_Z	Double	
AREA_ID	Integer	

Table 4.2 Example Table for the Spatial Delineation Database

Setting _Id	UTM_X	UTM_Y	UTM_Z	Area_ID
EPA1	345.45	1233.0	0	1
EPA1	345.45	1234.0	0	1
EPA1	346.45	1233.0	0	2
EPA1	347.45	1233.0	0	3

5.0 Specifications for the Site Definition Processor

The SDP organizes all data for input to the SSFs by accessing databases containing all the necessary information and executing a hierarchical protocol to read these databases and extract the required data. It also provides the preliminary simulation plan and control information that could be used by the MMSP. The SDP receives input from a series of databases. Output from the SDP populates the Site Definition Files.

5.1 Read and Write Requirements of the Site Definition Processor

Like all processors and modules, any errors that occur in the SDP are expected to be written to the GRF directory. The variable definitions for the header SSF can be found in Appendix A, Table A.1.1. The SDP is required to read and write in the following manner:

Call Arguments	**Description**
SSF_Dir	Site Simulation Files Directory (which is also the location of the Site Definition Files)
GRF_Dir	Global Results Files Directory
Header File Name	Contains all information needed to run entire simulation

Read expectations	Site-Based, Modified and Static Regional and National Databases (format discussed in Section 3.2), Chemical Properties Database (format discussed in Section 6.2)
Write expectations	Site Simulation Files (format discussed in Section 7.2)

5.2 Model Error Statistics, Site-Based, Static National, Static Regional, National Statistics, Regional Statisics, and Site Survey Database Formats

The SDP uses a standardized Microsoft® Access database to define inputs for the seven databases listed below. File names are included in parentheses.

1) Model error statistics (ModErrStat.mdb)
2) Site based (SiteData.mdb)
3) Site survey (SitSurv.mdb)
4) Static regional (StaticRegData.mdb)
5) Regional statistics (RegData.mdb)
6) Static national (StaticNatData.mdb)
7) National statistics (NatData.mdb)

All databases, except the regional statistics and national statistics, are expected to be populated before execution of the SUI. The regional statistics and national statistics databases will be populated by the Distribution Statistics Processor (see Section 3.0).

The database format for the model error statistics, site based, site survey, static national, and static regional databases will contain four tables. The four tables are similar to the four in the regional and national environmental setting statistics databases. The four tables are as follows:

1) Variable distribution data (see Table 5.1) and example of variable distribution data (see Table 5.2)
2) Cross-correlation data (see Table 5.3) and example of cross-correlation data (see Table 5.4)
3) User-defined distribution data (see Table 5.5) and example of user-defined distribution data (see Table 5.6)
4) Reference data (see Table 5.7) and example of reference data (see Table 5.8).

Table 5.1 Variable Distribution Data for Site Definition Processor Input Databases

Field Name	Type	Number of Characters
Row_ID	Number (Integer)	
Setting_ID	Text	20
Data_Group_Name	Text	20
Variable_Name	Text	20
Units	Text	20
Index_1	Number (Integer)	
Index_2	Number (Integer)	
Index_3	Number (Integer)	
Index_4	Number (Integer)	
Index_5	Number (Integer)	
Index_6	Number (Integer)	
Data_Type	Text	10
Reference_Index	Text	20
Distribution_Type	Text	12
Central_Tendency	Number (Double)	
Variance	Number (Double)	
Minimum	Number (Double)	
Maximum	Number (Double)	

Field Name	Type	Number of Characters
User_Defined_Dist._Index	Text	20
String_Value	Text	80
Logical_Value	Logical	
Weight	Number (double)	
Auto_Filled	Logical	

Table 5.2 Example of Variable Distribution Data for Site Definition Processor Input Databases

Field Name	Record 1	Record 2	Record 3	Record 4	Record 5	Record 6
Row_ ID	1	2	3	4	5	6
Setting_ID	EPA_1	EPA_1	EPA_1	EPA_1	EPA_1	EPA_1
Data_Group_Name	SRC_1	SRC_1	SRC_1	SRC_1	SRC_1	SRC_1
Variable_Name	sw_var46	sw_var46	sw_var46	sw_var46	ATArea	veg
Units	cm/yr	cm/yr	cm/yr	cm/yr	cm2	fraction
Index_1	1	1	1	1	0	0
Index_2	1	2	3	4	0	0
Index_3	0	0	0	0	0	0
Index_4	0	0	0	0	0	0
Index_5	0	0	0	0	0	0
Index_6	0	0	0	0	0	0
Data_Type	float	float	float	float	float	float
Reference_Index	Example1	Example1	Example1	Example1	Example1	Example1
Distribution_Type	Normal	Normal	Normal	Normal	User Defined	Normal
Central_Tendency	15	15	15	15	0	0.5
Variance	5	5	5	5	0	0.15
Minimum	0	0	0	0	25	0
Maximum	30	30	30	30	45	1
User_Defined_Dist._Index					ATArea	
String_Value						
Logical_Value	F	F	F	F	F	F
Weight	1.0	1.0	1.0	1.0	1.0	1.0
Auto_Filled	F	F	F	F	F	F

Table 5.3 Cross-Correlation Data for Site Definition Processor Input Databases

Field Name	Type	Number of Characters
V_ Setting_ID	Text	20
V1_Data_Group_Name	Text	20
V1_Variable_Name	Text	20
V_Units	Text	20
V1_Index_1	Number (Integer)	
V1_Index_2	Number (Integer)	
V1_Index_3	Number (Integer)	
V1_Index_4	Number (Integer)	
V1_Index_5	Number (Integer)	
V1_Index_6	Number (Integer)	
V2_Data_Group_Name	Text	20
V2_Variable_Name	Text	20
V2_Index_1	Number (Integer)	
V2_Index_2	Number (Integer)	
V2_Index_3	Number (Integer)	
V2_Index_4	Number (Integer)	
V2_Index_5	Number (Integer)	
V2_Index_6	Number (Integer)	
Cross_Correlation	Number (Single)	

Table 5.4 Example of Cross-Correlation Data for Site Definition Processor Input Databases

Field Name	Record 1	Record 2	Record 3	Record 4
V1_Setting_ID	EPA_1	EPA_1	EPA_1	EPA_1
V1_Data_Group_Name	SRC_1	SRC_1	SRC_1	SRC_1
V1_Variable_Name	sw_var46	sw_var46	sw_var46	sw_var46
V_Units	cm/yr	cm/yr	cm/yr	cm/yr
V1_Index_1	1	1	1	1
V1_Index_2	1	2	3	4
V1_Index_3	0	0	0	0
V1_Index_4	0	0	0	0
V1_Index_5	0	0	0	0
V1_Index_6	0	0	0	0
V2_Setting_ID	EPA_1	EPA_1	EPA_1	EPA_1
V2_Data_Group_Name	SRC_1	SRC_1	SRC_1	SRC_1
V2_Variable_Name	veg	veg	veg	veg
V2_Index_1	0	0	0	0
V2_Index_2	0	0	0	0
V2_Index_3	0	0	0	0
V2_Index_4	0	0	0	0
V2_Index_5	0	0	0	0
V2_Index_6	0	0	0	0
Cross_Correlation	0.85	0.85	0.85	0.85

Table 5.5 User-Defined Distribution Data for Site Definition Processor Input Databases

Field Name	Type	Number of Characters
User Defined Dist. Index	Text	20
Value	Number (Double)	
Cumulative Probability	Number (Double)	

Table 5.6 Example of User-Defined Distribution Data for Site Definition Processor Input Databases. The number of user defined distribution entries is implied by the number of records in the database that match the User_Defined_Dist._Index (for example ATArea).

Field Name	Record 1	Record 2	Record 3	Record 4
User Defined Dist. Index	ATArea	ATArea	ATArea	ATArea
Value	25	35	40	45
Cumulative Probability	10	60	90	100

Table 5.7 Reference Data for Site Definition Processor Input Databases

Field Name	Type	Number of Characters
Reference Index	Text	20
Reference Description	Memo	Unlimited

Table 5.8 Example of Reference Data for Site Definition Processor Input Databases

Field Name	Record 1
Reference Index	Example1
Reference Description	This is a reference for example purposes only!