AD-A248 374

E200949

2

PL-TR-91-2298(III)

INTELLIGENT EVENT IDENTIFICATION SYSTEM. VOLUME III: SOFTWARE MAINTENANCE MANUAL

Sam Carter
Michael Maxson
Jeanne Carney
Kathleen Ziegler
Douglas Baumgardt

ENSCO, Inc.
5400 Port Royal Road
Springfield, VA 22151-2301

DTIC
ELECTE
MAR 10 1992
S B D

November 1991

Final Report (Volume III)
1 March 1990-30 September 1991

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

92-06110

PHILLIPS LABORATORY
AIR FORCE SYSTEMS COMMAND
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731-5000

92 3 [illegible] 089

SPONSORED BY
Defense Advanced Research Projects Agency
Nuclear Monitoring Research Office
ARPA ORDER NO. 5307

MONITORED BY
Phillips Laboratory
Contract F19628-90-C-0049

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

This technical report has been reviewed and is approved for publication.

[illegible]
JAMES F. LEWKOWICZ
Contract Manager
Solid Earth Geophysics Branch
Earth Sciences Division

[illegible]
JAMES F. LEWKOWICZ
Branch Chief
Solid Earth Geophysics Branch
Earth Sciences Division

[illegible]
DONALD H. ECKHARDT, Director
Earth Sciences Division

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify PL/IMA, Hanscom AFB, MA 01731-5000. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE November 1991	3. REPORT TYPE AND DATES COVERED Final Report (1 Mar 1990-30 Sep 1991)

4. TITLE AND SUBTITLE
Intelligent Event Identification System.
Volume III: Software Maintenance Manual

5. FUNDING NUMBERS
PE 62714E
PR 9A10 TA DA WU AB
Contract F19628-90-C-0049

6. AUTHOR(S)
Sam Carter
Kathleen Ziegler
Michael Maxson
Douglas Baumgardt
Jeanne Carney

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ENSCO, Inc
5400 Port Royal Road
Springfield, VA 22151-2301

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Phillips Laboratory
Hanscom AFB, MA 01731-5000

Contract Manager: James Lewkowicz/GPEH

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
PL-TR-91-2298(III)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The objective of this project is to design and develop an Intelligent Event Identification System, or ISEIS, which will be a prototype for routine event identification of small explosions and earthquakes and to serve as a tool for discrimination research. The first part of this report gives an overview of the system design and the results of a preliminary evaluation of the system on events in Scandinavia and the Soviet Union. The system was designed to be highly modular to allow the easy incorporation of new discriminants and/or discrimination processes. Because the main objective of the system is the identification of small events, most of the initial ISEIS prototype discriminants utilize regional seismic data recorded by the regional arrays, NORESS and ARCESS. However, ISEIS can easily process other regional array data (e.g., from GERESS and FINESA), as well as data from three-component single stations, as more of this data becomes available. The second part of this report is entitled *Intelligent Event Identification System: User's Manual,* and gives a detailed description of all the processing interfaces of ISEIS. The third part of this report is entitled *Intelligent Event Identification System: Software Maintenance Manual*, which describes the ISEIS software from the programmer's perspective and provides information for maintenance and modification of the software modules in the system.

14. SUBJECT TERMS

	NORESS	High-frequency
Discriminiation	ARCESS	Waveform matching
Regional propagation	Coda	Dynamic time warp
Expert system		Ripple-fire

15. NUMBER OF PAGES
164

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

TABLE OF CONTENTS

☑
☐
☐

DTIC COPY INSPECTED 6

By
Distribution/
Availability Codes

Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

INTELLIGENT EVENT IDENTIFICATION SYSTEM
Maintenance Manual

1.0 OVERVIEW AND DESIGN PHILOSOPHY

This document describes the Intelligent Seismic Event Identification System from the point of view useful to software maintainers. This document assumes that the personnel maintaining this system are familiar with the Sun UNIX-operating system principles, the X window system, the NMRD software development guidelines, and the ORACLE database design as used in NMRD. This document concentrates on the ISEIS software and database usage.

1.1 FUNCTIONAL HIERARCHY

The ISEIS software is composed of four significant functional areas. These are the top level discrimination summary, the interactive map, the ISEIS signal processing functions, and the ISEIS discriminants. The top level discriminant summary presents the operator with the first view of the ISEIS processing results. This information is presented in the form of a set of "buttons" which display the results in the form of lighted buttons. This function controls the majority of the processing within the ISEIS system. The only exception is the interactive map which operates in parallel with the top level summary to provide the operator with additional information concerning the location of the events, and geological and demographic information which may be associated with the event.

In addition to the four primary elements of the ISEIS software, an automation driver which combines the signal processing and discriminant processing together is designed to automatically classify events as they are placed into the NMRD database. This automatic processing provides initial classifications which may be reviewed by the operator using the top level summary. The automatic processing is an epoch driven element which maintains a processing epoch time in the NMRD database. This epoch time is used to identify new events for processing. As events are processed and classified, the epoch time is advanced to reflect the latest event.

ISEIS is modularly designed to permit easy integration of additional discriminants when they become available.

1.2 OPERATIONAL HIERARCHY

Operationally, the ISEIS system consists of the automatic processing, which is then evaluated by the ISEIS operator using the top-level spreadsheet process and the interactive map. Supporting this analysis are the interactive display functions and the signal processing functions of ISEIS.

The philosophy of the interactive processing is a tiered approach to data analysis. The operator may evaluate the results of the discrimination processing using the top level spreadsheet process. This process shows the results of the discrimination processing as a series of lights which use colors to indicate the processing results. In general, a green light indicates results which require no operator intervention, i.e., earthquakes, complete data, similar case-based results; red lights indicate conditions which the operator should investigate further, i.e., possible nuclear explosions, missing data, or dissimilar events. Yellow or orange lights indicate results which may require operator intervention, but which are not critical. This includes incomplete data and economic explosions.

The operator can see the results for all selected events at a glance by examining these lights. Should further evaluation be desired, the operator can select one of the lights for an event and get a textual description of the processing involved in the evaluation. Also, the operator can call up the interactive signal processing associated with each discriminant and examine or modify the parameters used in the processing.

1.3 OPERATOR INTERFACE CONCEPTS

The operator interface used in the ISEIS system is modeled on the pulldown menu interface which is widely used in the computer industry. These menus include the standard **FILE, EDIT**, and **VIEW** menus, as well as specific menus which are appropriate to the currently active function. These menus are activated by selecting the menu button with the left button and then dragging the cursor down to the desired option and releasing the button. This action selects the marked menu option and activates the associated processing.

In addition to the pulldown menus, pop-up dialog boxes have been implemented to query the operator for additional information when necessary. These dialog boxes use a combination of selection buttons, lists, and command buttons to specify the options necessary for the processing requested.

1.4 DISCRIMINANT FUNCTIONAL OVERVIEW

The discrimination processing is the mainstay of the ISEIS system. This software has been developed with the intent that scientists and researchers can easily add new discriminants to the system as they become available. The discrimination processing software consists of five separate tasks along with possible data-analysis tasks which are used to generate parameters for the discrimination tasks. Three of the tasks perform the actual discrimination processing of data assessment, case-based discrimination and model-based discrimination. The remaining two tasks are interface drivers which allow the discriminant to communicate with the top level spreadsheet task. These two tasks accept requests from the operator via the spreadsheet and perform the appropriate processing. One task performs both data analysis and rule-based processing, whereas the other task simply performs the rule-based processing in order to update the identification after manual data processing by the operator.

1.5 SIGNAL PROCESSING FUNCTIONAL OVERVIEW

The ISEIS system performs a great deal of signal processing data analysis above and beyond that performed by the IMS system. This signal processing is required to provide the necessary parametric information required by the ISEIS discriminants. The signal processing tasks predominantly used by the ISEIS system include incoherent beamforming, single and continuous spectral generation, cepstral evaluation, dynamic signal warping, amplitude and spectral ratio computation, sonogram generation, and continuous F-k processing.

1.6 AUTOMATIC PROCESSING FUNCTIONAL OVERVIEW

In order to provide the operator with timely results and eliminate the need for tedious and time-consuming data processing, the automatic discrimination process performs the bulk of the discrimination on each new IMS event prior to operator intervention. This process executes a predefined set of well established discriminants and the associated signal processing on each event and generates a preliminary identification whenever possible. The results of the processing are stored for later operator review and modification when necessary. If the operator modifies any of the parametric data, the discrimination processing can be reapplied to update the identification interactively.

1.7 DATABASE DESIGN AND INTERFACE OVERVIEW

The ISEIS database is modeled on the Center for Seismic Studies (CSS) 3.0 database and every attempt has been made to make the databases compatible. The standard 3.0 relations have been used whenever possible along with a set of additional relations to contain processing results unique to the ISEIS system. These additional relations have been developed to be fully compatible with the existing 3.0 relations with minimal duplication of data. All database relations are accessed via the SQL/ORACLE interface using the existing access routines whenever possible.

All inter-process communications in ISEIS use the Dispatcher of the Intelligent Array System (IAS) for internal communications, and the Intelligent Monitoring System (IMS) ISIS process for external communication with the rest of the NMRD system. Future plans call for converting the internal communications to use the ISIS process as well.

1.8 EXTERNAL AND INTER-PROCESS INTERFACE OVERVIEW

In general, the ISEIS system operates independently, relying on the IMS system to provide event hypocenter parameters, called "origins." This processing is controlled by a "TIMESTAMP" relation in the IMS database. Essentially, processing is controlled by identifying new IMS events based on their origin time as compared to the latest "timestamp" for ISEIS processing. Any event with a time later than the last timestamp is considered a "new event." New events are processed by the ISEIS system and the results are stored in the IMS database. The ISEIS system also updates the origin relation with the new event identification.

Future plans call for providing interaction between the IMS ARS process and the ISEIS interactive processing. In essence, the operator can select events using the ARS process and send these events to the ISEIS top level spreadsheet process for examination. Theoretically, the results for these events should have been generated by the automatic ISEIS processing and should be readily available. The operator should also have the ability to send events from the ISEIS interactive process to the ARS for waveform processing and phase selection. These results should then be available to the ISEIS system for re-evaluation.

2.0 ISEIS OPERATOR INTERFACE

This section describes the ISEIS operator interface. The operator interacts with the ISEIS functions through two main interfaces, the spreadsheet and the interactive map. This section provides the internals of these interfaces.

2.1 TOP LEVEL SPREADSHEET PROCESS INTERNALS

The Top Level Spreadsheet is one of the key components of the ISEIS system. It coordinates discriminant processing and enables viewing of processing results in a hierarchical fashion. In general, the design makes extensive use of UNIX environment variables and configuration files to facilitate easy configuration.

2.1.1 Initialization and Display Creation

The ISEIS Top Level Spreadsheet consists of one to 40 rows of events and one to 20 columns of discriminants and a special Combined Identification discriminant. The Spreadsheet serves as a means for executing discriminant processes and for examining results of previously executed discriminants. The maximum numbers of events and discriminants can be modified by changing the MAXEVT and MAXTECH define statements in the 'dsmain.h' include file. These numbers are set to these values to insure maximum performance and flexibility. The system administrator may want to rebuild the Spreadsheet with a larger MAXTECH if significant discriminant 'cloning' is going to be attempted.

The Spreadsheet first attempts to connect to the Dispatcher message routing process. If the Dispatcher is not running, then the Spreadsheet and most of the other processes will not run. Once the Spreadsheet has gained access to the Dispatcher, it installs a Dispatcher message handling routine such that incoming Dispatcher messages are seen as X window events. The Spreadsheet has to run in a completely asynchronous mode since operator actions have to be dealt with in a timely fashion. Once communications have been established, the display is created.

The Spreadsheet obtains its current list of origins or events from a scratch file called "eventFile.d." This file is created and modified whenever an "OPEN DB" function is selected from either the Map process or the Spreadsheet. The Spreadsheet uses the eventFile.d file, which was created last before the process begins. If the Map obtained a new set of events after "OPEN DB" or "OPEN DB" was selected from the Spreadsheet FILE menu, then the Spreadsheet reads the file and fills in the event data structures. Each event is represented by:

```
typedef struct _event
{
   int OriginId;
   int EventType;
   int epoch;
   Boolean IsSelected;
   Boolean IsActive;
   char EventLabel[80];
   int COMBidx;
   int NCOMBtechs;
   int COMBtechs[MAXTECHS];
} EVENT, *EVENTP;
```

The OriginId is a unique value taken from the ORIGIN relation in the database. Even types are also obtained from the ORIGIN relation, but they can also be updated by "ORID UPDATE" selections from the EXECUTE menu. Epoch is the time that the event occurred. It is used for date sorting when the operator selects "VIEW by DATE" from the spreadsheet VIEW menu. Whenever the operator highlights an event box by clicking there, the IsSelected flag is set to TRUE to indicate the state. The IsActive flag is used to indicate if information pertaining to the event should be displayed on the Spreadsheet. This field is used when DELETE and UNDO are selected from the EDIT menu. There are two sets of EVENT structure arrays. The 'InitialEvents' array is filled out first and then copied to 'Events.' If an operator deletes some events, the IsActive fields in the highlighted events are set to FALSE in 'InitialEvents,' all active InitialEvents are copied to Events and the Spreadsheet is redrawn. If an UNDO is selected, all InitialEvent IsActive flags are set to TRUE, InitialEvents are copied to Events, and the Spreadsheet is redrawn, thus restoring the original display. The EventLabel contains the string which is displayed on the Event buttons. It is created using parameters in the eventFile.d file. COMBidx, NCOMBtechs and COMBtechs[] are run-time variables which pertain to Combined Identification discriminant execution and they will be discussed in Discriminant Processing (Section 2.1.3).

The Spreadsheet obtains its current list of discriminant processes or techniques from the MachAlloc file, which is discussed in detail in Appendix B. The Spreadsheet scans the non-commented entries in this file looking for non '*' entries in the COLUMN_NAME field. For each column name, all '_' characters are replaced with ' ' and all '\' characters are replaced with carriage returns and the resultant name is used as a technique entry in the Spreadsheet. Thus, adding more techniques to the Spreadsheet display is as simple as adding another row of information into the MachAlloc with the COLUMN_NAME field set as a non '*.'

The Spreadsheet arranges the techniques so that the top-most entry in the MachAlloc is left most in the display. Conversely, the bottom most entry will be displayed to the far right in the display. The technique information is placed into two arrays of technique structures represented by:

```
typedef struct _descrim
{
  int TechId;
  Boolean IsSelected;
  Boolean IsActive;
  char Column_name[40];
} DESCRIM, *DESCRIMP;
```

The TechId field is loaded from the TECHID column of the MachAlloc file. The IsSelected field is set to TRUE when the operator highlights a column of techniques by clicking on a technique name. The IsActive flag is used to indicate if information pertaining to the technique should be displayed on the Spreadsheet. This field is used when DELETE and UNDO are selected from the EDIT menu. There are two sets of DESCRIM structure arrays. The 'InitialTechTypes' array is filled out first and then copied to 'TechTypes.' If an operator deletes some techniques, then the IsActive fields in the highlighted techniques are set to FALSE in 'InitialTechTypes.' Then all active 'InitialTechTypes' are copied to 'TechTypes' and the Spreadsheet is redrawn. If an UNDO is selected, then all InitialTechType IsActive flags are set to TRUE, InitialTechTypes are *copied to* TechTypes and the Spreadsheet is redrawn, thus restoring the original display. An UNDO will restore both Events and Techtypes to original settings.

Once the set of events and techniques is known, the individual cell objects are created. Data structures for cells are dynamically allocated, based upon the number of events and techniques. Currently, no more than 40 events and 20 (MAXTECHS) techniques are allowed. Each cell consists of a hierarchy of Athena toolkit X window Form, Command and Label widgets. Each Cell is indexed by an OriginId and a TechId and its parameters are represented by:

```
typedef struct _dsbuttonp
{
  Boolean IsSelected;
  Widget bform;
  Widget lwov_vote;
  Widget cwbuttons[NBUTTONS];
  Widget lwvalues[NBUTTONS];
  int button_status[NBUTTONS];
  float button_values[NBUTTONS];
  Boolean BHasProb[NBUTTONS];
  Boolean PendingExecution;
  Boolean Executing;
  Boolean OHasProb;
  float fov_vote_value;
} DSBUTTON, *DSBUTTONP;
```

The IsSelected flag is set to TRUE when the operator clicks on the top most descriptive text widget in the cell. The bform background is set to the highlight color when clicked. lwov_vote is the widget. The three colored button widgets below the text are stored in cwbuttons[NBUTTONS]. In this case, NBUTTONS is equal to three. lwvalues represents the label widgets that lie below the colored 'Status,' 'Model' and 'Case' command button widgets. The colors and values of cwbuttons and lwvalues are set by accessing the DISCTECH relation in the database for the appropriate parameters. The DISCTECH contains information as to whether a particular discriminant has enough data to run, whether it was an Earthquake, Explosion, Mine

Blast, Marine blast or undecided, and whether it was Similar, Dissimilar or Atypical for reference events in the region, etc. The Spreadsheet converts these codes to colors by checking what colors are assigned to each of these codes.

There are entries in '.Xdefaults-com' for colors and they are:

```
dsmain.NoDataStatusColor: Red
dsmain.SufficientDataStatusColor: LimeGreen
dsmain.IncompleteDataStatusColor: Yellow
dsmain.EarthquakeEventColor: LimeGreen
dsmain.NuclearEventColor: Red
dsmain.EconomicEventColor: Coral
dsmain.UnidentifiedEventColor: Yellow
dsmain.NoDataEventColor: Gray40
dsmain.MatchRefWellColor: LimeGreen
dsmain.MatchRefIntermediateColor: Yellow
dsmain.MatchRefPoorlyColor: Red
dsmain.ItemSelectColor: Red
dsmain.ItemDeSelectColor: Gray40
dsmain.NotAvailable: White
dsmain.Pending: Blue
```

These can be set to other colors if desired.

Once all data is collected and all widgets have been created, the Spreadsheet is realized and the display becomes visible. The display is implemented as an Athena Toolkit Pop-up shell class. Thus, when the Spreadsheet configuration changes (i.e., DELETE, UNDO VIEW, ACTIVATE, OPEN DB) the new spreadsheet is popped-up over the old one and the old one is destroyed. This reduces trauma because the operator doesn't see the spreadsheet suddenly vanish while it is being re-configured.

2.1.2 Run State Processing and Monitor Interface

Once the Spreadsheet is up and running, it is totally event-driven. Possible events include incoming messages from the Monitor or other processes which were originally started by the Spreadsheet. The Spreadsheet also receives regular alarm signals which it uses to check its process queue. At every alarm, it first checks to see if it is already running the maximum number of processes as specified in the environment. If it is not, it sends GET_SYSTEM_LOAD messages to all system Monitors to check on system loading. When all Monitors have replied, with GET_SYSTEM_LOAD messages, the Spreadsheet runs the process by sending a PROC_START_REQ to the desired system monitor if the reported system loads were low enough. Otherwise, the Spreadsheet waits until the next alarm signal and it repeats the same cycle again. Lastly, the Spreadsheet responds to user mouse events and menu selections.

2.1.3 Discriminant Processing

After an operator has done an OPEN DB function or the Map process has sent a NEW_ARGS message to the Spreadsheet telling it to display new events, he may find that some cells have no processing results *(all three buttons are white if we use the above color resources)*. If this is the case, then the operator first needs to determine if there is adequate data to perform the processing and then must execute the discriminant. Any references to colors below pertain to the current resources defined above in Section 2.1.1.

2.1.3.1 Assessing Data Status for Discriminants

If the STATUS button in a cell is white, then it is not possible to view results summaries and the discriminant cannot be executed. Once the user has selected the cell(s) desired for data assessment, he should select ASSESS/SELECT or ASSESS/ALL from the EXECUTE menu. The spreadsheet first queues all OriginId/TechId cells which were not previously executing and then it queries all system Monitors for system loads as described in 2.1.2. The Spreadsheet continues to query the Monitors, receive replies, and execute until the number of active processes is equal to that given by the UNIX environment variable ISEIS_MAX_PROCESSES. At this point, the Spreadsheet resumes Run State Processing mode (2.1.2). The global variable 'Wid_parms.NActiveProcs' is incremented every time a process is executed and decremented every time one finishes. 'Wid_parms.NQueuedProcs' is incremented every time a process is queued and decremented every time one is executed. The PendingExecution flag in the DSBUTTON structure is also set to TRUE. When the discriminant begins executing, it sends a DSM_EXECUTING or a TRM_EXECUTING message to the Spreadsheet. The message is ignored for discriminant processes, but the spreadsheet displays a status message when other types of processes started by the Spreadsheet begin executing. After the discriminant process has completed any required numerical processing and has completed its rule processing, it returns a DSM_FINISHED or TRM_FINISHED message to the Spreadsheet. This message protocol is the same for all types of discriminant processes, whether they are assessing data, executing or updating. At this time, the Spreadsheet accesses the DISCTECH relation in the database to obtain the new data status and the data status button color is set accordingly. If the data status is insufficient (Red), then the status summary should be viewed to find out why an action should be taken to provide the appropriate data. The cell must then be assessed again.

2.1.3.2 Executing Discriminants

Once successful data status has been achieved for cell(s), they can be executed by first selecting the desired cell(s) and selecting EXECUTE/SELECT or EXECUTE/ALL from the EXECUTE menu. As explained before, the message protocol is the same as above. However, the execution processing ultimately determines model-based and/or case-based classification for the given event and discriminant. Execution-type discriminants may also perform numerical computation. The DTW discriminant is one example of a process which must perform numerical computation prior to rule-based evaluation. Discriminant processing differs from signal processing in that discriminant processing is used to directly derive a statistic which will later be used in the rule-based evaluation. Discriminant processes always perform CLIPS rule processing and database updates.

2.1.3.3 Updating Discriminants

Sometimes, the user only wants to redo the rules and not the numerical processing for discriminants. The UPDATE/SELECT or UPDATE/ALL options from the EXECUTE menu do this.

2.1.3.4 <u>The Combined Identification Discriminant</u>

The Combined Identification discriminant is a special type of technique which derives a final model-based and case-based result for a given event based on all of the other discriminant model- and case-based results. The EVENT data structure is repeated below for clarity:

```
typedef struct _event
{
   int OriginId;
   int EventType;
   int epoch;
   Boolean IsSelected;
   Boolean IsActive;
   char EventLabel[80];
   int COMBidx;
   int NCOMBtechs;
   int COMBtechs[MAXTECHS];
} EVENT, *EVENTP;
```

The Combined discriminant is assessed, executed and updated just like all other discriminants, except for cases where the Combined discriminant is selected with other discriminants for the same event. In this case, 'COMBidx' is set to either STATUS_BX, EXECUTE_BX, or UPDATE_BX, depending on whether the group is being assessed, executed or updated. Then, the 'COMBtechs' array is loaded with the TechIds associated with each non-combine discriminant in the row. 'NCOMBtechs' is set to indicate the number of non-combine discriminants which must finish and be accounted for before the Combined ID discriminant is run. In group selects, the Combined ID discriminant is always run last. The COMBidx, NCOMBtechs and COMtechs fields are not used when a Combined ID discriminant is run individually.

Whenever a discriminant finishes with a TRM_FINISHED or DSM_FINISHED message, the Spreadsheet checks to see if there are any COMBtechs registered for the event. If there are, the Spreadsheet removes the techid associated with the discriminant that just finished from the

COMBtechs list and NCOMBtechs is decremented. When all registered discriminants have finished, the Combined ID processing begins.

If the group is being assessed, then no Combined ID processes are started since it is meaningless to assess Combined ID data status for techniques which have not been executed yet. The data requirements for Combined ID consist of the individual technique results which can only be derived by execute or update processing.

However, group execute processing is done in a different way. When the last discriminant in the COMBtechs list has finished, a Combined ID data assess process is started (.DTA) since Combined ID data assessment is now valid. Once the assessment process has finished, then the Combined ID execute process is started. This is a rule-based process just like any other discriminant and when it finishes, the Spreadsheet accesses the DISCTECH relation and updates Combined cell button colors. It also updates the relative weights of all of the other discriminant cells in the row.

The Combined rule update processing functions just like the execute processing, except that a Combined ID update process (.UPD) is run after all of the other techniques in the row have been updated.

2.1.3.5 Discriminant Processing Internals

The root name for the discriminant executable is given in the EXECUTABLE column in the MachAlloc file. Once the Spreadsheet determines where to run the discriminant process, it determines the X window display name by checking the X_DISPLAY column in the MachAlloc. It then appends a '.DTA' onto the root name for data assess operations, an '.EXE' for execute operations or an '.UPD' for update operations. The following message is then sent to the Monitor:

```
START_PROC executable[.DTA||.EXE||.UPD] TechId OriginId DataBase_Userid dsmain.Any -d Xdisplay >/dev/null
```

where TechId and OriginId are the technique and event ids, respectively. The parameter dsmain.Any is the Dispatcher address of the Spreadsheet. This is how all of the processes started by the Spreadsheet know where to send their executing and finished messages. In fact, all processes started by the Spreadsheet use the same first four arguments, namely:

TechId	Technique Identification;
OriginId	Event Identifier;
DataBase_Userid	Login Id for database; and
Parent	Dispatcher destination @ to send execute & finished msgs.

2.1.4 Other EXECUTE Menu Options

The KILL/SELECT option is used to abort discriminants and to initialize the Spreadsheet execution state so that discriminants can be run again. KILL/SELECT is necessary under these possible scenarios:

1. Spreadsheet solicits loads from Monitors and one or more fail to reply.
2. Spreadsheet starts discriminants and they terminate abnormally before TRM_FINISHED, TRM_ABORTED, DSM_FINISHED, DSM_ABORTED msgs are sent.
3. Dispatcher timeouts result in above messages being dropped.

Any of the above scenarios could prevent the Spreadsheet from running additional discriminants. KILL/SELECT flushes all queued processes and sets the 'PendingExecution' and 'Executing' flags to FALSE.

The ORID UPDATE/SELECT and ORID UPDATE/ALL options are used to update the ORIGIN relation with new event types, which were determined by the special Combined Id discriminant. In addition, the new event type is sent to the map process so that event icon colors can be updated there as well.

The operator selects events that he wants to update and then he selects ORID UPDATE/SELECT. The Spreadsheet first updates the ORIGIN relation via a library routine and this routine sends a DSM_CHANGE_EVENT_ID to the Map process.

2.1.5 Results Summaries

2.1.5.1 Standard Discriminant Summary Displays

Results Summaries are displayed when the user mouses on the Status, model and case colored buttons in each of the Spreadsheet cells. The discriminant processes produce *.DTA, *.STD, *.MDL, *.MBD, *.CSE, and *.CBD explanation and dribble files with the '*' set to the event id. Every discriminant has its own subdirectory where all explanation and dribble files for various events are stored. A dribble file is of the form *.??D and it represents a trace history of the CLIPS rule processing. It is rather meaningless to those not familiar with the CLIPS inference engine. It is usually viewed by a trained CLIPS programmer to verify the legitimacy of rule assertions and firings. A set of CLIPS rules files of the form *.STR, *.MBR, *.CBR exist for each discriminant in a separate directory.

When the user attempts to view a summary, the Spreadsheet first checks the button status to see if it is possible to view the summary. If the user mouses on a non-assessed status button (White), or he mouses on a non-executed model or case button (White), the Spreadsheet will display error dialogs since there will not be any explanation or dribble files to display in the summary displays. If a summary is viewable, the Spreadsheet creates three Athena toolkit transientShell pop-up widgets which contain text for an explanation, a dribble and rules. The dribble and rules pop-ups contain QUIT and SAVE buttons on the bottom of the scrollable text region. SAVE allows user edits to be saved and QUIT removes the display. The SAVE option provides an easy interface for a CLIPS programmer who wishes to change CLIPS rules within the ISEIS environment. Initiall,, the Spreadsheet displays a summary for either a data assessment, a model-based result or a case-based result, depending upon whether the user moused on a status, model or case button. Every explanation summary has at least three option buttons on the bottom of the display called RETURN, DRIBBLE and RULES. A mouse on the RETURN button causes the Spreadsheet to remove the explanation summary, as well as any dribble or rules displays. DRIBBLE and RULES enable the dribble and rules displays, respectively. Every discriminant has an associated *.P optional summary procedures file, which contains entries specifying optional processes which can be started from the explanation display. The entries consist of button names followed by the name of the executable process, which is to be started when the button is moused upon. The Spreadsheet uses this file to lay out additional summary option buttons to the right of the standard three buttons. This allows the user to start up processes from the summary displays. The user usually does this when he needs to create some data needed for discriminant execution. All summary processes can also be started from the Spreadsheet PROCESS menu, but this provides a more direct interface.

The Spreadsheet uses either the ALIAS_NAME field or the EXECUTABLE field in the associated entry for the discriminant in the MachAlloc file to determine the name of the explanation file subdirectory and the name of the optional summary processes file (*.P). The ALIAS_NAME field is always used unless it is a '*.' In this case, the EXECUTABLE field is used.

2.1.5.2 Combined ID Summary Displays

The Combined ID summaries are implemented differently than those of individual discriminants. A user mouse on a Combined ID status button enables a status summary just as described in Section 2.1.5.1. However, the Spreadsheet displays special color bar summaries for model and case mouse picks. Each completed discriminant for an event is represented by a shaded colored bar. If it is a model-based summary, the color is the same as the model button on the discriminant cell. Likewise, a case summary has colors that are the same as the case-based results buttons on the discriminant cells. The discriminant weight is used to determine how large the bar is and in what direction it extends. A special proprietary X toolkit widget is used to create color bars for each completed model or case-based discriminant. The discriminant names displayed to the left of each color bar are obtained from the RULEDISC relation and user mouse-clicks in these buttons result in the display of standard summaries. The Spreadsheet allocates X window color cells in the default system color map so it may not be possible to display another process that allocates color cells like the Map on the same display.

2.1.6 Process Selection and Execution

The Spreadsheet starts non-discriminant processes differently than discriminant processes. A non-discriminant process is any process that doesn't end in DTA, EXE, UPD, mbid or cbid. The Spreadsheet starts a non-discriminant process by first randomly assigning a system for execution. It then sends a PROC_START_REQ to the selected system Monitor to start the process. Non-discriminant processes are always started since they are usually interactive processes and response time is tantamount. Every non-discriminant process should send a TRM_EXECUTING message to the Spreadsheet so that the Spreadsheet can display a processes executing status message. These messages reassure the operator that the process did, in fact, start and that it is not necessary to start it again.

2.1.7 Make Path Function

The Spreadsheet displays a form which allows the user to make entries into the PATH relation when MAKE PATH is moused on from the FILE menu.

2.1.8 Automatic Discrimination

Automatic discrimination is used by the casual user who wishes to have ISEIS automatically assess and execute discriminant processes to arrive at individual and combined results. During the course of automatic discrimination, the ISEIS autoDS process computes Incoherent Beams, phase picks and other necessary signal processing data files before invoking the discriminant processes. These are the same assessment (*.DTA) and execution (*.EXE) processes that are invoked by the Spreadsheet during interactive operation. When all discrimination processes have been completed, autoDS sends a FINISHED message back to the spreadsheet to signify completion.

Automatic processing is invoked by selecting the RUN AUTO/SELECT or RUN AUTO/ALL options from the Spreadsheet EXECUTE menu. An event is processed in automatic mode if at least one discrimination cell is selected in the event row and there are no automatic or interactive assessments or executions in progress for the given event. The Spreadsheet displays a warning dialog if there were any active discriminants in a selected event row. The Spreadsheet first queues the autoDS process and executes it like any other discriminant process. This includes the system load query protocol (see Monitor section). If system loads are low enough and there are not too many processes already running, then the Spreadsheet starts the autoDS process. When the Spreadsheet runs a discriminant, it compares 'Wid_parms.NActiveProcs' with ISEIS_MAX_PROCESS to determine if it is allowed to run. However, in the automatic case, the Spreadsheet verifies that there are N active process slots available where N is equal to the number of discriminant processes currently configured into the Spreadsheet. The

'Wid_parms.NActiveProcs' counter is then incremented by this amount to indicate that the automatic process could simultaneously run this many processes. The number of processes must be controlled because it is very easy to exceed the maximum amount of database users. The Spreadsheet then marks the entire row of discriminant cells for the event to indicate that processing is underway.

2.1.9 Discriminant Cloning

When the user selects COPY from the EDIT menu, discriminant cloning functionality is enabled. A clone is a discriminant that is identical to the discriminant it is derived from, except that a different set of CLIPS rules is used for result classification. A library routine in libDscU is called in response to the user selection. The library routine prompts the user for the desired discriminant to clone and then it prompts him for its name and description. A happy face dialog appears when the database (RULEDISC), MachAlloc file, rule files and optional summary processes files have been prepared. The library routine copies the rules from the parent discriminant to the cloned discriminant. It also copies the $ISEIS_CONFIG_DIR/sumprocs/*.P file using the clone name. Furthermore, a unique discriminant techid is selected by determining the largest techid in the current MachAlloc file and incrementing it. Then a new MachAlloc entry is created. Thus, when the Spreadsheet is started again, it will contain the new cloned discriminant. The ORACLE RULEDISC relation is also augmented.

2.1.10 Rule Editing Functions

CLIPS rules may either be modified via the results summary interface, or they may be modified from the menubar interface. The menubar interface is usually used after a discriminant has been cloned (see Section 2.1.9). It would not be possible at this time to edit the rules for the cloned discriminant, since nothing has been assessed or executed. Status, Model-based and Case-based CLIPS rules are edited by first selecting a discriminant cell in a column that corresponds to the desired discriminant and selecting either STATUS RULES, MODEL RULES or CASE RULES from the EDIT menu in the Spreadsheet. These functions enable editing windows similar to the text editing windows seen in summary displays. The user can then modify the CLIPS rules and save changes with the SAVE button.

2.1.11 Spreadsheet Help Dialogs

The Spreadsheet process provides help dialogs similar to UNIX man pages. The user obtains help by mousing with the MIDDLE mouse button on a Spreadsheet command button. The help dialogs for the Status, Model, Case and Cell Title buttons provide general information on the use and function of the button. The text for the help dialog windows is obtained from $ISEIS_CONFIG_DIR/help/dsmain/FILENAME where FILENAME is obtained from an X window translations resource entry in the .Xdefaults-com X window resource file. Help file names for discriminants are of the form TechsXX.h where XX is equal to the technique ID for the discriminant. These technique IDs are obtained from the TECHID field in the MachAlloc. Textual information for the Event dialogs is created from the $W/eventFile.d file and it is merged with the general Events.h file in $ISEIS_CONFIG_DIR/help/dsmain. Help for Spreadsheet menu items is obtained by first selecting MENU ITEM HELP from the Spreadsheet VIEW menu and then selecting the desired menu item from the menubar in the help dialog. Menu help files have the form M.XX.YY.h where XX is the menubar item index and YY is the menu item index. For example, if the user selected menu help and then he selected the second menu item under the first menu bar, then the menu help file M.00.01.h would be displayed in the resultant menu help dialog. The help dialog will also display the menubar item name and the menu item name as part of the help dialog title.

2.2 ISEIS STARTUP INTERNALS

A UNIX csh script called '.StartIseis' along with the MachAlloc file control the start up of ISEIS and the initial process configuration. The .StartIseis script first performs 'ps' commands on the AESIR_HOST system (see Section 8.0) to see if the Dispatcher IPC process is running (see Section 7.0). .StartIseis invokes an 'rsh' for it on AESIR_HOST if it is not. Once the Dispatcher is running, .StartIseis performs similar status checks on each of the ISEIS member systems specified in ISEIS_HOSTNAMES to insure that the system Monitors are running. Monitors are started on all member systems without active Monitors. Again, status checks for X window servers are performed on all systems which are part of ISEIS_HOSTNAMES AND which have at least one entry for the system display in the MachAlloc file. For example, if 'seismic' is a system in ISEIS_HOSTNAMES, but there is no 'seismic:0' display listed in the X_DISPLAY column in the MachAlloc file, then X windows is not started on 'seismic.' Conversely, if there is at least one 'seismic:0' entry in X_DISPLAY, then X windows is started on 'seismic.' For each host name in ISEIS_HOSTNAMES, there should be a .XI[hostname] and a .xi[hostname] file. These scripts are responsible for starting X windows on the member systems. The .XI* files usually perform a 'xinit' specifying the .xi* counterpart as the file to execute from 'xinit.' Now that the ISEIS run-time environment is established, all processes listed in PROC_LIST are started with the arguments specified in PROC_ARGS. .StartIseis retrieves the host system name and X window display name from the MachAlloc and invokes the .StartProc script for each process. The .StartProc UNIX script is used for starting processes for which there can only one be run-time copy. ISEIS is not designed to run multiple Spreadsheets, Maps or RGB color editors. The .StartProc script insures that the given process is not executing already. If the process is not executing, then .StartProc sends a startup message to the desired system Monitor using the 'send_args' utility. If the process is running, then .StartProc merely sends a 'NEW_ARGS' message to the indicated process to load a new set of events. The Map and Spreadsheet reconfigure themselves in response to NEW_ARGS messages. The RGB color editor simply ignores a NEW_ARGS message. The .StartIseis script is automatically run whenever a user logs on to ISEIS and answers 'Y' to the "Start ISEIS (Y/N)" prompt.

2.3 ISEIS INTERACTIVE MAP PROCESS

2.3.1 Map Initialization Procedures

The Map performs many of the same initialization functions as the Spreadsheet process. It obtains its initial events via the $ISEIS_WORKSPACE_DIR/eventFile.d event file and it establishes Dispatcher connections. The Map must also operate in an event-driven mode, so it installs an X window event handler for Dispatcher messages using XtAddInput(). Whenever the Map loads new events, it attempts to load the smallest map that will contain the current list of events. This insures maximum magnification for event locations. The Map 'FindMap()' procedure compares the event locations with the latitude/longitude bounds for each map listed in the $ISEIS_CONFIG_DIR/MAPS/MapInfo file. Each entry in this file contains this information, as well as the projection used. If a small map that encloses all of the events cannot be found, the default world map is used. Once a suitable map is found, it is loaded and displayed. The Map then creates shell widgets for each event and displays them on the map using XtPopup(). In addition, Map also creates a bulletin entry for each event. Each bulletin entry displays textual parameters for each event, and the bulletin list can be scrolled when there are more than five events. Clicking on bulletin entries results in highlighting of the bulletin entry and its corresponding event symbol. An inverse video highlighting scheme is used to indicate which event is selected in the bulletin.

2.3.2 Loading New Events

The Map loads new events whenever the OPEN DB function is selected from the FILE menu and DONE is selected and it also loads new events when a NEW_ARGS message is received from the Spreadsheet process via the send_args utility. When new events are to be loaded, the Map first removes the current events by destroying the event symbols and bulletin entries. Events are destroyed by invoking the Athena toolkit functions XtPopdown() and XtDestroyWidget(). Once the events are destroyed, the Map creates new event widgets by creating new event symbols and bulletin entries as described in Section 2.3.1. A suitable map is also selected and loaded as described in 2.3.1.

2.3.3 Overlays

The Map can display overlays which may be one or more of city, geo-political boundaries, station symbols and latitude/longitude lines. Overlay data files exist in the $ISEIS_CONFIG_DIR/overlays directory and the file names are derived from data in the MapInfo file. When overlays are selected by selecting OVERLAYS from the VIEW menu, the desired files are loaded and displayed. Overlay selections disappear when maps are changed.

2.3.4 Message Interaction with Top Level Spreadsheet Display

The Map receives NEW_ARGS messages from the Spreadsheet as described in Section 2.3.2, and it also sends this message to the Spreadsheet. NEW_ARGS messages are sent to the Map from the Spreadsheet in response to a SHOW MAP selection from the Spreadsheet PROCESS menu and NEW_ARGS messages are sent to the Spreadsheet from the Map in response to a DISCRIMINATE selection from the Map EXECUTE menu. When the Spreadsheet sends this message to the Map, it loads events and re-configures the display as described in Sections 2.3.1 and 2.3.2. When the map sends events to the Spreadsheet, it displays a pop-up dialog box, which allows the user to send all events to the Spreadsheet, to send just the selected ones or to cancel the operation. If all events are sent, then a NEW_ARGS message is sent. However, if the 'selected events only' option was selected, the Map creates a sublist file in $ISEIS_WORKSPACE_DIR/sublist which contains a list of the selected events. When the Spreadsheet receives this message, it reads the events from the $ISEIS_CONFIG_DIR/eventFile.d file just as usual, but it only loads events which are listed in the sublist file. It is important to note that these messages are not directly sent. Instead, they are indirectly sent via a special 'send_args' message transmission utility. This intermediate step is done because the Map and Spreadsheet must first be started before they can receive messages. Any NEW_ARGS messages are sent from the .StartProc UNIX script via send_args. The script first determines if the desired process is already running. If it is, only a NEW_ARGS message is sent via send_args. However, if it is not, the script starts the process using arguments listed in the environmental variable PROC_ARGS.

2.3.5 The Cross-Section Display

The ISEIS Map has a cross-section display capability which allows users to view elevation and MOHO data for given slices on the Map. The Map obtains the topographic data from the $ISEIS_TOPO_FILE/etopo5.d file. The NMRD LYNESS calcomp emulation widgets are used for this display.

2.3.6 Region Operations

The Map provides an interface which allows users to create, modify, and delete ISEIS regions. A region is a database entity which groups similar events together and associates them with a geographical area. When users invoke the region functions, the REFREGION and REFEVENT database relations may be modified. The Map provides create, edit and view region functions.

The CREATE REGION function is selected from the FILE menu and in response, the Map displays a legend creation form on the right side of the screen. The user must have previously selected one or more events by either clicking on bulletin items or event symbols. Either action should result in the highlighting of both a bulletin item and an event symbol. The user then enters the name of the region and selects DONE or CANCEL. At this time, the Map determines if the region already exists by searching for the name in REFREGION relations in the database. Assuming that the region does not exist, the Map enters the region into the REFREGION table in the database and sets the region latitude/longitude bounds to the minimum dimension that will enclose the selected events. It then enters all of the selected events into the REFEVENT relation in the database.

The EDIT REGION option under the EDIT menu allows the user to add events to a region, delete events from a region and to delete regions. If the user desires to add events to a region, he selects the desired region by clicking on the desired region in the region editing display. When a region is selected, the bounding box for the region is displayed. At this time, the user can perform any editing function. If the user wants to add events to the region, he selects the events that he wishes to add and selects the 'ADD EVENTS TO REGION' option. The Map will not add duplicate entries into a REFREGION relation. The Map adds events to REFREGION only when the selected events are not already present. Thus, subsequent selections of 'ADD EVENTS TO REGION' with the same set of selected events results in no change to the database. Similarly, the Map only removes events from the REFEVENT relation if the events are present there. The Map uses the set of currently selected events to determine which events to remove from an active region. The 'DELETE REGION' option causes the REFREGION relation and all associated REFEVENT relations for the selected region to be deleted from the database. When a region is deleted, the Map destroys the region-editing display and re-creates it to reflect the modified database. The Map always highlights the events which are in a particular region whenever a new region is selected. Thus, the user always knows which events are in a given region.

The VIEW REGION option is selected by selecting REGIONS under the VIEW menu. This is a view-only option which permits no modification of the database relations. The view display is similar to the edit display, but there are some differences. The Map allows more than one region to be displayed at a time, whereas the edit function only allows one to be displayed at a time. When multiple regions are displayed, the Map highlights the set of events which are contained in the set of all currently displayed region bounding boxes.

2.3.7 <u>Other Functions</u>

The NEW MAP function under the FILE menu allows the user to display other maps. When the user selects another map, the Map hides the event symbols via XtPopdown() and loads the new map image. The Map then determines which events lie within the new map and it only displays the visible event symbols. Any corresponding non-visible bulletin items are dim-lighted to indicate that these events loaded, but that they are not visible on the current map.

The GREAT CIRCLE PATH function simply plots a great circle path between two user selected points. The STATION GC PATH function simply plots great circle paths from each event to each of its associated stations. Other functions are explained in the user's guide and the implementation simply involves the use of well-established algorithms.

3.0 DISCRIMINANT PROCESS DETAILS

The primary components of the ISEIS software are the discriminant processes. These processes have been designed to provide modularity, reusability, and functionality. The discrimination processes are a combination of database routines written in C and data analysis routines developed in the CLIPS expert system language.

Each discrimination process is divided into three distinct functions, data assessment, model-based discrimination and case-based discrimination. The model-based and case-based discrimination may also include some additional data processing prior to application of the rules to determine the event identification. Rules can be applied separately if the data processing has been performed previously. This functionality consists of five separate processes: one for data assessment; one for model-based reasoning; one for case-based reasoning; one to perform data preprocessing and then initiate the model and case-based processes; and one to simply execute the model and case-based processes without performing the data preprocessing. These programs have specific extensions to identify functions of each one. The extensions and their functions are:

• .DTA	Data Status evaluation;
• .mbid	Model Based Discrimination;
• .cbid	Case Based Discrimination;
• .EXE	Complete Discrimination Processing; and
• .UPD	Discrimination Update.

The intent of the dicriminant design is to provide both a model and a procedure to facilitate the addition of new discriminants as they are developed. To support this design, a library of routines have been developed to support the common functions of storing and retrieving the discriminant data and results. Also, a set of routines have been developed to serve as models for accessing data from the database and passing this information to the CLIPS rules.

3.1 DISCRIMINANT PROCESS HIERARCHY

The discriminant processes are found in the **CLIPS** subdirectory of the **discrim** directory. Each process has its own subdirectory which is further divided into a library directory and five executable directories, one for each process. Figure 1 shows a sample directory hierarchy for a typical discrimination process.

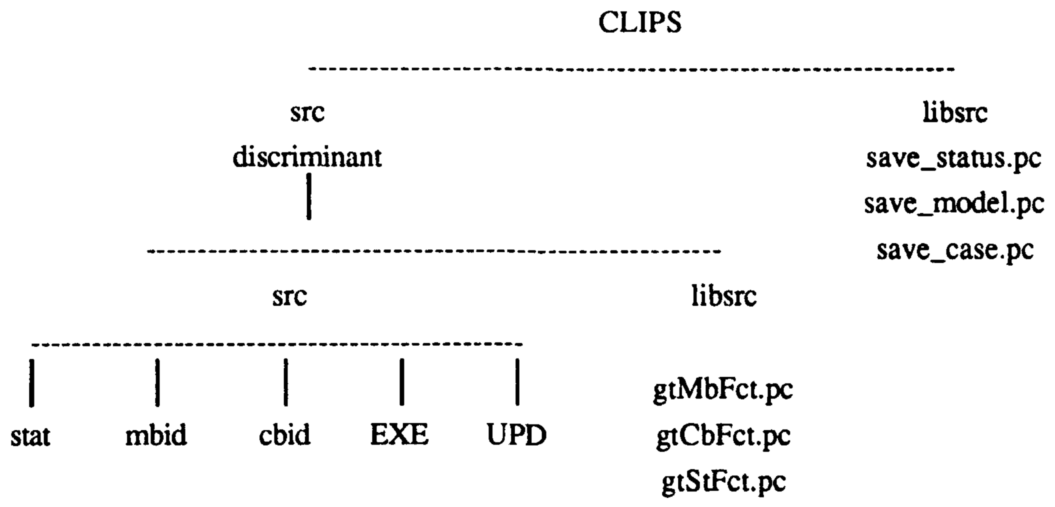

FIGURE 1: TYPICAL DISCRIMINANT DIRECTORY HIERARCHY

3.2 DISCRIMINANT DATABASE INTERFACE

The database interface for the discriminant is contained in two separate directories. Routines to fetch and retrieve information from the discrimination relations are contained in a **libsrc** subdirectory at the **CLIPS** level. The routines to fetch databases for each discriminant are contained in a **libsrc** directory under each discriminant. The next section describes the general discriminant interface routines. Section 3.2.2 describes the general nature of the specific discriminant database interface routines. The discriminant database relations are described in Section 3.2.3.

3.2.1 General Interface Modules

Three routines have been developed to interface with the discrimination relations. These routines save the results of the status, model-based and case-based processing, and are used by all of the discrimination processing and are invoked by the CLIPS rules when a result has been obtained. Generally, these routines can be used by all discriminants without modification.

3.2.2 Specific Interface Modules

Each discriminant uses a set of three database interface modules to acquire information for the specific discrimination processing. These routines collect the specific information required by the status, model-based, or case-based processing and assert the information in the form of facts for the rule-based processing. The general format of the routines is the same for each discriminant, but the nature and format of the information which is asserted are not, as facts vary with the needs of the specific discrimination process. Developers of new discriminants should copy a sample of these routines and tailor the database access sections to meet the needs of the new discriminant.

3.2.3 Discriminant Database Relations

The ISEIS database maintains two relations to retain information necessary for the discrimination processing. These are the **ruledisc** relation and the **disctech** relation. The **ruledisc** relation contains the path names and filenames for each discriminant. These names include the path name and filename for the rules for each discriminant, as well as the path name and filename for the result files which are produced during the processing.

The **disctech** relation contains the results of the discrimination processing for each discriminant. This relation also contains the relative and absolute weights used to compute the combined confidence for the Composite discriminant. This data is used to generate top level ISEIS display.

3.3 DISCRIMINATION PROCESSING

The discrimination processing begins with the data status process. This process uses a set of rules to evaluate the condition of the data associated with an event and a particular discriminant. The process determines whether sufficient data exists to generate a valid result. The database status process evaluates the data for both the case-based and the model-based processes. The data status process saves a status of *no-data, incomplete,* or *complete* for the discriminant. This status is then used to determine whether further processing is possible. Discriminants which have incomplete status may be executed, but the confidence of the results will be reduced due to missing data.

Once the data status process has been performed, the model-based and case-based processes can be performed for any discriminant which has an incomplete or complete status. Some discriminants, such as depth or magnitude, require only the application of the specific discrimination rules to generate the discrimination result. Other discriminants, such as Ripple-fire and DTW, require preliminary signal processing to generate intermediate results which are then used in the discrimination processing. In order to facilitate the differences in these different requirements, the specific case-based or model-based processing have been encapsulated in two drivers. The first driver performs any preliminary signal processing before the rules are executed, while the second process simply executes the rules assuming that the preliminary processing has already been performed. This dichotomy of processes was developed to allow the user to reapply the discrimination rules without re-executing the signal processing after it had been performed once. This prevents the user from having to wait for lengthy signal processing functions which have already been performed and which provide no additional information. In some instances, these two processes are identical, but the distinction is maintained for consistency among the discriminants. These two processes are contained in the EXE and UPD directories respectively.

The actual rule-based discrimination is performed by the **.cbid** and **.mbid** processes. These are contained in the **cbid** and **mbid** directoiies, respectively. These processes are executed by the **.EXE** and **.UPD** processes. The **cbid** process performs the case-based discrimination processing and the **mbid** process performs the model-based processing.

4.0 DISCRIMINANT SIGNAL PROCESSING DETAILS

The backbone of the ISEIS processing is comprised of the discriminant signal processing functions. These functions compute the parametric information which is used by the discrimination functions to classify events in the ISEIS system. These functions currently consist of:

Incoherent Beam Processing,
Single Fourier Spectra Processing,
Continuous Fourier Spectra Processing,
Dynamic Time Warp Processing,
Continuous F-k Processing,
Amplitude Ratio Processing,
Spectral Ratio Processing, and
Ripple-fire Detection Processing.

This section presents a description of each of these processes, how they are invoked, the nature of their processing, and the results of the processing.

Each of these functions may be invoked either by the automatic processing or from the interactive processing functions. If invoked by the interactive processes, the user is given an option to view the results of the processing. When invoked from the automatic processing, the viewing option is not available directly but may be accessed via the top level spreadsheet VIEW menu options.

4.1 DISCRIMINANT SIGNAL PROCESSING INITIATION

In the interactive mode, the signal processing for the discrimination processing is initiated via menu selections from the EXECUTE menu of the ISEIS top level spreadsheet or the interactive map. These functions are invoked by first selecting an origin of interest and then invoking the function from the EXECUTE menu. The user has the ability to interact with the selected function to choose various processing parameters such as stations, channels, filters, and viewing options. The results of the processing are stored in the ISEIS database for use by other discrimination functions. In the interactive mode, the origins available for processing are limited to those which are currently displayed in the top level spreadsheet and the map.

The automatic processing driver invokes these same processes on every new origin using a default set of parameters which are designed to supply the minimum set of parametric information necessary for initial classification of events by the automatic processing. When invoked by the automatic process, the viewing options are disabled. The results of the processing are still stored in the database. The automatic processing may also be invoked from the interactive processes, i.e., the spreadsheet and the map, on a set of selected origins which may or may not have already been processed. This mode is available to allow the user to easily repeat the automatic processing for origins which have acquired new data.

4.2 COMMON DISCRIMINANT SIGNAL PROCESSING ISSUES

Each discriminant requires feature information for input into the CLIPS rule-based inference engine. Some features are already present in the database whereas others have to be computed by signal processing support processes. Processes described in the following sections function as display and feature generation processes. Before any features can be determined, certain rudimentary data must be generated. There are several basic steps that must be performed before discrimination features may be determined:

1. Compute short-term averages (incoherent beams) from raw waveform data for a given event.
2. Define seismic phases for all incoherent beams for the given event. A phase is simply a time window on the beam. Each seismic phase occurs at a unique time.
3. Compute Fourier spectra for each phase from waveform data for the given event. Locations and durations of spectral phases for spectral computation may differ from those defined for incoherent beam computation.
4. Compute discrimination features from the above data. Each discriminant signal processing support process derives different types of statistics.

The above steps are done in roughly the same order regardless of whether interactive or automatic discrimination is invoked.

Time-consuming processing is usually done in the background and when it completes, a display process may be started as an icon to signify completion. A user is then able to mouse upon the icon to view the data that was computed.

In general, most signal processing display processes query the database to determine the set of stations where the given event arrived. Once the set of stations is known, all channels and waveform IDs for each channel are retrieved. Station/channel information is necessary for most signal processing display processes and there are boilerplate library routines to do it.

Signal processing display processes appear as a window with a menubar. The user may view existing data by selecting options from the VIEW menu. If he is not satisfied with the existing data, he may select different phases and/or processing parameters by choosing options under the SELECT menu. He may compute new data at any time by choosing options under the EXECUTE menu. In some applications, he may quit the display process after computation has begun. An icon will appear later indicating that the computation has completed as described earlier in this section.

4.3 SPECIFIC PROCESS DETAILS

This section contains descriptions of the signal processing functions which compute and extract the discrimination parameters for use in the discriminants. The descriptions include a discussion of the initialization procedures, the user options, and the processing details.

4.3.1 Incoherent Beam Processing

Incoherent beam processing can be done in either automatic or interactive modes. The automatic mode runs the beams for an origin ID for a default set of filters that will be described later. The interactive mode allows the user to choose the filter bands of interest. It also provides a means of viewing the output results.

4.3.1.1 Interactive Mode

The display process IBeam is the driver for the interactive mode. It has Selection, Execution, and View options. IBeam uses the Menu Widget to get to the various options. A list and description of the global variables used by IBeam is contained in the include file: *IBeamg.h* and the related files in *include/Xdsp*.

DATA INITIALIZATION

Data initialization is performed partially at startup, and partially as needed. At startup, or when a new seismic network is selected, the routine *doinit* is called. This routine finds the available networks and the channel names and associated waveform IDs for the current origin (routine *gwfidstas - libdbio*). The sample rate and channels for the current default station, given by the global variable "ipbsta," are then obtained. A filter band list is filled with either the default filter bands (*defilts - libdbio)* for the IBeam process or with the filter bands which have been processed by the view results display, IBeamVws. A global variable "iRflag" indicates whether IBeam or IBeamVws is calling the initialization routine. The routine also sets up the default values for features to be displayed, signal and noise windows, filters to be displayed, number of beam and time series points, and channel marking arrays. The actual beam and time-series data is read in by the plotting routines as it is needed.

SELECTIONS

Selection options in IBeam allow the user to select stations, filters, channels, and the time range for computation of the incoherent beams. The selection routines from libXdsp are used here. The include files which contain global variables needed by these function are, therefore, included in the IBeam software. These files are found in *include/Xdsp*.

EXECUTION

Choosing the Execute option initiates the routine WExecute. First, this routines cleans out all the old entries in the database and the **.sta** beam files for the current origin and network. This means the beam relations stadisc and stapick are removed from the database. Next, a temporary file is set up in the directory defined by the environmental variable "ISEIS_TEMP_DIR" with the name "dosta.IPID.tmp" where IPID is replaced with the process id. This file is filled with the data needed for computing the beams. WExecute then starts the beam computation driver process **dostacomp**.

Beam computation is broken into two processes: mathematical computations and database access. This was done so that the computations could be done on the Stardent computer which currently does not allow access the the ORACLE database. The process **dostacomp** runs beam computation for a list of filter bands and input parameters by accessing the database, writing a scratch file "ISEIS_TEMP_DIR"/IPID.stain, and starting up the process **stacomp** on the either the SPARC or the Stardent for each filter band. Each **stacomp** process completes and starts up the process **storbmstuf** which reads the scratch file and stores the stadisc relation for the current beam in the database. When all the **stacomp** processes have completed, **dostacomp** initiates the view results process **IBeamVws.** All scratch files are deleted when they are no longer needed.

VIEW

The view option allows the user to view the time-series or the beam processing results. The time-series viewing routine in libXdsp is used to view the time series data. The beam processing results are presented by the **IBeamVws** process as indicated above.

IBeamVws starts by displaying the beam plot for the default station and filters. It uses the Menu Widget to allow for selection of different filter bands, networks, and for redisplay. The selection routines from libXdsp are used for these functions. The global variables used by the process **IBeamVws** can be found in the include file IBeamg.h and the related include files are in directory "include/Xdsp." Data initialization is done as described above.

The routine that displays the beams is ViewBms. This routine handles the control for adding filter band traces to the plot, or changing the feature that is plotted. For each filter added, the beam data for all features is read if it has not been previously read. This is controlled by the global variable *bmdatset*. The routine **DoBmsPlt** is then called to plot the data. All plots are generated using the Xyplot widget.

4.3.1.2 Automatic Mode

The automatic beam computation mode, the routine **dolBeam** is called by the automatic processing driver. This routine takes the place of the **dostacomp** process in the interactive version. This routine uses a list of default filters which are defined in the routine **defilts**. The view results display is not initiated in the automatic mode. For a more complete description refer to the description of **dostacomp** in Section 4.3.1.1.

4.3.2 Single Fourier Spectra Processing

This process computes Fourier spectra for a single time windows. The data is displayed in a two-dimensional display.

4.3.2.1 Initialization

The Single Spectrum (fsdisplay) program is started by the Spreadsheet Top Level Display in response to the SINGLE SPECTRUM option under the PROCESS menu. This process uses the Unified Process Initiation Interface which insures that the Spreadsheet is informed when it starts and when it stops. After fsdisplay has finished initial communication with the Spreadsheet and has created its Top Level shell widget, it invokes the wfidinit() routine to retrieve the waveform IDs for the given event. Subsequently, stationinit(), and chaninit() are called to obtain beams, stations and channels. In summary, the following initialization routines are used:

stationinit()	- get stations, channels, and waveforms for orid;
chaninit()	- set up default channels;
beaminit()	- read beam ids;
fsdefaultinit()	- setup default spectra selection parameters;
fsinit()	- set P phase.

4.3.2.2 Selection Events

Users select stations/channels options under the SELECT menu item. The library functions, StationSelect() and ChanSelect() obtain the user selections. These libXdsp library routines create an X window pop-up shell for station/channel selections. This library also contains routines for time range selection on beams using the Xyplots widget and correction parameter specifications. With fsdisplay, the user can respecify any default seismic phases. However, these new phases do not replace the phases in the database, but only apply to the spectra being computed. The phase information is entered into the FSDISC relation and the STAPICK phase picks remain unchanged.

The smoothing option under SELECT allows the user to select waveform smoothing parameters. Both cosine taper and Hanning windowing functions are applied to the signal and the user may specify relevant parameters for these functions in the smoothing displays. The Hanning function is actually implemented as a frequency domain convolution instead of a time domain multiplication for efficiency.

The user has the option of applying noise, instrument and von Seggern-Blandford corrections when computing spectra. When noise correction is applied, the power from a user specified noise window is subtracted from each power bin in the signal spectra, thus removing noise from the signal. Fsdisplay obtains instrument frequency response spectra from files stored under $ISEIS_CONFIG_DIR/resp. Depending upon the station used, the appropriate instrument correction is subtracted from the computed spectra, removing any instrument generated signal bias. A von Seggern-Blandford correction models the signal source as an explosion. The model explosion frequency characteristics are then removed from the computed spectra. The select SNR option allows the user to define the frequency band for signal-to-noise computations.

4.3.2.3 Viewing Events

The user has the option of viewing time-series or previously computed spectra. Fsdisplay uses standard viewing functions in the libXdsp.a library for viewing these data types. They are:

- ViewBeams() - displays incoherent beam data;
- ViewTS() - displays time-series data; and
- ViewFS() - displays spectra data.

4.3.2.4 Execution

When new single spectra are computed, any previously computed spectra are removed. The results are then displayed using the ViewFs option.

4.3.3 Continuous Fourier Spectra Processing

This signal processing research process is used to compute Fourier spectra in a series of overlapping or non-overlapping time windows. All windows are of the same duration and each window starts at a user-specified time after the previous window. The resultant data defines a surface which depicts the frequency vs. time characteristics of a given signal. The data is displayed in a three-dimensional relief type display. Static mesh and dynamic shaded sonogram plot styles are available.

4.3.3.1 Initialization

The Continuous Spectrum (cfsdisplay) program is started by the Spreadsheet Top Level Display in response to the SONOGRAM option under the PROCESS menu. This process uses the Unified Process Initiation Interface which insures that the Spreadsheet is informed when it starts and when it stops. After cfsdisplay has finished initial communication with the Spreadsheet and has created its Top Level shell widget, it invokes the wfidinit() routine to retrieve the waveform IDs for the given event. Subsequently, stainit(), and chaninit() are called to obtain stations and channels. Finally, default parameters are set and quit functions are installed. These functions are called whenever the user closes the station select pop-up. They are:

- chaninit() - retrieve new channels from database; and
- fsdefaultinit() - setup default selection parameters.

4.3.3.2 Selection Events

Users can select stations/channels using options under the SELECT menu item. The library functions, StationSelect() and ChanSelect(), are used to obtain user selections. These libXdsp. routines create an X window pop-up shell that allows station/channel selections. This library also contains routines for time range selection using the Xyplots widget and correction parameter specifications. With cfsdisplay, the user can respecify any default seismic phases. However, these new phases are not entered into the database, but only apply when new spectra are computed. The phase information is entered into the CFSDISC relation and the STAPICK phase picks remain unchanged.

The smoothing option under SELECT allows the user to select waveform smoothing parameters. Both cosine taper and Hanning windowing functions are applied to the signal and the user may specify relevant parameters for these functions in the smoothing displays. These functions facilitate smoother spectra when applied. The Hanning function is actually implemented as a frequency domain convolution instead of a time domain multiplication for efficiency.

The user has the option of applying noise, instrument and von Seggern-Blandford corrections when computing spectra. When noise correction is applied, the power from a user-specified noise window is subtracted from each power bin in the signal spectra, thus removing noise from the signal. Cfsdisplay obtains instrument frequency response spectra from files stored under $ISEIS_CONFIG_DIR/resp. Depending upon the station used, the appropriate instrument correction is subtracted from the computed spectra, removing any instrument generated signal bias. A von Seggern-Blandford correction models the signal source as an explosion. The model explosion frequency characteristics are then removed from the computed spectra.

4.3.3.3 Viewing Events

The user has the option of viewing time-series or previously computed continuous spectra. Cfsdisplay uses standard viewing functions in the libXdsp.a library for viewing these data types. They are:

- ViewTS() - display time-series data.
- When the user wishes to view previously generated continuous spectral data, cfsdisplay starts the 'cfsplot' program via the prExecuteNoWait() Unified Process Initiation Interface. The cfsplot program always starts as an icon and the icon must be clicked upon to enable the display.
- From the cfsplot program, the user has the option of viewing either a Mesh plot, which appears on the same display as the cfsplot display, or a shaded sonogram display which appears as a DORE application on the Stardent system.

The mesh plot display cannot be dynamically viewed and it is rendered as a hidden line image at a fixed viewing angle. It provides a useful means of ·wing continuous spectral data when the Stardent system is unavailable. In addition, some analysts prefer non-shaded renderings over shaded ones.

When the mesh plot option is selected, a pop-up shell is created and the data is rendered using LYNESS widgets in the window. A floating horizon type of hidden line algorithm is used. Basically, the data is plotted front-to-back and any plot lines which fall below the trace (which lies in front of the current line) are not drawn. The algorithm does have certain limitations and artifacts are sometimes apparent in the final plot.

When the DORE option is selected from the VIEW menu, cfsdisplay uses prExecuteNoWait() to start the Sono3D shaded sonogram viewing program.

4.3.3.4 The Shaded Sonogram Program

The Stardent resident sonogram program allows dynamic viewing and manipulation of continuous spectral sonogram data. The program uses the Stardent DORE graphics package. This is a hierarchical object-oriented graphics package, which has built-in shading and hidden line removal capabilities.

Using parameters supplied in the command line by the cfsplot program, Sono3D loads the data and uses it to create a DORE TriangleMesh object. This essentially defines the data surface. Other data in the command line is used to create and annotate the plot axis objects. Sono3D sets up the initial viewing transformations and light source information. The Ggrph3D widget in libXe3d.a is used to create the plot and utility functions in libXeu.a are used to affect different viewing transformations.

During initialization, Sono3D determines if color table interaction was desired (-c option) and if default user initial viewing angles were specified. Default viewing angles are specified as -o X, Y, Z where X, Y, and Z are the rotation angles of the X, Y, and Z axes, respectively. The angles are given in positive and negative degrees. The default angles are 30, -30 and 0 degrees. If Color Editor interaction was desired, Sono3D first loads the *.cfs data and creates all widgets needed for the display. If colors were desired, Sono3D attempts to load color table information from the RGB Color Editor. The program does this by sending a

LOAD_DEFAULT_COLOR_REQ message to the RGB color editor and when the color editor responds with a NEW_COLORS message, Sono3D loads the new colors into the three-dimensional plot widget. If the Color Editor process is not running, a local default set of colors is used. DORE allows individual colors to be assigned to each vertex in a TriangleMesh object. The graphics package then blends these colors together, using Gouraud shading. The location and direction of the light source affects brightness. Sono3D determines the amplitude range of data and assigns colors to vertices based upon the vertex amplitude. Whenever a Sono3D process is started with the -c option, LOAD_DEFAULT_COLOR_REQ messages are sent to the color editor. The Color editor changes its axis parameters to those of the client process. In other words, the color editor axis will be changed to look like the amplitude axis of the last Sono3D process to startup or request color table information.

The UPDATE COLOR TABLE option under the EDIT menu causes Sono3D to send LOAD_COLOR_REQ messages to the Color Editor which responds with its current color table. Similarly, the UPDATE COLOR EDITOR option loads the current color table of the Sono3D process into the Color Editor. Each Sono3D process has its own private color table which is sent to the Color Editor every time the UPDATE COLOR TABLE option is selected.

The VIEW menu is used to set the viewing mode. ORBIT, PAN, ZOOM, DEFAULT and TWIST options are available. These allow real-time rotation panning, zooming, default view and twisting of the model. Sono3D does an XtSetValues() function call to set the navigation mode in the Ggrph3d widget. This sets the motion context in the widget so that the widget knows how to respond when the mouse is dragged in the plot window. Navigation is performed by changing the DORE viewing parameters. In particular, view plane normal vector, the view up vector, the view reference point and the clipping window parameters may be changed, depending upon which of the options is selected. Viewing parameters in DORE are objects just like spheres, cylinders, etc. DORE allows addition, replacement and deletion of objects. The replacement of view objects changes the view.

The MARKING OPTIONS menu is used to define how points on the surface are marked when the MIDDLE mouse button is clicked in the plot area. Marking is used to query frequency, time and amplitude values at specific locations on the frequency/time surface. The point is marked and the 3D coordinates of the point are displayed right underneath the menubar. Depending upon which marking option was selected, the point will be marked with a line, an X plane, Y plane, Z plane, or all three planes. Plane markers are transparent so that data may be viewed through them. The point of intersection is calculated by shooting a ray parallel to the viewing vector and determining where it intersects on the surface. The ray is shot from the point where the user clicked the MIDDLE mouse button. It is possible that the ray will miss all facets on the surface. If this is the case, nothing is done. However, if an intersection is found, line, or plane objects are positioned at the point of intersection and the plot is redrawn.

4.3.3.5 Execution

When new continuous spectra are computed, any previously computed data are removed and then the new continuous spectra are computed in the background by the cfspect process. This process can run on any machine. When it is finished, it starts the endcfspect process to update the CFSDISC relation in the database with the new data file name. This process then starts the cfsplot process as an icon to signify the completion of execution. The user may click on this icon to enable the cfsplot display.

4.3.4 Dynamic Time Warp Processing

The display process **DTWDisplay** is the driver for the interactive dynamic time warp processing. The processing can be done for selected filter bands, reference events and phase windows, or for a default set. The automatic DTW processing simply runs the default set of filters and references. The execution section below describes this in further detail.

DTWDisplay has Selection, View and Execution options. It uses the Menu Widget to get to the callbacks for the various options. Viewing results takes the user to the DTWVws process. This process uses the same data initialization routine and globals as DTWDisplay. For a list and description of the global variables used by DTWDisplay see the DTWDglobals.h. include file and the related includes in "include/Xdsp" (the libXdsp include directory).

4.3.4.1 Data Initialization

Data initialization is done partially on startup and partially as needed. On startup the DTW display processes find the networks available for a given origin ID (routine getstas - see libdbio). A default network is then set (global variable "sta"). Whenever a new default network is set routine "datainit" is called. This routine gets the filter bands and reference events available. In the case of DTWDisplay, these are all the filter bands processed for the origin ID and all of the reference events in the same region up to a maximum of MXRF (defined in include file DTWDglobal.h). For DTWVws, the results display, all filters and reference event pairs which have DTW results available will be stored. The process calling the "datainit" routine is identified by the value of the global variable iViewRflag (see DTWDglobal.h). Next, "datainit" gets the beam IDs for the origin ID and filter band combinations. Event description information is initialized in the results display case. Routine "datainit" also sets the default values for the filters and reference events to use and initializes the number of points in the beams and the boolean data availability marking arrays. Beam data is read in by the display routines as needed.

4.3.4.2 Selections

Selection options in the DTW display processed allow the user to select filters, networks, seismic phases, the reference events. The selection routines from libXdsp are used. The reference selection widget as specified is specific to the DTW display processes. It simply stores the list of chosen reference origin IDs in the case of DTWDisplay, or stores the reference origin IDs and initializes the reference beam data for process DTWVws.

4.3.4.3 View

The view option allows the user to view the beams for the current origin ID or to view the processing results. The library routine viewBeams in libXdsp is used to display the beams. Selections the view results options initiates the results display, DTWVws.

The results display DTWVws uses the selection and data initialization procedures described above. It uses the same global variable file as DTWDisplay, which is DTWDglobal.h. It has options for viewing the results in a number of different ways. For each display method, there is a "view" routine which handles the control and a "DoPlt" routine which does the actual plotting. For example, the "View Results by Similarity" option first calls the routine "ViewSim" to set up the control for the display, then "ViewSim" calls "DoSimPlt" to produce the actual plot. For a description of the types of displays available, see the DTW Users Guide.

4.3.4.4 Execute

There are two execute options for DTW processing: "Execute Automatic Defaults" and "Execute Selections." For the automatic defaults case, the callback routine "ExecuteAll" is called. This routine starts process "DTW_main" which is the same process started by the automatic processing driver for the ISEIS system. DTW_main gets the automatic default filter bands, which are defined in routine getautofilts (see libdbio), and the first MXRF (defined in DTW_main) reference events returned by routine getrefvts (see libdbio). It then runs the beam matching process, "dtw", for each filter band and reference event combination. The "dtw" process accesses data from the stadisc and stapick relations. It outputs all the match data to the proper database relations (dtw and dtwstat). For a description of the relations see Appendix C.

For the "Execute Selection" case, the callback routine "ExecuteS" is called. This routine sets up a scratch file (dodtw.IPID.tmp where IPID is the process ID) in the directory defined by the environmental variable ISEIS_TEMP_DIR. This scratch file contains a list of the chosen filter bands and reference events. Routine "ExecuteS" then starts process doDTW and returns. Process doDTW reads the scratch file and calls "dtw," the beam matching program, for each filter band and reference event combination. After all of these have been processed, doDTW initiates the view results display (DTWVws). The scratch file is deleted when it is no longer needed.

4.3.5 Continuous FK Processing

This signal processing research process is used to compute continuous FK spectra in a series of overlapping or non-overlapping time windows. All windows are of the same duration and each window starts at a user specified time after the previous window. The resultant data define a "template" which depicts the FKx vs. FKy characteristics of a given signal as a function of time.

4.3.5.1 Initialization

The Continuous FK (cfkdisplay) program is started by the Spreadsheet Top Level Display in response to the CONTINUOUS FK option under the PROCESS menu. This process uses the Unified Process Initiation Interface, which insures that the Spreadsheet is informed when it starts and when it stops. After cfkdisplay has finished initial communication with the Spreadsheet and created its Top Level shell widget, it invokes the wfidinit() routine to retrieve the waveform IDs for the given event. Subsequently, stainit(), and chaninit() are called to obtain stations and channels. Finally, default parameters are set and quit functions are installed. These functions are called whenever the user closes the station select pop-up. They are:

chaninit()	- Retrieve new channels from database;
InitializeWindowParams()	- sets window lengths, number, overlaps; and
InitializeFreqParams()	- Use default frequency parameters.

4.3.5.2 Selection Events

Users can select stations/channels using options under the *SELECT* menu item. The library functions, StationSelect() and ChanSelect(), are used to obtain user selections. These libXdsp.a routines create an X window pop-up shell that allows station/channel selections. This library also contains routines for time range selection using the Xyplots widget and correction parameter specifications.

The user also has the option of selecting which frequency bands to view in the cfkplot process. Thus, previous FK data is computed and saved for individual frequency bands and can be viewed later by selecting the appropriate band.

4.3.5.3 Viewing Events

The user has the option of viewing time-series or previously computed FK data. Cfkdisplay uses standard viewing functions in the libXdsp.a library for viewing these data types. They are:

- ViewTS() - display time-series data.
- When the user wishes to view previously generated FK data, cfkdisplay starts the 'cfkplot' program via the prExecuteNoWait() Unified Process Initiation Interface. The cfkplot program always starts as an icon and the icon must be clicked upon to enable the display.

From the cfkplot program, the user can view four windows of data that shows time, azimuth, velocity and F-Statistic information. The user selects on any plot to cause selections to appear on the other three plots.

4.3.5.4 Execution

When new continuous FK templates are computed, any previously computed templates are removed and the new continuous FK templates are computed in the background by the cfkexecute process. In cfkdisplay, the new *.cfk file name is entered into the CFKDISC relation prior to actual execution. The cfkexecute process performs the actual FK processing. The end result of this processing is a *.cfk file which contains time, azimuth, velocity and F statistic data for each window in the waveform. This is the data which is displayed by cfkplot in four separate windows. When the cfkexecute program has completed, it starts cfkplot as an icon. The cfkexecute program uses a $ISEIS_CONFIG_DIR/ hires.in file to specify plot contour resolutions.

4.3.6 Amplitude Ratio Processing

This signal processing discriminant support process computes seismic phase ratio statistics which are later utilized by CLIPS rule processes for discrimination purposes. Both an interactive (ardisplay) and automatic process (arauto) exist.

4.3.6.1 Initialization

The Amplitude ratio (ardisplay) program is started by the Spreadsheet Top Level Display in response to the AMPLITUDE RATIOS option under the PROCESS menu or an option in a Summary display. This process uses the Unified Process Initiation Interface, which insures that the Spreadsheet is informed when ardisplay starts and when it stops. Now ardisplay performs the following:

1. getstas() - obtain all stations where event arrived and select default;
2. beaminits() - load all relevant incoherent beams;
3. phaseinit() - obtain all defined seismic phases; and
4. arrinit() - obtain arrival picks.

Finally, default parameters are set and quit functions are installed. These functions are called whenever the user closes the station select pop-up. They are:

- beaminits();
- phaseinit(); and
- arrinit().

Now ardisplay creates the top level shell widget and menubar and the display appears.

4.3.6.2 Selection Events

Users can select stations/filters using options under the SELECT menu item. The library functions, StationSelect() and SelectFilters(), are used to obtain user selections. These libXdsp routines create an X window pop-up shell that allows station/filter selections. When Ratio Selection is chosen, ardisplay creates a pop-up which allows the user to specify numerator and denominator phases. This actually determines what particular type of ratios are computed and entered into the AMPRATIO relation. The phase selection option allows the user to redefine seismic phases and these new phase selections are entered into the STAPICK relation. Once in the phase selection display, the user has the option to select incoherent beam filter sets and subsequent phases for each beam. Any previously defined phases are displayed and marked.

Selection of the SNR option causes ardisplay to access the SNRIB relation and then display an SNR display. This display allows the user to compute signal-to-noise for each incoherent beam and create entries in the SNRIB relation. Each beam is displayed with marked phases and background noise lines. These default noise floors were derived from the noise windows specified in the incoherent beam calculation. The user has the option of dragging to define a noise floor if the default noise floors are not acceptable. SNR is computed per phase according to the method selected on the form. The MAX option divides the maximum value in the given phase by the noise floor value and the AVG option divides the average amplitude in the phase by the noise floor value.

4.3.6.3 Viewing Events

The user has the option of viewing results by REGION, DISTANCE and FREQUENCY. Each option first enables a Legend widget containing all of the currently defined regions in the REFREGION relation. The user may select as many regions as desired before selecting DONE. Ardisplay now enables another selection pop-up which consists of a collection of Legend widgets. The user may select default FILTERS, RATIOS, METHOD and plot types for the VIEW by REGION option. Options selected here determine how many REGION vs. RATIO plots will appear. Plots are created using the LYNESS calcomp emulation plotting package. The REGION plot shows the selected RATIO vs. each REGION selected in the REGION pop-up. Each event is represented by a classification symbol with error bars.

The VIEW by DISTANCE option displays a DISTANCE vs. RATIO plot, which displays event classification symbols without error bars.

The VIEW by FREQUENCY option displays frequency bands against the ratio type selected in the intermediate display.

The RB*.c files contain code, which is used to display the intermediate forms, which allow the user to select FILTERS, RATIOS, METHODS and PLOT TYPES. These forms simply define the environment to be used for the LYNESS plots which will be subsequently invoked.

4.3.6.4 Execution

When new phase ratios are computed, ardisplay computes them according to the method specified. If the AVERAGE method is specified, the values within the phase windows are averaged prior to phase computation by division. However, if the MAXIMUM method is selected, then the maximum values in each phase are first determined and then the ratio is computed. Ratios are computed for all NUMERATOR / DENOMINATOR values listed in the SELECT/RATIOS display. All ratios are entered into the AMPRATIO display for subsequent discrimination by the amplitude ratio PSratio.* discrimination processes.

4.3.7 Spectral Ratio Processing

This signal processing discriminant support process computes seismic spectral ratio statistics, which are later utilized by CLIPS rule processes for discrimination purposes. Both an interactive (srdisplay) and automatic process (srauto) exist.

4.3.7.1 Initialization

The Spectral ratio (srdisplay) program is started by the Spreadsheet Top Level Display in response to the SPECTRAL RATIOS option under the PROCESS menu or an option in a Summary display. This process uses the Unified Process Initiation Interface, which insures that the Spreadsheet is informed when srdisplay starts and when it stops. Now srdisplay performs the following:

1. assocstainit() - obtain all stations where event arrived and select default;
2. FSphaseinit() - load spectra for default station;
3. ratioinit() - set method names and default ratios;
4. ampinit() - set amplitude names and default amplitude; and
5. corrinit() - initialize correction parameters.

Finally, default parameters are set and quit functions are installed. These functions are called whenever the user closes the station select pop-up. They are:

- FSphaseinit()
- ratioinit()

Now srdisplay creates the top level shell widget and menubar and the display appears.

4.3.7.2 Selection Events

Users can select stations using an option under the SELECT menu item. The library function StationSelect() is used to obtain user selections. This libXdsp.routine creates an X window pop-up shell that allows station selections. When phase name Selection is chosen, srdisplay creates a pop-up with a Legend widget, which contains a list of the current phase names. It is used to allow the user to select the seismic phases for spectral processing. When the frequency selection option is chosen, srdisplay creates a pop-up which allows the user to specify numerator and denominator frequency bands. The user can either enter the frequency bands via the keyboard or he can select the drag option which results in the display of the spectra on a phase per phase basis. By dragging on the Xyplots display, the user can specify both numerator and denominator frequency bands. These numerator/denominator frequency bands are eventually used to compute the actual spectral ratios which will be entered into the SPECRATIO relation.

The user has the option of applying noise, instrument and Q corrections when computing spectra. When noise correction is applied, the power from a user-specified noise window is subtracted from each power bin in the signal spectra, removing noise from the signal. Srdisplay obtains instrument frequency response spectra from files stored under $ISEIS_CONFIG_DIR/resp. Depending upon the station used, the appropriate instrument correction is subtracted from the computed spectra, removing any instrument-generated signal bias. Q correction removes propagation bias from the signal. The PATH relation contains region-dependent propagation parameters which are used to perform Q corrections.

4.3.7.3 Viewing Events

The user has the option of viewing results by REGION and DISTANCE. Each option first enables a Legend widget containing all of the currently defined regions in the REFREGION relation. The user may select as many regions as desired before selecting DONE. Srdisplay now enables another selection pop-up, which consists of a collection of Legend widgets. The user may select default PHASES, RATIOS, METHOD, CORRECTIONS and plot types for the VIEW by REGION option. Options selected here determine how many REGION vs. FREQUENCY plots will appear. Plots are created, using the LYNESS calcomp emulation plotting package. The

REGION plot shows the selected RATIO vs. each REGION selected in the REGION pop-up. Each event is represented by a classification symbol with error bars.

The VIEW by DISTANCE option displays a DISTANCE vs. FREQUENCY RATIO plot. which displays event classification symbols without error bars.

The SRrb*.c files contain code which is used to display the intermediate forms which allow the user to select FILTERS, RATIOS, METHODS and PLOT TYPES. These forms simply define the environment to be used for the LYNESS plots which will be subsequently invoked.

4.3.7.4 Execution

When new spectral ratios are computed, srdisplay computes them according to the method specified. If the AVERAGE method is specified, the values within the frequency windows are averaged prior to ratio computation by division. However, if the MAXIMUM method is selected, then the maximum values in each frequency band are first determined and then the ratio is computed. Ratios are computed for all NUMERATOR/DENOMINATOR values listed in the SELECT/RATIOS display. All ratios are entered into the SPECRATIO relation for subsequent discrimination by the spectral ratio SRratio.* discrimination process.

4.3.8 Ripple-Fire Processing

This signal processing discriminant support process computes and extracts parameters for characterizing and identifying economic chemical explosions which exhibit delayed firing. Mine explosions of this type produce regional phase spectra with identical spectral modulations. The Fourier transforms of log spectra of seismograms, called cepstra, have peaks at the same "quefrency" for two or more phases. If well-defined peaks can be identified at the same quefrency for two or more phases, the event can be identified as a multiple event and is very likely an economic explosion.

4.3.8.1 Initialization

The Ripple-fire (MERSYDPY) display program is started by the Spreadsheet Top Level Display in response to the RIPPLEFIRE option under the PROCESS menu or an option in a Summary display. This process uses the Unified Process Initiation Interface, which insures that the Spreadsheet is informed when MERSYDPY starts and when it stops. Now MERSYDPY performs the following:

1.	doMinit()	- Initialize structures;
2.	SelectPhaseCep()	- Select peaks for cepstra processing;
3.	ViewFs()	- View associated Fourier spectra;
4.	ProcessQuefSlice()	- Perform cepstra processing; and
5.	DoneCepPeaks()	- Save results of cepstra processing.

The default parameters are set and quit functions are installed. These functions are called whenever the user closes the station select pop-up.

4.3.8.2 Selection Events

Users can select stations using an option under the SELECT menu item. The library function StationSelect() is used to obtain user selections. This libXdsp routine creates an X window pop-up shell that allows station selections. When phase name Selection is chosen, MERSYDPY creates a pop-up with a Legend widget, which contains a list of the current phase names. It is used to allow the user to select the seismic phases for spectral processing. When the frequency selection option is chosen, MERSYDPY creates a pop-up which allows the user to specify numerator and denominator frequency bands. The user can either enter the frequency bands via the keyboard or he can select the drag option which results in the display of the spectra on a phase per phase basis. By dragging on the Xyplots display, the user can specify slices for quefrency processing. These slices are eventually used to compute the actual cepstra which will be entered into the CEPPKS relation.

4.3.8.3 Viewing Events

The user has the option of viewing results by REGION and DISTANCE. Each option first enables a Legend widget containing all of the currently defined regions in the REFREGION relation. The user may select as many regions as desired before selecting DONE. MERSYDPY now enables another selection pop-up, which consists of a collection of Legend widgets. The user may select default PHASES, RATIOS, METHOD, CORRECTIONS and plot types for the VIEW by REGION option. Options selected here determine how many REGION vs. FREQUENCY plots will appear. Plots are created, using the LYNESS calcomp emulation plotting package. The REGION plot shows the selected RATIO vs. each REGION selected in the REGION pop-up. Each event is represented by a classification symbol with error bars.

The VIEW by DISTANCE option displays a DISTANCE vs. FREQUENCY RATIO plot, which displays event classification symbols without error bars.

The SRrb*.c files contain code which is used to display the intermediate forms which allow the user to select FILTERS, RATIOS, METHODS and PLOT TYPES. These forms simply define the environment to be used for the LYNESS plots which will be subsequently invoked.

4.3.8.4 Execution

When new cepstra are computed, MERSYDPY initiates the process MERSY to compute them according to the method specified. The MERSY process computes two types for cepstra, one consisting of standard Fourier transforms of the spectra, and the other using a maximum entropy algorithm to enhance the processing results. The cepstra are then examined to find potential cepstral peaks which may indicate multiple firings. These peaks are stored in the database and the results are displayed by MERSYDPY.

5.0 AUTOMATIC PROCESSING DETAILS

We developed the ISEIS Automatic Process in order to facilitate operator use of the system. The purpose of the automatic process is to execute the discrimination processing in the background, thereby freeing the operator to evaluate the results. The function of the operator is intended to be one of evaluation, refinement and confirmation, and the function of the system is to perform the routine analysis to generate the results for this evaluation.

The automatic process has been developed to perform all of the routine data assessment, generation, and evaluation required to generate the composite event identification for evaluation by the operator. As such, the automatic process must execute all of the data generation and discrimination processes in the background for each new event which is generated by the IMS process. The automatic discrimination process is designed to execute periodically. This period is adjustable by the ISEIS system administrator. When the process is activated, it first queries the database for any events which have been generated by the IMS system since the last execution of the automatic discrimination process. These events are evaluated to determine whether sufficient auxiliary data exists for discrimination processing. All events which have sufficient data are processed by the ISEIS system and the results are saved for later review by the operator. Once all current events are processed, the automatic discrimination process suspends itself until the next period expires.

5.1 PROCESS HIERARCHY

The Automatic Discrimination Process consists of a driver which simply executes each of the automatic discrimination processes in turn for each new event in the IMS database. The Automatic Discrimination Process itself extracts origin information from the database for each event which has been generated and evaluates the data status for each event. Data status evaluation is similar to the evaluation done by the discriminants as described in Section 3.0; however, this evaluation is concerned with very basic data such as origin information, arrival information, and waveforms. This evaluation simply queries the database to verify that these basic elements exist, and it does not use CLIPS or rules for the evaluation. The evaluation process compiles a list or origins which are then submitted to the discrimination processing.

In order for ISEIS to perform discrimination processing on an event, a significant amount of data not generated by the IMS must be generated by the automatic discrimination process. This includes incoherent beams, spectra, cepstra, and amplitude ratios. This data is generated by a set of signal processing tasks which the automatic discrimination processes executes prior to invoking the discrimination routines. These signal processing tasks are outlined below.

5.1.1 Incoherent Beam Processing

Once the automatic processing driver has determined which events have sufficient data to proceed with the discrimination processing, the first step in this processing is to generate incoherent beams for the event. The software to perform this processing is contained in the directory:

/iseis/common/src/auto/src/stacompdrv.

This directory contains the main module for the process as well as copies of certain modules which are unique to the automatic incoherent beam processing. The software in this directory is based on the interactive incoherent beam software which is stored under the **/iseis/common/src/sigpro** directory and described in Section 4.0 of this document. Both sets of software use libraries stored in **/iseis/common/libsrc** for accessing the database, and performing general signal processing computations.

5.1.2 Automatic Phase Controller

Once the incoherent beams are computed, the automatic controller invokes the automatic phase selection software. This function uses the incoherent beams to generate automatic phase selections for the event. The software uses the IMS-generated phase selections as initial estimates of the phase time and uses the incoherent beam to refine the phase time selection.

The source code for this function consists of one file **autoPhase.c** which is stored in the directory:

/iseis/common/src/auto/src/PhaseSel.

This function also uses general database access routines stored in the global **libsrc** directory.

5.1.3 Fourier Spectra Generation

The next step in the automatic discrimination processing is to generate spectra for the phases identified in the automatic phase selection process. This spectra is used both for amplitude and spectral ratio processing and for the cepstral processing used in the Ripple-fire discriminant.

The software for the automatic spectral generation is stored in FSauto.c in the directory:

/iseis/sparc/src/sigpro/src/spectra/src

along with the interactive spectral processing software.

5.1.4 Amplitude and Spectral Ratio Computation

The last two steps in the signal processing required for automatic discrimination are Amplitude and Spectral Ratio computation. Each of these functions computes ratios of amplitudes for the various phases identified for the event. These ratios are stored in the database for later use by the discriminants. The software for each of these functions is stored in the corresponding interactive signal processing directories.

5.1.5 Automatic Data Assessment

Once all of the signal processing has been completed, the automatic discrimination controller invokes each of the discriminant data assessment processes in turn. These processes are described in Section 3.0 of this manual. The controller simply invokes the data assessment processes in parallel and waits for each to respond with a completion message.

5.1.6 Automatic Discrimination Processing

After all discriminant data assessment processes have responded with a completion message, the automatic discrimination controller queries the database for each of the discriminants. The appropriate case-based and model-based processing is invoked for each discriminant which has a complete data status. These processes are also invoked in parallel and the automatic controller again waits for a completion message from each.

Once all discriminants have been completed, the automatic discrimination controller performs the data assessment for the discriminant composite process and if a complete status is returned, invokes the composite discriminant process. Once this process is completed, the automatic discrimination controller updates the timestamp and begins the processing cycle on the next available event. Once all new events have been processed, the automation controller hibernates until the next activation period.

5.2 DATABASE ISSUES

In addition to the standard database relations, such as origin, arrival, wfdisc, etc., and the discrimination database relation, disctech and ruledisc, the automatic discrimination process also uses one other relation to control its own processing. This relation is the **timestamp** relation. This relation is used by many of the NMRD systems to keep track of the last epoch processed by each individual process. The automatic discrimination process records the time of the last epoch processed by the ISEIS system and uses this timestamp to determine which events in the IMS database are "new." All events which have origin times later than the last ISEIS timestamp are considered new events and are considered for processing by the automatic discrimination process.

6.0 EXTERNAL AND INTER-PROCESS INTERFACE DETAILS

Inter-process communication and process initiation is handled by the ISIS interprocess communication package, the Dispatcher inter-process communication package and the ISEIS system Monitor processes.

6.1 DISPATCHER IPC INTERFACE

The Dispatcher software will not be documented here. Dispatcher Documentation can be found separately. However, Dispatcher implementation details will be discussed here.

Both event-driven and timed Dispatcher interfaces are utilized by ISEIS. Asynchronous applications are used where both X window events and message arrival events need to be serviced on an equal priority. The Top Level Spreadsheet process must be able to process user-generated X window events, as well as incoming messages from discriminant processes. The ISEIS color editor, the Stardent resident Shaded Sonogram Display and the Interactive Map process are examples of processes which must utilize event-driven Dispatcher events. After a d_open() call to initialize the Dispatcher socket, processes invoke get_fd() to obtain the UNIX file descriptor associated with the socket and then an Athena toolkit XtAddInput() call which associates the file descriptor with a message processing callback function. This allows Dispatcher message arrival events to be processed just like other X window events. When a message arrives, the process performs a d_listen() to read the message. However, most processes just use d_send() and d_listen() in a sequential manner.

6.2 UNIFIED PROCESS INITIATION SOFTWARE

All processes in ISEIS which need to start other processes use library functions which take care of the details involved in process execution. Process execution within ISEIS can be somewhat complicated. The UNIX environment must be queried to determine how many systems are members of the ISEIS network, the host names and system types of each system must be determined and the cpu load limits for each system must also be obtained. Then the MachAlloc file entry for the process must be retrieved to determine the X window display that the process will use, and the preferred and required systems where it can run. Once this information is obtained, the software sends GET_LOAD_REQUEST messages to each of the ISEIS Monitors and waits for their load replies. Based upon MachAlloc, environmental and cpu load information, the interface software determines:

1. Are system loads low enough for the process to run?
2. On what system will the process run?
3. What X window display will the process use?

If the systems are too busy and execution is denied, an error message is returned by: prExecuteWait(), prExecuteNoWait(), prExecuteWaitWithParent(), prExecuteNoWaitWithParent(), or pr_com_execute().

Any NULL error returns means that execution succeeded. If an error was returned, processes search the error string for the words "Fully Loaded" and retry later if found. General purpose system information and process execution functions are of the form pr*() and sy*().

A system is selected from the network in the following manner. If the REQUIRED_SYSTEM column for a given entry in the MachAlloc file is not set as 'Any' and it is not set to 'SUN,' then the given process can only run on the system specified. Load limits are obtained and the process is executed on that system if the loading is low enough. However, if the field is set to 'SUN,' that means that the process must run on Sun3 and Sun4 workstations. The set of valid Sun workstation host names is specified in the environment variable ENV_ISEIS_SUN_HOSTS. Now the PREFERRED_SYSTEM column can also determine system selection. An 'Any' entry means that the system will try to select the least busy Sun type system for execution. If a specific Sun host member is specified, the system will run it on that system, assuming that the process load on that machine is not too great. It is also possible to set the REQUIRED_SYSTEM entry to 'Any' and the PREFERRED_SYSTEM entry to 'SUN' meaning that the process can run anywhere, but all attempts should be made to first run it on Sun type workstations. The most important point to note here is that the PREFERRED_SYSTEM specification should always be a subset of the REQUIRED_SYSTEM specification. Another important point is that the MachAlloc file specifies where processes can run. So, processes which require access to ORACLE database facilities should not be configured to run anywhere, since only one system or a group of systems running SQLNET may have database access capability. The Stardent graphics workstations do not support ORACLE or SQLNET.

In summation, this interface library handles all of the details of system selection, X window display selection, system cpu load queries, and eventual process execution. Any functions of the form prExecute*Wait*() utilize the rsh system interface and functions of the form prExecute*NoWait*() start processes by sending PROC_START_REQ messages to the system Monitors.

6.3 ISEIS MONITOR PROCESSES

An ISEIS system Monitor process runs on each system which is a member of the ISEIS network and performs three basic functions:

1. Periodically queries system loads with the UNIX uptime command;
2. Services START_PROC_REQ process startup request messages; and
3. Provides an ISIS/DISPATCHER IPC interface.

Periodically, each Monitor invokes a UNIX 'uptime' system call to query the current system loading statistics. When a GET_SYSTEM_LOADS message arrives from a client process, the Monitor returns its current value for the system, minute-by-minute cpu load. This load query time is specified in seconds by the environment variable ISEIS_LOAD_CHECK_INTERVAL. Too short a time may cause Monitor performance to degrade because it is constantly checking the load and too long a time may result in out of date and inaccurate load statistics.

Upon receipt of a START_PROC_REQ message, the Monitor invokes a UNIX system call using the arguments in the message. This effectively starts the process.

Each Monitor also supports an ISIS/DISPATCHER IPC message routing facility. Most of the NMRD systems now use an IPC package called ISIS whereas ISEIS uses the older Dispatcher IPC facilities. Each Monitor can receive an ISIS message bound for an ISEIS process or it can receive an ISEIS message bound for an ISIS process. This facility is described in greater detail in the next section.

6.4 ISIS INTERFACE TO OTHER NMRD PROCESSES

All non-ISEIS NMRD systems use a communications package known as ISIS. The ISEIS system Monitors provide an interface between ISIS and Dispatcher communications packages. Any NMRD process that wishes to communicate with an ISEIS process should send a message to any one of the system Monitors. This message is:

```
ISIS_DISPATCHER iseis_process_addr isis_process_addr {data .... }
```

where ISIS_DISPATCHER is the message id, iseis_process_addr is the desired address of the iseis process and 'isis_process' is the name of the ISIS process address. The data can be in any format as long as it is comprised of an ASCII string less than 1024 bytes.

A 'DISPATCHER_ISIS' message id is used to send messages from an iseis process to an ISIS process. Some of the more common iseis_process_addr names are listed below:

dsmain.	Any	- Dispatcher address for the Top Level Spreadsheet Display
map.	Any	- Dispatcher address for the ISEIS Interactive Map Display
RGBcolor.	Any	- Dispatcher address for the ISEIS Color Editor process

ISEIS Monitor Dispatcher addresses are formed by appending 'Monitor.Any' onto the system host name where the Monitor is running. For example, if ARS sends a list of event ids to the ISEIS Top Level Display, the message via ISIS would be:

```
send_message("honerMonitor.Any",ISIS_DISPATCHER,
                    "LOAD_EVENTS ARS dsmain.Any 2 10011 10012");
```

meaning that the Spreadsheet should add two events with ids 10011 and 10012.

An ISEIS process could send a message to an ISIS process in a similar way:

```
d_send("trymMonitor.Any","DISPATCHER_ISIS",
                        "LOAD_EVENTS dsmain.Any ARS 1 2100003",40);
```

which would send one orid to ARS. Any Monitor can be used, the choice being strictly arbitrary.

For ISIS - DISPATCHER interfacing to work properly, the Dispatcher, ISIS, system Monitors and one ISIS agent process must be running. Agent processes are not part of the ISEIS package, but they should be available as part of the NMRD system software.

7.0 INSTALLING AND CONFIGURING ISEIS

7.1 ISEIS DIRECTORY HIERARCHY

The ISEIS directory hierarchy is set up to ease installation. The root directory starts at ./iseis and it contains the 'data,' 'tools,' 'tmp' and 'common' subdirectories, as well as the .cshrc, .login, ISEIS configuration files and the X window defaults files. The ./iseis/data subdirectory contains ISEIS configuration and parameter files, such as the MachAlloc file. It also contains the subdirectories: 'MAPS,'.'explanations.' 'kbases,' 'general,' 'overlays,' 'resp' and 'sumprocs.' MAPS and all of its subdirectories are used by the Interactive Map process for configuration purposes. The 'overlays' directory is also used by the Map when it displays cities, geopolitical boundaries, etc. The 'explanations' subdirectory has subdirectories for each discriminant process in ISEIS and these contain explanation and dribble files (see 2.1.5.1) which are produced by the discriminant processes. The 'kbases' subdirectory contains the rules files which are executed by CLIPS during the rule-based processing phase of discriminant execution. The 'general' subdirectory contains general purpose filter coefficients and other ad hoc parameter files which are used by the signal processing applications. The 'resp' directory contains instrument response parameter files. Files in the 'sumprocs' directory are accessed by the Top Level Spreadsheet display when summaries are enabled (see 2.1.5.1). The 'tools' directory contains the Makefile and source needed to build the mimic utility. Mimic should be available with the NMRD software, but it was included in the ISEIS tree in case it is not. This utility is used during ISEIS installation. Any process that creates temporary scratch files places them into the ./iseis/tmp directory. The ISEIS_WORKSPACE_DIR environmental variable indicates the path name to this directory. Similarly, ISEIS_CONFIG_DIR specifies a path name to the ./iseis/data directory. The system administrator can change the locations of these directories if he also changes the environmental variables (see Section 2.3.1). The 'common' subdirectory contains all of the ISEIS source code with the necessary Makefiles to compile and link. Every subdirectory under the './iseis/common' directory can have one or more of the standard subdirectories 'include,' 'libsrc,' 'lib,' 'src,' 'SCCS,' 'bin' and 'doc' plus additional subdirectories. Every directory has a purpose:

include:	contains include files (*.h) used to specify custom data types, constants and externs.
libsrc:	contains source code used for the generation of libraries. These may contain *.r, *.c, *.f, *.F, *.pc and *.y files.

src: contains source code used for the generation of executable applications and utilities. *.r, *.c , *.f, *.F, *.pc and *.y files may reside here. The file types are explained below:

- *.r - RATFOR source code.
- *.c - C language source code.
- *.f - FORTRAN source code.
- *.F - FORTRAN source code that needs to be processed by the cpp C precompiler before compilation. This allows Fortran include files to be in separate directories.
- *.pc ORACLE embedded SQL source used for database operations. Files are precompiled by PCC and then compiled with CC.
- *.y These are yacc source code files. These are input to the YACC (yet another compiler compiler) to convert to C and then they are compiled with CC.

lib: contains complete UNIX libraries (*.a).

bin: contains executable UNIX processes.

doc: contains ISEIS documentation and MAN pages.

SCCS: contains sccs history files which are used for source code control and configuration management during development and maintenance.

The 'common' directory also contains a 'scripts' and an 'icons' directory. The scripts directory contains UNIX csh scripts, which are used by ISEIS and general purpose utility scripts. The 'icons' directory contains all icons which are used by the ISEIS system.

Data files (waveforms, spectra, beams, etc.) are stored in either ./iseisdata, ./iseisdata1 or ./iseisdata2. The database contains relations which specify the locations of these files.

A complete directory overview with process and library descriptions is provided in the appendix.

7.2 ISEIS INSTALLATION

Before attempting ISEIS installation, re-read Section 7.1, since a thorough understanding of the directory hierarchy is essential.

The ISEIS system comes delivered on a nine track tape created by the UNIX utility 'tar' at 6250 bpi. It should, therefore, be restored using 'tar -xvf /dev/rmt8' using a tape drive that supports a density of 6250 bpi. After the tar has completed, there should be an iseis subdirectory created in the current directory. This directory should contain all of the ISEIS configuration files, such as the .cshrc, .login, .Env* .StartIseis, .rhosts, .awmrc, .twmrc, .Xdefaults, .Xdefaults-*, .xdesktop, .MA*, .XI*, and .xi* files. The 'iseis' subdirectory should also contain the 'common' subdirectory. Installation consists of the following steps:

1. Log in to iseis account and cd to directory where iseis is to reside.
2. tar -xvf /dev/rmt8
3. cd /iseis/tools/mimic
4. make
5. cd ./iseis; mkdir tmp; mkdir sparc; mkdir sun; mkdir stardent; mkdir bin; mkdir lib
6. rlogin [sun4_host_name][sun3_host_name][stardent_host_name]
7. cd sparc (Make sure that you are on a SUN4 system.)
8. ../tools/mimic/mimic ../common
9. Setup environment variables in ./iseis/.EnvEnsco or .EnvCenter. See Section 7.3.1 for information.
10. make (This will take a while.)
11. Verify that all ISEIS processes were built Check one of the MachAlloc files (./iseis/.MA*) to see which processes should be built in ./iseis/sparc/bin. Process names are listed in the far left column under EXECUTABLES.
12. Repeat steps 5 through 11 using stardent (for the Stardent system) and Sun (if there is a Sun3 system in the network). In each case, it is imperative that you are logged on to the proper system.

13. Use tar to restore any ISEIS data files and database export files. This would include the ./iseis/data subdirectory.

14. Log in to database and import the data base tables.

15. Since the data files are probably in a different place than they were when the database was exported, all *DISC relation fields which make references to path names need to be updated so that ISEIS processes can find them. This can be tedious and there should probably be scripts to 16. ISEIS is ready to go!

However, Step #9 is rather complicated and Section 7.3 should be thoroughly understood before proceeding. ISEIS can be started by logging onto any ISEIS member system and answering 'Y' to the Start ISEIS prompt.

7.3 CONFIGURING ISEIS

The ISEIS system is designed to be adaptable to a variety of network and X window display configurations.

7.3.1 Setting Up the Environment

Two .Env* files are supplied with the system. They are:

- EnvCenter - Used for ISEIS configuration at the Center for Seismic Studies;
- EnvEnsco - Used for ISEIS configuration at ENSCO, Inc.

Chances are that these files cannot be used as is, but they are a good place to start. The .EnvCenter script will probably be most useful.

Configuration of the environment is explained in the following paragraphs. Variables with an '*' after them are most important and should be changed if they do not suit the current system configuration. Non-'*' variables can probably be left as is with no ill effects. Variables which define path names, library names and network configurations should be most closely scrutinized since they are most likely to change. More stars are used to emphasize importance and greater likelihood of change.

The following environment variables are only used when ISEIS is running:

HMD*:** This is set to the path name of the ISEIS root directory. In some cases, this directory may not be the same as the HOME variable which points to the default login path. The configuration files should always reside in the HOME directory and ISEIS should always be in the HMD directory. In the .EnvCenter example, HMD is set to /data/honer/iseis.

ISEIS_HOSTNAMES*:** This UNIX environment variable defines which systems are part of the ISEIS network. In .EnvCenter this variable is set to (trym dvalin honer). This means that one ISEIS System Monitor will run on each of these systems and that each of these systems will be considered as a possible host for process execution. These names are found by typing 'host name' on each system.

ISEIS_SUN_HOSTS*:** This variable defines which of the above systems are Sun3 or Sun4 workstations. Since most processes require use of the database, ISEIS must know which systems are Suns and which are Stardents which do not support ORACLE or SQLNET. The Stardent also runs UNIX system V and the Suns run the Berkeley variety of UNIX. These operating systems are very similar, but there are sufficient differences which necessitate this classification. In this particular example, there was no Stardent at the Center so the set of ISEIS_SUN_HOSTS was the same as that of ISEIS_HOSTNAMES. However, if 'trym' was a Stardent computer, then this variable would be set to (dvalin honer).

ISEIS_BIN_DIRS*:** This variable defines the path names for ISEIS executable processes for each system that is part of ISEIS. There should be a one-to-one correspondence between a bin path and a system listed in ISEIS_HOSTNAMES. In the .EnvCenter example, all of the ISEIS_HOSTNAMES are Sun4 type systems so they each reference the same path name. However, if 'dvalin' was a Sun3 instead of a Sun4, then the variable would be set as: ($HMD/sparc/bin $HMD/sun/bin $HMD/sparc/bin). Please note that ($HMD/sun/bin $HMD/sparc/bin $HMD/sparc/bin) would be improper since ISEIS would then assume that the executables for trym are located in $HMD/sun/bin. By convention, Sun4 executables are located in ./iseis/sparc/bin, Sun3's in ./iseis/sun/bin and Stardent's in ./iseis/stardent/bin. However, these could be placed anywhere.

PSFLAGS*:** This variable should not really be necessary. However, because UNIX System V uses different flags for the 'ps' command than Berkeley UNIX, flags for the 'ps' command have to be specified for every system in ISEIS HOSTNAMES. All Sun4 and Sun3 type systems should use '-ax' and Stardent (System V) systems should use '-ef'. In the example, all PSFLAGS are specified as '-ax' since there are no Stardent systems. However, if honer was a Stardent system, then PSFLAGS would be specified as (-ax -ax -ef). Again, the order of specification must agree with ISEIS_HOSTNAMES. (-ef -ax -ax) would be improper since ISEIS would assume that trym was a System V type system. Ultimately, these flags are used by the Monitors to determine cpu load results.

ISEIS_LOAD_LIMITS:** This variable is used for selecting systems for execution. All processes that wish to start other processes first query the monitor for cpu load statistics. The loads received from all of the system monitors are compared with the load limits specified in this environmental variable. A system is too busy if the cpu load obtained for it exceeds the corresponding value in the environment. If these values are set too low (0.0 to 0.5), then nothing may execute. Higher numbers (2.0 to 5.0) will allow more processes to execute. These values can be set to very high values to be sure that processes will always run. This effectively overrides the system load criteria for ISEIS and interactive performance may degrade.

ISIS_WANTED*: This should be set to 'TRUE' if communication between Dispatcher and ISIS processes is desired.

ISIS_HOME*: Path name for the ISIS agent program.

ISISPORT:** The ISIS agent port number. 2001 is used in the example.

AESIR_HOST*:** System where Dispatcher is to run. This must be a host in ISEIS HOSTNAMES.

DISP_HOME*:** This is where the Dispatcher executable is located.

DPFLAGS* UNIX 'ps' flags:** These are used during startup to check if the Dispatcher is running.

ISEIS_MAX_PROCESSES:** This is used by the Top Level Spreadsheet process to regulate how many discriminant processes may run simultaneously. The Spreadsheet continues to start discriminant processes until the number of active processes is equal to this number (see Section 2.1). This number should vary from (2 - 10). If the number is too high, then some discriminants could terminate abnormally due to ORACLE access denials. ORACLE only allows a finite number of processes to be logged on at a time. If it is set too low, then performance may degrade. This variable is only used for discriminant processes, since there is a potential for very large numbers of discriminant processes to run simultaneously.

ORACLE_HOME*:** Used by ORACLE and the ORACLE embedded SQL precompiler PCC. This is the home directory for the ORACLE software.

UI*:** ORACLE user id.

RHOST*: This could be any host name which is part of the ISEIS system which has SCCS capability. The Stardent machine currently has no sccs software. This is used when the system is compiled and modified.

ISEIS_LOAD_CHECK_INTERVAL*: This UNIX environmental variable determines how often the system Monitor processes query the system for cpu system loads. Too low of a value will result in degradation of Monitor performance since the Monitors will consume an excessive amount of cpu time to create the load files. The process is moderately time consuming since the Monitor process must use the UNIX system() interface to start the UNIX 'uptime' command and redirect its output to a file. This file must then be accessed. If the value is set too high, then system load requests may return values which are misleading. Values between three and 10 are reasonable.

ISEIS_TOPO_FILE*:** This variable is only used by the Interactive Map process to determine the location of the topographic database file which is used when the Map displays the cross-section.

ISEIS_ELEV_NAME*: Name of the topographic database currently in use. This name appears on the Interactive Map's cross-section display.

ISEIS_AUTO*:** ISEIS automatic processing is allowed if this environment variable is set to TRUE.

ISEIS_DEBUG: This is used for debugging only. It should always be set to '0' for normal use.

ISEIS_AUTO_TIMEOUT*: This is the time in seconds that an automatic discrimination process should wait for messages from its subordinate processes. This value is used in d_listen() Dispatcher function calls.

ISEIS_AUTO_SLEEP*: ISEIS automatic process sleep time.

ISEIS_CONFIG_PATH*:** Same as ISEIS_CONFIG_DIR.

ISEIS_DATA_DIR*:** Directory where any newly created data files should go. This path name ultimately finds its way into the *DISC relations in the database.

PROC_LIST*: This variable specifies a list of ISEIS processes which should be started initially when the ISEIS system is booted. At least one of [dsmain,map] should be specified and the RGBcolor process should be specified.

PROC_ARGS*: This variable specifies a list of default process command line arguments which should be used when the processes in PROC_LIST are started. They are specified as:

("arg1_for_proc1 arg2_for_proc1 .. argn_for_proc1","argn_for_procN")

where 'n' equals the argument number and 'N' is the process number. For example: If PROC_LIST is equal to (dsmain RGBcolor) then PROC_ARGS could be specified as:

("$W/eventFile.d $UI" "0 0 0 . -USE_PNAME").

Note that it would be improper to specify the variable as:

("0 0 0 . -USE_PNAME" "$W/eventFile.d $UI")

since ISEIS would assume that dsmain is started as:

<dsmain 0 0 0 . -USE_PNAME>

which is wrong and the program will not start properly.

ISEIS_HAS_DATA_BASE*: This variable should always be set to TRUE. The only time it is set to FALSE is when ORACLE is out of commission for some reason and a developer wants to test some other function in the Top Level Spreadsheet Display which is unrelated to the database.

ISEIS_WORKSPACE_DIR*, ISEIS_TEMP_DIR***, W***:** These variables should be set to a directory name (all the same) where temporary scratch files are to be placed.

IEID_RESULTS_FNAME*:** This variable specifies the root directory for explanation and dribble files. As explained in Section 2.1.5.1, this directory contains subdirectories for each discriminant process which contain text files that are used in the creation of the Top Level Spreadsheet summary displays.

The following variables are only used when the ISEIS system is being compiled and linked during installation and development

UNFORMATTED_IO*:** This variable is always set to BSD to force all Fortran code in ISEIS to interpret file record length arguments in bytes rather than words. This allows FORTRAN files compiled under UNIX System V and UNIX 4.3 to read and write binary files the same way. See f77 man pages.

C5LIBS*, LC***:** Specifies the System V UNIX C library for linking. In the example, it is set to /usr/5lib/libc.a. This variable is used in some Makefiles to insure that UNIX System V libraries are used instead of Berkeley UNIX libraries. This is absolutely necessary in some applications which use both FORTRAN, C and SQL libraries. If not specified, the applications will link with Berkeley C libraries and the applications will abort when run due to a bug involving the Berkeley C libraries and SQL.

PCCINC*:** Location of ORACLE include files.

OL*:** Location of ORACLE libraries

SQLLIBS*:** This UNIX environment variable should consist of a list of libraries which are needed for creating applications using embedded SQL to access and change the database. It is used in Makefiles.

FSQLLIBS*:** This variable is referenced in Makefiles by applications which access ORACLE using FORTRAN interfaces.

AR: The name of the UNIX archive librarian utility. Usually ar.

CCO: The name of the UNIX system V C compiler. Usually /usr/5bin/cc.

CC: The name of the UNIX default C compiler. Usually cc.

MV: The name of the UNIX file move utility. Usually mv.

PCC: The name of the ORACLE embedded SQL pre-compiler. Usually pcc.

YACC: The name of the UNIX Yet Another Compiler Compiler. This is another precompiler which is used for creating modules which use the finite deterministic automaton algorithm. It is usually yacc.

RM: UNIX file remove command. Usually rm.

PRINT: UNIX command to print files. Usually 'lpr -p.'

SCLEAN: UNIX command to remove files which can be re-created with the sccs utility and a compiler/linker. Usually 'sccs clean.'

RANLIB: UNIX utility to prepare a library for random access. 'ranlib' usually used.

GET: UNIX utility used to retrieve files from sccs. 'sccs get' usually used.

CSSLIB: Library name for routines used for flat file I/O, information retrieval and seismic numerical processing algorithms.

INSTALL: UNIX command used for installing programs into the default system executable directories (ISEIS_BIN_DIRS). 'install -m 4755' usually used.

DORE_LIB:** This variable only pertains to applications which use the Stardent DORE graphics package. It is set to: '/usr/lib/dore.o -lXd -lXtitan -lXB' by default.

XINCLUDE1*, XINCLUDE2***:** These environmental variables specify where X window and Athena toolkit include files reside. XINCLUDE2 is used if XINCLUDE1 is not sufficient. /usr/include/X11 is usually used.

XLIBS*:** These are the X window library file specifications. '$XL/libXaw.a $XL/libXt.a $XL/libXmu.a $XL/libX11.a' is usually used where XL may be set to /usr/lib or /usr/lib/X11.

LDFLAGS: This variable specifies the flags that are to be used when processes are linked into executable modules. '-g -43' is usually used.

FFLAGS: This variable specifies the FORTRAN compilation flags. setenv FFLAGS '-g -cpp -43 -D_STARDENT' is currently used on the Stardent and setenv FFLAGS "-g -D_SUN" is used on Sun type systems.

CFLAGS: This variable specifies the C compilation flags. setenv CFLAGS '-g -43 -O0 -D_STARDENT -D_X11R2' is currently used on the Stardent and '-g -D_SUN -D_X11R4' is used on Sun's.

FORLIBS*: This variable specifies the list of FORTRAN libraries. The variable is not used on the Stardent and it is set to '/usr/lib/libF77.a /usr/lib/libU77.a /usr/lib/libI77.a' on Sun type systems. A /usr/lang/SC0.0/libF77.a specification may have to be used depending upon site.

CONTRIB*:** This variable specifies the path name to the NMRD library containing the LYNNES calcomp X window plotting utilities.

CONTRIB_INCLUDES*:** Location of LYNESS include files.

CONTRIB_LIBS*:** Specification of all libraries needed for LYNESS widgets. Library names may be different depending upon installation site.

F77: Name of FORTRAN compiler. Set to 'fc' on Stardent and 'f77' on Suns.

MATHLIB: C math library specification. Usually /usr/lib/libm.a.

COM_LIBS: Library specification for libraries needed for Dispatcher IPC interface. $HMD/{sparc,sun,stardent}/lib/libiseis.a usually used.

COM_SRCS: List of all modules needed for Dispatcher/ISIS communications package interfaces. This is set up in this fashion to allow Dispatcher or ISIS only modes of operation.

PCCFLAGS: Defines ORACLE pcc flag list which is used when *.pc files are pre-processed by the pcc pre-processor embedded SQL code.

ENVUSGSTAB: This is an environmental variable used by routines in the ibseis library which specifies a path to a *.binary file containing station location data.

There are other environmental variables in the .Env file, but these are derived from the above variables and should not need modification. Once the Environmental parameters are defined, the administrator should do a 'source .Env[Center][Ensco]' to insure that no csh syntax errors have been made. A 'printenv' may also be a useful way to insure that the environment was setup correctly. The operator should also verify that a 'source Env*' command is in the .cshrc file. If any errors occur during compilation or linking, then the environment is probably suspect. Common problems involve incorrectly specified include file directories, incorrect library names and incorrect CFLAGS or FFLAGS specifications.

Common run-time errors involve non-execution of discriminants due to missing executable software (check ISEIS_BIN_DIRS and MachAlloc file). Discriminant processes may also not run due to an unavailable database. Try to log on to SQL interactively to verify availability. Also, make sure that all path name type environmental variables are set correctly. In particular, ISEIS_BIN_DIRS, ISEIS_WORKSPACE_DIR, IEID_RESULTS_FNAME, HMD, ISEIS_CONFIG_DIR, ISEIS_DATA_DIR must be correctly specified. Discriminant processes may run for a long time before finishing. Therefore, the administrator may want to try UNIX 'ps -ax | grep process_name' commands on each of the member systems to check for the suspect processes before panicking. Also, verify that the Dispatcher and system Monitors are running. These may have gone away, or they could be in strange disk wait states (D). A UNIX system with D waiting process should be rebooted and ISEIS restarted.

7.3.2 Setting Up the MachAlloc File

Whereas the .Env* file defines system configuration and the compilation/ link environment, the MachAlloc file defines process specific information such as X window display, executable and alias file names, and Top Level Spreadsheet entry names for discriminant processes. Since the MachAlloc file contains documentation within itself, the comment section and some sample entries are included below:

```
# SCCSID: %W% %G%
# This file is used by DESCRIMINATE and MONITOR for process control
#
# Column definitions #
#  EXECUTABLE     - The actual name of the process in the
#               $HOME/[sun,sparc,stardent]/bin dir
#               In the case of discriminate processes, it is
#               the root process name
#               (i.e. the actual process name without
#                [.EXE, .DTA, .UPD, .MDL, .CSE] extensions)
#
#  ALIAS_NAME     - Used to derive explanations and kbases directories
#               If a '*' is in this column, then EXECUTABLE is used
#               to derive directory names.  This allows discriminants to
#               be "cloned" easily by aliasing different rules to same
#               program.
#
#  TECHID         - Technique Id associated with process - should be
      UNIQUE
#
#  PREFERRED      - Host that we would rather run on if given a choice
#               If there is no preference, then use 'Any' or Sun
#
#  REQUIRED       - Host that this process must run on
#               this host overrides any preferred hosts unless
#                it is specified as 'Any' or Sun.
#
#  COLUMN_NAME    - a non '*' name is used here to define column names
#               for discriminates in top level display
#               A '*' indicates that an entry should not be placed
#               into the top level display.
#
#  X_DISPLAY      - This column provides a capability to override
#             default .cshrc defined DISPLAY variables for
#             certain processes. Thus, an X process will always
#             display on the given X_DISPLAY regardless of which
#             machine it is running on.
#
# EXECUTABLE ALIAS_NAME TECHID PREFERRED REQUIRED COLUMN_NAME X_DISPLAY
    map          *        0     mapsys    mapsys      *        mapdis:0
   dsmain        *        1     topsys    topsys      *        topdis:0
  RGBcolor       *       12     grsys     grsys       *        grdis:0
```

Generic host names and X window displays have been used here for illustration, and the three essential ISEIS processes have been included. This defines the minimum allowable MachAlloc file. However, ISEIS won't be very useful without its other processes, and the .MA* files in the home directory should be used as templates to customize your installation.

Basically, the Interactive Map process (map) needs to run on a system which has access to the database, and it cannot display on a Stardent type X window display. The Top Level Spreadsheet Display (dsmain) can run on any system which has database support, and it can display on any system in the ISEIS network. The RGBcolor process must run on a Stardent system, and it must use the Stardent X window display. The Map and Spreadsheet must be configured to display on different displays since each process allocates a large number of color resources. However, they both can run on the same machine, if desired. In the above example, mapsys and topsys are subsets of ISEIS_SUN_HOSTS and grsys is a Stardent type system. 'mapdis' and 'topdis' are Sun type X window displays and 'grdis' is a Stardent X window display. The Stardent X window display is different from Sun displays in that it can display 24 bit or 'TRUE' colors. It does not need to use a color map like the Sun type displays. The Map cannot display on the Stardent due to the Stardent's poor implementation of X window color allocation standards.

All Display processes except the Sono3D process should be configured to display on 'topdis.' Sono3D is a high performance 3D dynamic surface viewing application which must display on Stardent type systems. The supplied .MA* files can be copied to $ISEIS_CONFIG_DIR/MachAlloc and modified. The EXECUTABLE, ALIAS_NAME and

TECHID fields should not be changed. 'SUN' and 'Any' items in the PREFERRED and REQUIRED fields should not be modified. All occurrences of a given system name should be changed to a system which is a member of the current ISEIS_HOSTNAMES, paying careful attention that non-database equipped systems are used for most processes. The Stardent system host name should definitely not be used for most processes. It should only be used for 'RGBcolor' and 'Sono3D.' 'Any' designations could run on the Stardent, but they have been specifically designed to not use the database. These are usually numerically expensive processes.

As a check for the MachAlloc's correctness, one may want to obtain directory listings of the executable directories and check them against the entries in the MachAlloc. There should be at least one file that matches a UNIX query of 'ls ./iseis/[sparc,sun,stardent]/bin/*EXECUTABLE*' for each entry in the MachAlloc. If the REQUIRED field for the entry is set to a specific host name, then the executable file should be present in the bin directory for that system. If it is set to SUN, then there should be executable versions available on all SUN systems. If it is set to 'Any', then there should be executable versions available in every bin directory listed in ISEIS_BIN_DIRS. If executable files are not found where expected, then the MachAlloc should be modified to reflect the true configuration. A correct MachAlloc must also have unique TECHIDs for each process. In addition, there should be a subdirectory name for each ALIAS_NAME created in $IEID_RESULTS_FNAME. Finally, there should be an ALIAS_NAME.P file in $ISEIS_CONFIG_DIR/sumprocs for every non '*' ALIAS_NAME in MachAlloc. If all of this information is correct, then the Top Level Spreadsheet Display process should be able to execute discriminant processes and display results summaries (2.1.5.1) successfully. The MachAlloc is primarily used by the Spreadsheet, although any process which starts another process via prExecute*() uses the MachAlloc as well.

APPENDIX A - GLOSSARY

This appendix contains definitions of some of the terms used in this manual which may not be obvious to the reader. Terms which are listed in UPPERCASE first appeared in the text in that form, although they may also appear in lower or mixed case.

AMPRATIO	This process computes the amplitude ratio between P phases and S phases.
ARS	The Analyst Review Station of the IMS system. Operator interface with ISEIS is initiated from this process.
CBR	Suffix allied to explanation files to identify files from the Case Based Reasoning process.
CEPPKS	Cepstral peaks data base relation.
CFKDISC	Continuous F-k database relation.
CPDISC	Cepstra database relation, describes cepstral beams.
CSE	Suffix applied to results of Case-Based Processing.
DORE	Graphics package supplied with Stardent computer.
DTA	Suffix applied to Data Assessment Results.
DTW	Dynamic Time Warp, Process used to match incoherent beam envelopes with reference patterns.
DTWDisplay	Process to display the results of DTW processing.
EXE	Suffix applied to the process which applies model-based and case-based signal processing and rules for discrimination.
FFT	Fast Fourier Transform.
FK	Frequency/wavenumber processing.
FSDISC	Database relation which describes spectral result files.
GC	Great Circle, mnemonic used in map menus.
Gouraud	A specific type of continuous shading of three-dimensional plots.

Hanning	A particular type of weighting applied when generating spectra.
IAS	Intelligent Array System.
IBeam	Incoherent beam process.
IMAKE	Independent make, a method whereby UNIX makefiles can be automatically generated from templates and site specific configuration files.
IMS	Intelligent Monitoring System.
IMakefile	The configuration files used by IMAKE.
IPC	Inter-process communications.
ISEIS	Intelligent Seismic Identification System.
MBR	Suffix applied to the explanation files for model-based reasoning.
MDL	Suffix applied to the results of model-based processing.
MERSTAT	Database relation which receives the results of ripple fire processing.
MERSY	The actual process name for the ripple-fire process.
MERSYDPY	The operator interface for the MERSY process.
MachAlloc	A table used by system load monitor to determine where process and displays should be assigned.
Makefile	Unix shell file used to generate executable images, documentation, etc.
NFS	Network File System, a driver which allows multiple computer to share files.
NMRD	Nuclear Monitoring Research and Development, the name for the overall system of which ISEIS is the classification process.
ORID	Origin Id, commonly used to identify specific events for processing.
PSratio	P phase, S phase ratio, a particular type of amplitude ratio.
RANLIB	A UNIX command for generating a randomly accessed object library.

RATFOR	Rational FORTRAN, a preprocessor which provides a structured element to the FORTRAN language.
RESPDISC	Database relation which describes recipes for signal processing algorithms.
RGB	Red, Green, Blue, a method for specifying colors.
RIPPLEFIRE	A process for detecting and classifying multiple explosion events.
RPC	Remote Procedure Call, a method for initiating processes on other computers on a network.
RULEDISC	The database relation which contains information about the rule files used by the discrimination processing.
SCCS	Source Code Control System.
SNR	Signal-to-Noise ratio.
SPECRATIO	The database relation which contains the results of the Spectral Ratio processing.
SQL	Structured Query Language, the language used to access the database.
SQLNET	A program which allows a process to access database information on a remote machine.
STADISC	The relation which describes the Short-Term Average files which are generated by the DTW process.
STAPICK	The relation which receives the phase picks from the Short-Term Average.
STR	Suffix applied to the results of the Data Assessment processing.
Stardent	Multiple processor computer developed by Stardent Computer Company.
Top Level Spreadsheet	The primary user interface for ISEIS.
TECHID	Technique Id, identifies the discrimination technique used in a particular set of processing.
UPD	Suffix applied to executable process which applies only rule-based processing to an event.
WFDISC	Database relation which describes particular waveform files.

X	User interface system developed by MIT, used to develop ISEIS operator interface.
Xdefaults	A defaults file which contains user preferences for the X system.
autoDS	The ISEIS automatic discrimination program.
awm	Ardent window manager, an X process which permits the user to manipulate various process windows.
awmrc	Awm resource file, contains defaults for awm.
cbid	Suffix applies to the process which applies case-based processing rules to an event.
cepstrum	The Fourier transform of a spectrum, used to identify multiple explosion events.
csh	C shell, the standard UNIX interface.
cshrc	Csh resource files, contains defaults for csh.
discriminant	A set of processes which attempts to classify an event by applying specific signal processing and rule-based analysis.
disctech	The database relation which contains information about particular discriminants.
dsmain	The name of the top level spreadsheet process.
dtwstat	The database relation which receives the statistics generated by the DTW processing.
f77	The UNIX FORTRAN compiler. The Stardent version is named FC.
login	A csh file which is executed upon login to perform one time only environment operations.
lpr	Command used to print files from UNIX file system.
mbid	The suffix applied to the program which performs model-based rule discrimination processing.
menubar	A window at the top of most ISEIS processes which contain menus for controlling the process.
pulldown	A pulldown menu, that is a menu which appears when an item in the menubar is selected.

quefrency	The independent variable used to display cepstrum results. Analogous to frequency in spectra displays.
rsh	Remote shell, a means for executing user commands on a remote machine.
sonogram	A form of three-dimensional spectra produced by the Continuous Fourier Spectra processing.
toolkit	A library of widgets for implementing various aspects of the X user interface.
twm	Tom's window manager, an X process used to manipulate windows in the X interface, similar to awm described above.
twmrc	Twm resource file, contains defaults for the twm program.
widget	A coined name for pseudo object oriented elements of the X system which are used to implement user interfaces. Examples are pushbuttons, scrollbars, etc.
xterm	An X program which implements a standard terminal type interface in an X window.
yacc	Yet another compiler compiler, a program which implement a particular lexicon for specifying program interfaces.

APPENDIX B - ISEIS ENVIRONMENT

B.1.0 ENVIRONMENTAL VARIABLES

See discussion in Section 7.0 and .Env* files in $HOME.

B.2.0 SYSTEM FILES

***** $ISEIS_WORKSPACE_DIR/eventFile.d

This file is created during 'OPEN DB' functions within the Interactive Map and Top Level Spreadsheet processes and is accessed by the same processes. Information from the ORACLE database is retrieved, formatted and written to file. This file is used for loading and changing events within these two tasks. The File consists of the number of events followed by ASCII records consisting of:

Origin_Id
Event_Latitude
Event_Longitude
Event_depth
Event_time(epoch)
Year
Month
Day
Hour
Minute
Second
Mb_statistic
Ms_statistic
Local_Magnitude
Semi_Major_Axis
Semi_Minor_Axis
Angle_Of_Semi_Major_Axis

Confidence_Ellipse_error
Event_class
Reference_Event_Flag
Number_of_stations
Station_Name
Station_Latitude
Station_Longitude
Elevation
Station_Type

There should always be 'Number_of_stations' sets of information following the 'Number_of_stations' field.

******* $ISEIS_CONFIG_DIR/MachAlloc**

This File is used by all processes in ISEIS that start processes via the Unified Process Initiation Software (see Section 6.2). The file provides system-wide data which is used when processes are started. Data from this file is used for assigning systems, locating explanation files for the Top Level Spreadsheet summary displays and X Window display determination. In addition, the Spreadsheet uses the COLUMN_NAME field to create discriminant names in the display. The file is discussed in detail in Section 7.3.2 so only the format is given here. The file may have comments anywhere and these are formed by placing a '#' in column 1. Valid entries consist of the following:

EXECUTABLE - Name of executable program. This may be the actual program name or the root name of a discriminant. For example, an entry for the PSratio discriminant would be specified as 'PSratio' which is used to form the executables PSratio.DTA, PSratio.EXE, PSratio.UPD, PSratio.mbid.

ALIAS NAME - An alias name for the executable image. This field is used when discriminants are 'cloned,' a process in which a different set of rules is used on the same set of data. This alias name is also used to derive Top Level Spreadsheet explanation and dribble file names, as well as Optional Summary process file names (see Section 2.1.5.1). An '*' in this field results in ALIAS_NAME and EXECUTABLE equivalence.

TECHID - Unique process identifier.

PREFERRED - One of host name, Any or SUN, where host name is one of the systems specified in ISEIS_HOSTNAMES. This field indicates the preferred system or system group for the process. Processes will be run on host name or the least busy system in ISEIS_SUN_HOSTS (SUN) or the least busy system of ISEIS_HOSTNAMES (Any) if it is a subset of REQUIRED and the system load is low enough.

REQUIRED - One of host name, Any or SUN, where host name is one of the systems specified in ISEIS_HOSTNAMES. This field indicates the processes system requirements. This specification always overrides PREFERRED if it is more limiting. For example, if REQUIRED is specified as host name and PREFERRED is specified as SUN, then only host name is considered. However, if REQUIRED is specified as SUN and PREFERRED is specified as host name, then the process will be executed on host name if the system load for host name is less than that specified in ISEIS_LOAD_LIMITS. Otherwise, the least busy system in SUN is used (if load limits for it are low enough).

COLUMN_NAME - A non '*' field indicates that this is a discriminant entry and that COLUMN_NAME should be used as the name of the discriminant in the Spreadsheet display. The '_' characters are translated to ' ' and '\' characters are translated to carriage returns in the Spreadsheet.

X_DISPLAY - X window display to use for the process's display. An '*' causes the default X window display to be used. In other words, one of the values specified in ISEIS_X_DISPLAYS.

******* $ISEIS_CONFIG_DIR/MAPS/MapDefaults.**

This file is used by the Interactive Map Process for defaults. A sample file is given below:

```
map_path    /iseis/data/MAPS/
map    mer_world.info
Base_Map_Colors  colors/base.cms
overlay_path    /iseis/data/overlays/
overlay_colors   colors/ovly.cms
ArcessSta    ARCESS.sta
FinesaSta    FINESA.sta
GeressSta    GERESS.sta
NoressSta    NORESS.sta
CdsnSta     CDSN.sta
GsettSta    GSETT.sta
IrisSta     IRIS.sta
NorsarSta    NORSAR.sta
DssSta      DSS.sta
```

The map_path entry indicates where the HOME directory is for the Map files. The map field indicates which map to load by default. In this case, the Map will load the world map by default. Base_Map_Colors specifies elevation colors to use for maps. Overlay_path specifies the absolute path name to the overlay files. Overlay_colors specifies the colors to use for overlays. The sta entries specify filenames in the overlay_path directory which contain station lat/lon information for station overlay plotting.

******* $ISEIS_CONFIG_DIR/MAPS/MapInfo**

This file specifies map specific data. The first value indicates how many map files are currently configured in the system. Each entry consists of:

1. The Name for the map which appears in the 'New Map' option in the FILE menu in the Map process.
2. The name of the information file which resides in 'map_path.'
3. The projection type (1 = Mercator, 2 = Azimuthal Equidistant).
4. The Longitude min/max and Latitude min/max.

An example entry is given below:

```
Africa     aze_africa.info     2 -40.0  -44.0  45.0  84.0
```

Every map has an *.info, *.elev, *.pde and a *.pb file. The *.info file contains descriptive text about the map. The *.elev file contains elevation data. The *.pde file has data pertaining to preliminary determination of epicenters. The *.pb file is the actual X window image data for the map. For the above example, there is an:

```
aze_africa.info     aze_africa.pde     aze_africa.elev     aze_africa.pb
```

B.3.0 ISEIS UNIX SCRIPTS

UNIX csh scripts for ISEIS either reside in $HMD/common/scripts or $HOME, and they are responsible for setting up the UNIX environment also used in process execution. It is important that each of the following scripts is executable. If they are not, the system administrator should perform a 'chmod a+x file_spec' on them:

******* $HOME/.EnvEnsco**

Sets up environment for ISEIS processing at the ENSCO, Inc., facility.

******* $HOME/.EnvCenter**

Sets up environment for ISEIS processing at the Center for Seismic Studies. The contents of these files are discussed in Part 1 of this appendix.

******* .cshrc**

This standard UNIX script is executed every time a csh shell is invoked and its main contents is a 'source .Env*' where '*' is either 'ENSCO' or 'Center.'

***** .login

This standard UNIX script is executed every time a user logs on and asks the user if he/she wishes to start ISEIS. A 'Y' answer causes the .StartIseis script to be executed.

***** .StartIseis

The operation of this script is described in Section 2.2. This script is responsible for Initializing and starting the ISEIS system.

***** .XI[hostname] .xi[hostname]

A pair of these files should exist for every host which is defined in ISEIS_HOSTNAMES. These scripts are briefly described in Section 2.2.

Each .XI* file first determines if X windows is active on its system and it executes 'xinit' (or xstart for Stardent systems) to start X windows if there is no X server currently running. For example, .XIhoner is responsible for determining X window activity on the honer system and invoking 'xinit' if it is not. Each .XI* file specifies a corresponding .xi* file as an argument to 'xinit.' However, Stardent type systems do not use this convention since the Stardent window manager (awm) always executes the .xdesktop upon startup. Each .xi* file functions like a .xdesktop specifying processes to start once the X server has started. The .xi* files usually just start a console window and the twm window manager.

***** .StartProc

The .StartProc script is located in $HMD/common/scripts and it is described in Section 2.2. It is a general purpose process execution script which insures that only one process of the given type is run. Its arguments are:

host name - system to execute the process on/
process - name of process to start/
xdisplay - xdisplay to use for process/

psfflags - flags for 'ps' command arguments -/
command line arguments to use when starting process

If the script determines that the process is not active, it sends a START_PROC message to the Monitor on 'host name' using the given xdisplay and arguments. Otherwise, it sends a NEW_ARGS message to the already running process. It uses the ISEIS utility 'send_args' to handle the Dispatcher message sends from the script. 'send_args' is invoked by:

send_args *process_addr message_id arguments* where

process_addr is |hostname|Monitor.Any or dsmain.Any, map.Any or RGBcolor.Any, *message_id* is a character message id string and *arguments* are the arguments for the process. For example:

```
.StartProc mickey dsmain walt:0 -ax $W/eventFile.d $UI
```

would start the Top Level Spreadsheet (dsmain) on mickey using walt:0 as an X window display and $W/eventFile.d $UI as arguments to dsmain.

B.4.0 ISEIS PARAMETER FILES

These files are used by ISEIS processes for parameters needed in computation:

******* $ISEIS_CONFIG_DIR/resp**

Files in this directory are of the form *.resp and contain instrument correction parameters which are used in signal processing algorithms to remove the instrument response prior to processing. They consist of a collection of ASCII floating point numbers.

******* $ISEIS_CONFIG_DIR/general/FILTER.DAT and FILTER1.DAT**

These are filter coefficient files.

******* $ISEIS_CONFIG_DIR/general/degfree.75 and degfree.99**

These are statistical degrees-of-freedom files which are used in continuous spectra calculations.

******* $ISEIS_CONFIG_DIR/general/hires.in**

This file contains continuous FK plotting parameters.

******* $ISEIS_TOPO_FILE**

This file is used by the Cross-Section display in the Interactive Map process for elevation data.

B.5.0 X RESOURCE FILES

ISEIS is configured so that the appearance and colors of its displays can be configured from the standard X window resource files. The format of these files is described in the Xlib Programming Manuals, Volumes I & II.

$HOME/.Xdefaults

This file contains resources which define defaults for the Stardent awm window manager and the xterm X window terminal emulator. It is loaded when an X server initializes by the .xi* files and .xdesktop (Stardent).

$HOME/.Xdefaults-com

Symbolic links should be setup between this file and .Xdefaults-host where host represents each system in ISEIS HOSTNAMES. If (joe mary) are in ISEIS HOSTNAMES, then there should be symbolic links for .Xdefaults-joe and .Xdefaults-mary. The .Xdefaults-[hostname] file contains general resources which apply to all processes, as well as process specific resources. These resources define where windows appear and how big they are (geometry). Foreground, background colors, fonts and border colors are also defined. Top Level Spreadsheet resources are denoted by dsmain* entries. The spreadsheet status and result colors can be modified by modifying these entries:

```
dsmain.NoDataStatusColor: Red
dsmain.SufficientDataStatusColor: LimeGreen
dsmain.IncompleteDataStatusColor: Yellow
dsmain.EarthquakeEventColor: LimeGreen
dsmain.NuclearEventColor: Red
dsmain.EconomicEventColor: Coral
dsmain.UnidentifiedEventColor: Yellow
dsmain.NoDataEventColor: Gray40
dsmain.MatchRefWellColor: LimeGreen
dsmain.MatchRefIntermediateColor: Yellow
dsmain.MatchRefPoorlyColor: Red
dsmain.ItemSelectColor: Red
dsmain.ItemDeSelectColor: Gray40
dsmain.NotAvailable: White
dsmain.Pending: Blue
```

Entries of the form *DIALOG* pertain to resources for status and and error dialog boxes. *euDbShell* entries pertain to the OPEN DB window which appears when the option is selected from the Map or Spreadsheet. *MODEL*, *DATA* and *CASE* entries pertain to the appearance of the Spreadsheet summary displays. Resources are defined for other ISEIS processes as well.

$ISEIS_CONFIG_DIR/.mapdefs

This is an X window resource file which is used exclusively for the Interactive Map Process.

$HOME/.xdesktop

This file is only used when X windows is started on Stardent type systems. It starts initial X window processes.

$HOME/.twmrc

This is a twm window manager startup file. The file contains entries to define which ISEIS processes will not have twm window titles (all ISEIS processes do not have titles). It also has entries that specify the processes which start iconified like cfsplot and cfkplot and IBeamVws.

$HOME/.awmrc

This file is only accessed by the Stardent when the awm Window manager is started on it. It contains specifications for the Applications menu which is invoked when the user presses the middle mouse button on the root window.

B.6.0 FUNCTIONAL DESCRIPTIONS AND SOURCE CODE HIERARCHY

B.6.1 GENERAL DESCRIPTION OF ISEIS DIRECTORY STRUCTURE

The ISEIS directory structure is always rooted in a directory called iseis. This path is specified in the HMD UNIX environment variable. Section 7.1 describes the hierarchy in detail. This section is basically meant to explain how software is compiled and linked to run on different systems. Every system has access to the ISEIS directory via the Network File System (NFS) and a subdirectory for each type of system should be present in the ./iseis directory. In the current configuration, sun, sparc and stardent subdirectories should be present. These correspond to Sun3, Sun4 and Stardent system types. Some ISEIS configurations may not contain all of these system types. For example, if the systems in ISEIS_HOSTNAMES are all Sun4's, then only a sparc subdirectory need be created. Conversely, it is conceivable that Macintoshes or PC's may one day be part of the ISEIS network (however these systems do not currently support NFS). The 'mimic' utility in ./iseis/tools/mimic is used to create parallel directory hierarchies for each system type in iseis. This utility replicates a duplicate directory hierarchy under the given system subdirectory. This hierarchy is identical to the common hierarchy. The mimic program also creates UNIX symbolic links for all SCCS directories. Figure B-1 shows how the symbolic links are set up. This symbolic link design insures that source code changes are tracked in only one SCCS directory. When the system is built, source code is retrieved from a given SCCS directory and copied to the given parallel directory for for compilation on the given system.

B.6.2 GENERAL PURPOSE SYSTEM LIBRARIES

The general purpose ISEIS system libraries are used by most of the ISEIS processes and the source code for them is located in:

$HMD/[MACHINE]/[LIBNAME]

where MACHINE is one of (sparc, sun, stardent) and LIBNAME is one of the library subdirectories. The subdirectory name is derived from the name of the library. For example, the library libtest.a *would have source code in libtest. See Figure* B-2 *for a graphical representation of* the main source code components. Each library is described below:

******* libDscU.a *******

Function: This library contains general purpose code for the ISEIS discriminant 'cloning' feature. This feature allows different CLIPS rules to be applied to the same data set.
Language: C with SQL.
IPC: NONE.

******* libEio *******

Function: This library contains routines which are used for accessing, reading and writing ISEIS data files which may include *.w, *.sta, spectra and filter files.
Language: FORTRAN and C.
IPC: NONE.

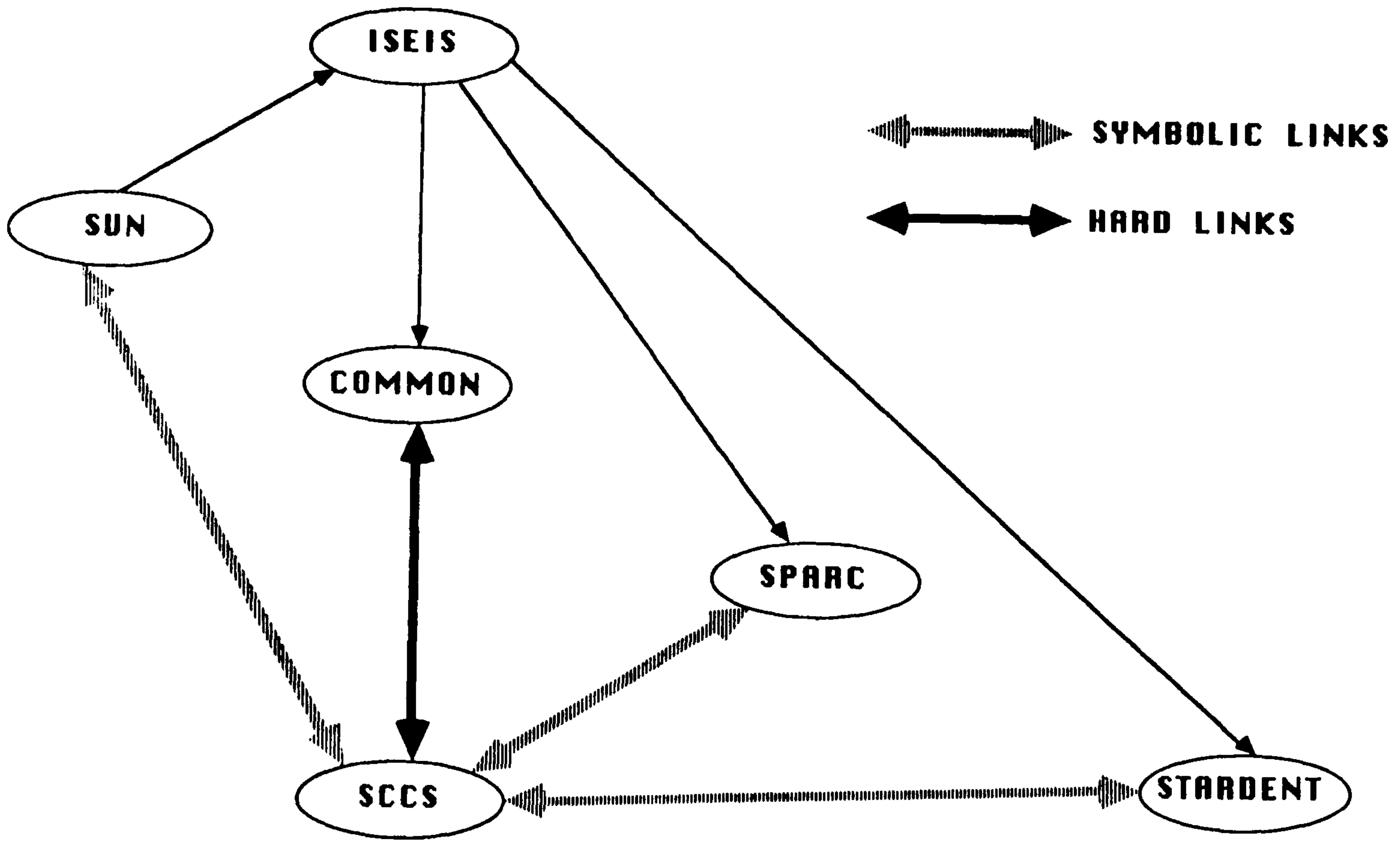

FIGURE B-1: ISEIS SYMBOLIC LINKS

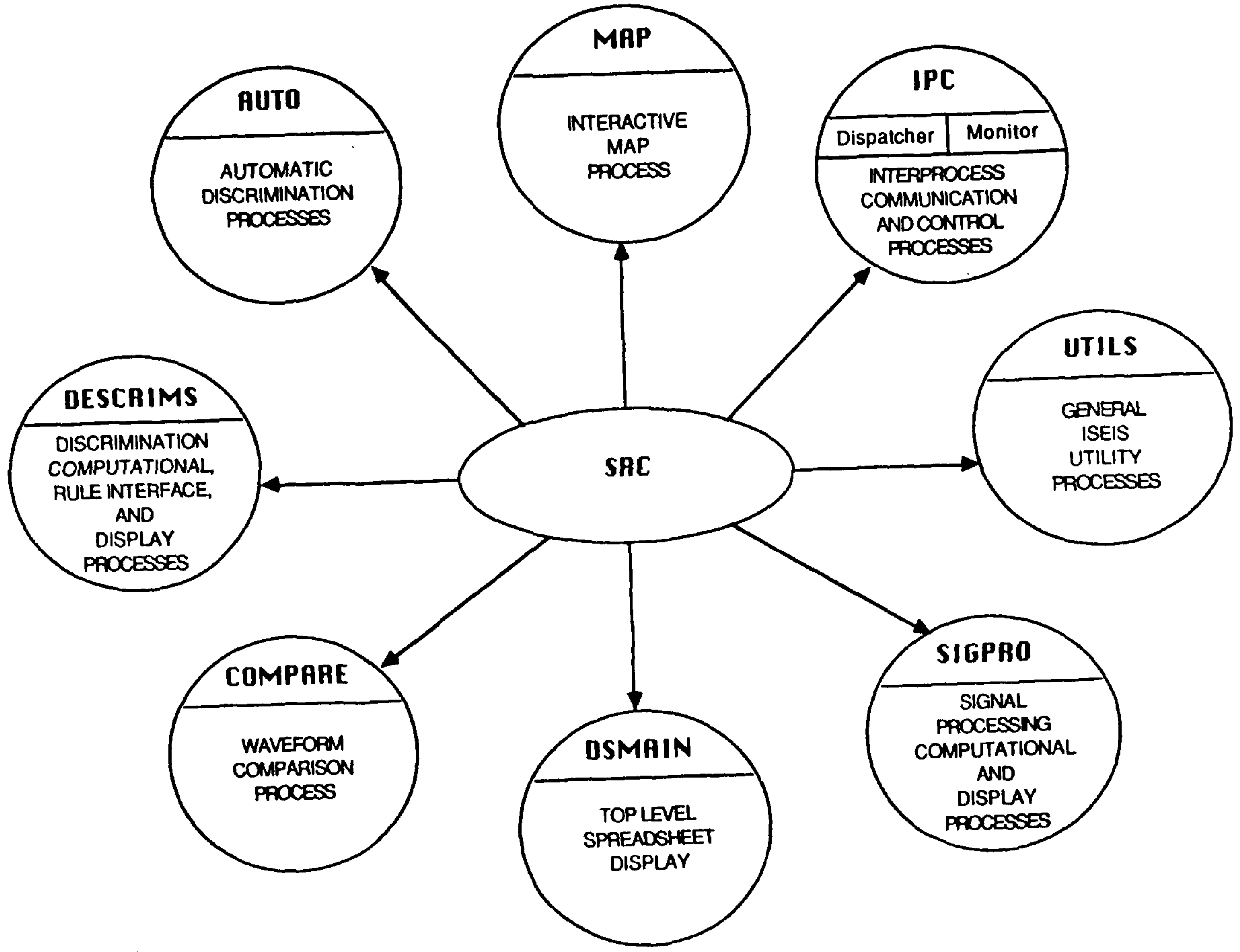
AUTO
AUTOMATIC DISCRIMINATION PROCESSES
MAP
INTERACTIVE MAP PROCESS
IPC
Dispatcher
Monitor
INTERPROCESS COMMUNICATION AND CONTROL PROCESSES
DESCRIMS
DISCRIMINATION COMPUTATIONAL, RULE INTERFACE, AND DISPLAY PROCESSES
SRC
UTILS
GENERAL ISEIS UTILITY PROCESSES
COMPARE
WAVEFORM COMPARISON PROCESS
DSMAIN
TOP LEVEL SPREADSHEET DISPLAY
SIGPRO
SIGNAL PROCESSING COMPUTATIONAL AND DISPLAY PROCESSES

FIGURE B-2: MAIN SOURCE CODE COMPONENTS

******* libEsigpro *******

Function: This library contains general purpose signal processing code which is used to compute FFTs, apply windowing functions and for digital filtering.

Language: FORTRAN and C.

IPC: NONE.

******* libPath *******

Function: Library contains an X window popup shell which is used as an interface for creating and updating the PATH relation in the database.

Language: C with SQL.

IPC: NONE.

******* libXdsp *******

Function: This library contains routines which provide general purpose Athena toolkit X window display pop-up shells which are used for obtaining station, channel, phase, filter and other selections. Utilities are also provided for viewing of time-series and frequency spectral data.

Language: C.

IPC: NONE.

******* libXe2d *******

Function: This library contains general purpose ISEIS developed Athena toolkit widgets which provide facilities for the menubar interface, XY, sequential and polar plotting, a legend widget that supports both a check box selection and menu-like selection interface and a function to easily create scrollable lists of objects. Every process in ISEIS which has X window display capability uses at least one widget in this library.

Language: C.

IPC: NONE.

******* libXe3d *******

Function: This library contains general purpose ISEIS developed Athena toolkit widget which provide facilities for three-dimensional display of data which is represented in a grid-type format. This library uses the Stardent DORE graphics package and is only supported for use on the Stardent computer.

Language: C with DORE interfaces.

IPC: NONE.

******* libXeu *******

Function: This library contains general purpose X window utilities which are used for color table manipulation, viewing transformations, marker drawing utilities and X window pixmap utilities.

Language: C.

IPC: NONE.

******* libbbfk *******

Function: This code is used for broadband FK calculations.

Language: FORTRAN.

IPC: NONE.

******* libcalcomp *******

Function: This library contains routines used by processes that utilize the LYNESS calcomp emulation plotting widgets.

Language: FORTRAN.

IPC: NONE.

******* libdbio *******

Function: This library is used for general purpose ORACLE database access and modification of relations and data associated with the *DISC ORACLE relations. These routines read and write station pick files, waveform files, spectral files, peak data and general signal processing parameters. Any manipulation of the ISEIS *DISC relations and associated data files is usually done through a routine in this library.

Language: FORTRAN and C with SQL.

IPC: NONE.

******* libenora *******

Function: This library is used for general purpose database information manipulation on frequently used ISEIS ORACLE relations. Most ISEIS processes use at least one function in this library. This library also contains standard interfaces for opening the database (logging on from embedded SQL), closing and rollback.

Language: C with SQL.

IPC: The euDbUpdt.pc routine is used by the Top Level Spreadsheet Display process to send updated event classifications to the Interactive Map process. A d_send() Dispatcher interface is present.

******* libeu *******

Function: This is the most general of the ISEIS libraries. It contains general system information routines (SysInfo.c) and the functions which are used for the Unified Process Initiation interface (prExecute* functions). It also contains error and status routines which are used system-wide for X window display of dialog boxes. Every process in ISEIS uses at least one function from this library.

Language: C and FORTRAN.

IPC: This library contains the bulk of the IPC Dispatcher function calls in ISEIS. Most ISEIS processes interface with the Dispatcher solely through one or more functions in this library.

******* libextf *******

Function: This library is used for flat file I/O. Since there are not very many ISEIS processes which use flat files any more, this library is not very heavily used. In fact, all processes should be using the ORACLE database for information manipulation. The only processes that should be using these routines are those which were ported from earlier non-database platforms. However, the wfio.c routine is used in particular by processes which need to read/write non-FORTRAN type files from FORTRAN. These would be FORTRAN processes which need to manipulate files without having to worry about the FORTRAN record length and file type restrictions in the OPEN statement.

Language: C.

IPC: NONE.

******* libfk *******

Function: This library contains general purpose FK and signal processing functions.

Language: C and FORTRAN.

IPC: NONE.

******* libiseis *******

Function: This is the most general of the ISEIS libraries.

Language: C and FORTRAN.

IPC: Contains all of the Dispatcher and ISIS interface functions.

******* libseis *******

Function: This library is used retrieving seismic information on ISEIS events and stations.
Language: C and FORTRAN.
IPC: NONE.

******* libtime *******

Function: This library contains general date/time utility routines.
Language: C and YACC files.
IPC: NONE.

B.6.2.1 ISEIS Top Level Processes

******* dsmain *******

Function: This is the ISEIS Top Level Spreadsheet Display (Section 2.1). It is the primary interface to the ISEIS system a..d it is used to execute discriminant processes and view discriminant processing results.
Input: This process accesses the MachAlloc file and the eventFile.d file. It uses the MachAlloc to determine the run time and X window display environment for the discriminant processes. It uses the eventFile.d file as a vehicle for obtaining its set of events for display. It also accesses the UNIX environment which is set up by an .Env* file. See the section on ISEIS configuration.
Output: This process both reads and creates an eventFile.d file.
Language: C with SQL.
IPC: The Top Level Spreadsheet interfaces with the Interactive Map Process and all children processes which include the discriminant processes (*.DTA, *.EXE, *.UPD) and the discriminant display and signal processing processes. The children processes send EXECUTING and

FINISHED messages to the Spreadsheet and the Map sends event lists to the Spreadsheet via the NEW_ARGS message.

Parents: NONE.

Children: *.DTA, *.EXE, *.UPD, and most others.

******* map *******

Function: The Interactive Map Display is the primary geographical interface to the ISEIS system. It displays events on a map of the user's choice and permits functions which include region definition and viewing, overlays, cross-section displays, great circle path displays and percent error ellipse displays.

Input: The Interactive Map has the same input/output requirements as the Top Level Spreadsheet. In addition, it accesses a 'MapDefaults' file for initial default parameters and it reads a 'MapInfo' file which contains specific lat/lon and projection parameters for each map image used by the Map. The Map also uses its own private X window defaults file called '.mapdefs.'

Output: The Map creates a sublist file in the ISEIS_WORKSPACE_DIR directory which contains a list of event IDs for the Spreadsheet.

Language: C and FORTRAN with SQL.

IPC: The Map can send and receive NEW_ARGS messages to and from the Spreadsheet. These messages allow new event sets to be communicated between the Map and Spreadsheet.

Parents: NONE.

Children: NONE.

******* RGBcolor *******

Function: The ISEIS Color Editor process only runs on the Stardent system and it allows color thresholds to be specified for Stardent sonogram displays.

Input: NONE.

Output: NONE.

Language:	C.
IPC:	This process processes messages for loading color tables from client processes. It also responds to requests for color tables from clients by sending the color table information to the client.
Parents:	NONE.
Children:	NONE.

B.6.2.2 <u>ISEIS Discriminant Processes</u>

These programs provide an interface between the CLIPS inference engine, the Spreadsheet and the database. They are located under the CLIPS directory (see Figure B-3).

******* *.DTA *******

Function:	Discriminant Processes of the form *.DTA are used to provide a C/SQL/CLIPS interface for discriminant data assessment. The *.DTA class of discriminants acquire information from the database and then they invoke CLIPS to determine if there is enough data available to execute a given discriminant.
Input:	*.STR CLIPS rules files.
Output:	This process writes a CLIPS trace file called a dribble to the appropriate explanations directory. It has the form *.STD. It also writes an explanations file explaining why there is, or is not, sufficient data to execute. It has the form *.DTA.
Language:	C with SQL.
IPC:	This process uses the Unified Process Initiation Interface which is explained in Section 7.2. All discriminant processes send an executing in progress message to the Spreadsheet and an executing finished or aborted message to the Spreadsheet when they are done.
Parents:	Top Level Spreadsheet Display or Auto.
Children:	NONE.

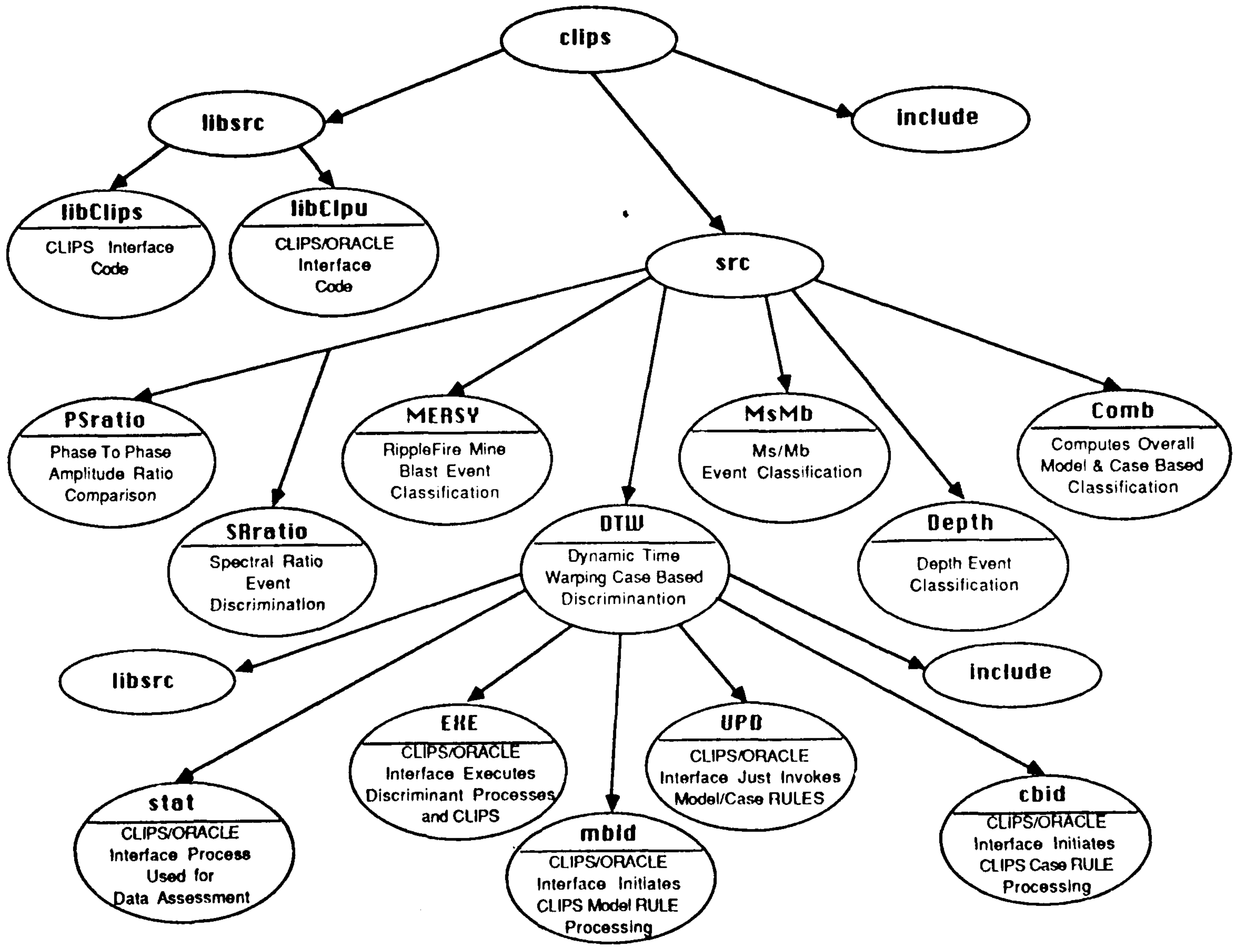
clips
libsrc
include
libClips
CLIPS Interface Code
libClpu
CLIPS/ORACLE Interface Code
src
PSratio
Phase To Phase Amplitude Ratio Comparison
SRratio
Spectral Ratio Event Discrimination
MERSY
RippleFire Mine Blast Event Classification
DTW
Dynamic Time Warping Case Based Discriminantion
MsMb
Ms/Mb Event Classification
Depth
Depth Event Classification
Comb
Computes Overall Model & Case Based Classification
libsrc
include
stat
CLIPS/ORACLE Interface Process Used for Data Assessment
EXE
CLIPS/ORACLE Interface Executes Discriminant Processes and CLIPS
mbid
CLIPS/ORACLE Interface Initiates CLIPS Model RULE Processing
UPD
CLIPS/ORACLE Interface Just Invokes Model/Case RULES
cbid
CLIPS/ORACLE Interface Initiates CLIPS Case RULE Processing

FIGURE B-3: ISEIS DISCRIMINANT PROCESSES

******* *.EXE *******

Function: Discriminant Processes of the form *.EXE are used to provide a C/SQL/CLIPS interface for discriminant execution. The *.EXE class of discriminants acquire information from the database and then they invoke CLIPS to determine the model-based and case-based classification for the given event and technique.

Input: *.MBR,*.CBR CLIPS rule files.

Output: This process writes a CLIPS trace file called a dribble to the appropriate explanations directory. It has the form *.MBD or *.CBD. It also writes an explanations file explaining the model- or case-based answer. These are of the form *.MDL and/or *.CSE.

Language: C with SQL.

IPC: This process uses the Unified Process Initiation Interface which is explained in Section 7.2. All discriminant processes send an executing in progress message to the Spreadsheet and an executing finished or aborted message to the Spreadsheet when they are done.

Parents: Top Level Spreadsheet Display or Auto.

Children: *.mbid or *.cbid process depending upon whether it is a model- or case-based process. It will start both *.mbid's and *.cbid's if it has both types of discrimination.

******* *.UPD *******

Function: Discriminant Processes of the form *.UPD are used to provide a C/SQL/CLIPS interface for discriminant rule updates. The *.UPD class of discriminants acquire information from the database and then they invoke the CLIPS *.mbid and *.cbid interface processes to perform rule-based processing ONLY on the discriminant. This differs from the *.EXE types of discriminants, since these can also execute computational processes, as well as rules. Therefore, updating of discriminants is faster than executing them.

Input: *.MBR,*.CBR CLIPS rules files.

Output: This process writes a CLIPS trace file called a dribble to the appropriate explanations directory. It has the form *.MBD/*.CBD. It also writes an explanations file explaining the model- or case-based answer. These are of the form *.MDL and/or *.CSE.

Language: C with SQL.

IPC: This process uses the Unified Process Initiation Interface which is explained in Section 7.2. All discriminant processes send an executing in progress message to the Spreadsheet and an executing finished or aborted message to the Spreadsheet when they are done.

Parents: Top Level Spreadsheet Display or Auto.

Children: *.mbid and/or *.cbid processes.

B.6.3 ISEIS DISCRIMINANT SIGNAL PROCESS SLAVE PROCESSES

******* DTWVws *******

Function: This process is used to view the results of Dynamic Time Warping Pattern Match processing.

Input: *.sta incoherent beam files and STADISC, STAPICK, DTW and DTWSTAT relations.

Output: NONE.

Language: C with SQL.

IPC: NONE.

Parents: Spreadsheet, DTWDisplay.

Children: NONE.

******* doDTW *******

Function: This process invokes the dtw process one or more times and then it invokes the DTWVws process upon completion.

Input: NONE.

Output: NONE.

Language: C with SQL.

IPC.	NONE.
Parents:	DTWexecute.
Children:	dtw and DTWVws.

******* DTWDisplay *******

Function:	This process is used to display Dynamic Time Warp Pattern Match processing results. It actually acts as a driver which starts processes to view and execute DTW.
Input:	*.sta incoherent beam files and STADISC relation.
Output:	Scratch file for input to do DTW and DTW_main.
Language:	C with SQL.
IPC:	NONE.
Parents:	Spreadsheet.
Children:	DTWVws, doDTW, DTW_main.

******* DTW_main *******

Function:	This is a computational process that runs the DTW Pattern Matcher for all the data for a given orid. This data set consists of all stations, default filter and all reference events up to a pre-determined maximum number (MXRF).
Input:	*.sta files.
Output:	STADISC,STAPICK,DTW, and DTWSTAT relations.
Language:	C with SQL.
IPC:	
Parents:	CLIPS dtw.EXE process, DTWDisplay.
Children:	NONE.

******* ardisplay *******

Function: This display process facilitates viewing of amplitude phase ratio data. Three types of plots are available. The user can view Ratio vs.

	Frequency, Ratio vs. Region and Ratio vs. Distance. Ratio is computed by dividing the average of one seismic phase with that of another.
Input:	*.sta incoherent beam files are used for display only.
Output:	NONE.
Language:	C with SQL.
IPC:	This process uses the Unified Process Initiation Interface.
Parents:	Spreadsheet.
Children:	NONE.

******* arauto *******

Function:	This process is invoked by autoDS or autoISEIS and it computes these phase ratios: Pn/Pg, Pn/Sn, Pn/Lg, Pg/Sn, Pg/Lg, Sn/Lg.
Input:	Phase picks in STAPICK relation.
Output:	NONE.
Language:	C with SQL.
IPC:	This process uses the Unified Process Initiation Interface.
Parents:	*autoDS and autoISEIS.*
Children:	NONE.

******* srdisplay *******

Function:	This process is used to view spectral ratio results in a manner similar to that of ardisplay. In this case however, only two plot types are available. These are Frequency vs. Region and Frequency vs. Distance. This process also provides an interface so that the user can select frequency windows within phase spectra.
Input:	*.fs spectra files.
Output:	SPECRATIO relation.
Language:	C with SQL.
IPC:	This process uses the Unified Process Initiation Interface.
Parents:	Spreadsheet Display.
Children:	NONE.

******* srauto *******

Function:	This process is invoked by either autoDS or autoISEIS for automatic computation of spectral ratio statistics.
Input:	NONE.
Output:	NONE.
Language:	C with SQL.
IPC:	This process uses the Unified Communications Interface.
Parents:	autoDS, autoISEIS.
Children:	NONE.

B.6.4 ISEIS SIGNAL PROCESSES

A signal processing process is defined as a process which performs some kind of computational processing using one or more signal processing algorithms and which produces a data file.

******* MERSY *******

Function:	This process computes maximum entropy and Fourier cepstra from all of the available spectra for a given orid. It also marks peaks.
Input:	*.fs spectral files. FSDISC, RESPDISC & network relations.
Output:	*.fc (Fourier cepstra) and *.mc (maximum entropy) files, CPDISC, MERSTAT, CEPPKS relations.
Language:	C , FORTRAN and SQL.
IPC:	
Parents:	CLIPS MERSY.EXE process.
Children:	NONE.

******* MERSYdpy *******

Function:	This process displays the results of the MERSY (Multiple Event Recognition System) processing.
Input:	*.fs spectra files, *.fc & *.mc cepstral files and data from FSDISC. CPDISC, RESPDISC, MERSTAT and CEPPKS relations.
Output:	NONE.
Language:	C with SQL.
IPC:	Startup messages are sent between the Spreadsheet and MERSYdpy during initialization.
Parents:	Spreadsheet.
Children:	NONE.

******* fsdisplay *******

Function:	This process is used for both Fourier spectra display and computation. It also computes signal-to-noise ratios for signals. Fourier spectra are computed for each phase picked. Noise, Instrument and Von Seggern-Blandord corrections are available.
Input:	*.w waveform files.
Output:	*.fs files.
Language:	C with SQL.
IPC:	Startup messages are sent between the Spreadsheet and this process during initialization.
Parents:	Spreadsheet Display.
Children:	NONE.

******* IBeam *******

Function:	This display process is used for parameter input and subsequent execution of new short term average (STA) incoherent beam computation or the display of previously computed incoherent beams.
Input:	*.w waveform files and data from the wfdisc relations.

Output:	Scratch parameter file for input into the STA beam program (stacomp).
Language:	C with SQL.
IPC:	Startup messages are sent between this process and parent/child processes.
Parents:	Spreadsheet.
Children:	dostacomp, IBeamVws.

******* IBeamVws *******

Function:	This process displays incoherent beams for a given orid.
Input:	*.sta incoherent beam files, STADISC relation and network relations.
Output:	NONE.
Language:	C with SQL.
IPC:	Startup messages are sent between this process and the Spreadsheet during initialization.
Parents	IBeam, Spreadsheet, dostacomp.
Children:	NONE.

******* dostacomp *******

Function:	This process controls the execution for beam computation on a list of stations and filter bands for a given orid. It invokes the 'stacomp' program to compute beams for each station and filter band. When all beams have been computed, it starts the IBeamVws beam display program.
Input:	Scratch file listing stations, filters, etc.
Output:	Scratch files for input to stacomp.
Language:	C with SQL.
IPC:	Startup messages are sent between this process and parent/children processes.
Parents:	IBeam.
Children:	stacomp.

******* stacomp *******

Function: This process computes short term averages from waveform files to produce *.sta incoherent beam data files.

Input: Scratch file containing pertinent parameters.

Output: *.sta short term average (incoherent beam files).

Language: C.

IPC: Startup messages between this process and dostacomp are sent during initialization.

Parents: dostacomp.

Children: storbmstuf.

******* storbmstuf *******

Function: This process is used to store parameters into ORACLE relations.

Input: Scratch parameter file.

Output: Updates the STADISC relation.

Language: C with SQL.

IPC:

Parents: stacomp.

Children: NONE.

******* cfkdisplay *******

Function: This process is used to setup parameters for FK analysis, view previously computed FK spectra and to initiate FK processing with currently selected FK parameters.

Input: NONE.

Output: NONE.

Language: C with SQL.

IPC: Startup messages are sent between CFKplot and cfkdisplay or CFKexecute and cfkdisplay.

Parents: Spreadsheet.
Children: CFKplot CFKexecute.

***** CFKexecute *****

Function: This computational process computes an FK spectrum from a time-series waveform. One FK is computed for each window over the waveform and the windows may or may not overlap. These FK spectra are then stacked and saved.
Input: waveform files *.w
Output: *.cfk files.
Language: C.
IPC: Startup messages are sent between CFKDisplay and CFKexecute Startup messages are also sent between CFKexecute and CFKplot when the latter program is started.
Parents: CFKDisplay.
Children: CFKplot.

***** cfkplot *****

Function: This process displays four different plots. These are the Velocity, Azimuth, F-Statistic, and Time-Series plots. As with most other display programs, the user can select station, channel and other parameters.
Input: *.cfk files.
Output: NONE.
Language: C with SQL.
IPC: Startup messages are sent between this process and the CFKDisplay process.
Parents: CFKDisplay and CFKexecute.
Children: NONE.

***** Sono3D *****

Function:	This is the Stardent resident sonogram viewing process. It allows dynamic real time viewing of sonogram data. The user can rotate, pan and zoom the image and he can also load new color table data to change the plot colors. In addition, the user can query time, frequency and amplitude information from the plot by clicking at a point on the surface.
Input:	*.cfs (continuous spectra files).
Output:	NONE.
Language:	C.
IPC:	Startup messages are sent between this process and cfsplot and color table messages are sent between this process and RGBcolor (Color Editor). The process may either load the current color table from the Color Editor or it can load its current table into the Color Editor via Dispatcher messages.
Parents:	cfsplot.
Children:	NONE.

***** CFSDisplay *****

Function:	This display process allows the user to select time ranges for noise and signal, apply signal corrections, specify input channels, specify FFT window lengths and smoothing parameters, etc. The user may also use this interface to view existing sonograms (*.cfs files) or he may compute another one using the current parameter specifications. In each of the latter two cases, either cfspect (the computational program) or cfsplot (the plotting program) will be started.
Input:	WFDISC, ORIGIN, CFSDISC relations.
Output:	NONE.
Language:	C with SQL.
IPC:	Startup messages are sent between cfspect and CFSDisplay or cfsplot and CFSDisplay depending upon menu selection.

Parents: Spreadsheet.

Children: cfsplot, cfspect.

***** **cfspect** *****

Function: This process computes stacked Fourier spectra which are eventually displayed with the Sono3D and cfsplot processes. Basically, this process computes Fourier spectra in a series of time windows over the waveform. Thus, each spectra corresponds to a different time range in the waveform. The program accepts a multitude of command line arguments which specify channels, time ranges, window lengths, etc.

Input: *.w (waveform files).

Output: *.cfs files (continuous spectra files).

Language: C.

IPC: Startup messages are sent between cfsplot and cfspect when cfspect starts cfsplot for display.

Parents: CFSDisplay.

Children: CFSplot.

***** **cfsplot** *****

Function: This process is used to display continuous Fourier spectra. These data are computed by stacking spectra computed from sliding time-series windows. Thus, the data show time-dependent frequency characteristics. The data is displayed as a three-dimensional surface representation with frequency vs. time vs. amplitude (vertical axis). Two 3D plot types are available. The Mesh plot option results in a non-dynamic line-plot surface display which appears on the same screen as this program. The DORE plot option causes cfsplot to start Sono3D on the Stardent for dynamic sonogram viewing.

Input: *.cfs files produced by cfspect.

Output: NONE.

Language: C with SQL.

IPC: Startup messages are sent between cfsplot and Sono3D when the Dore sonogram plot program is started.

Parents: cfspect, CFSDisplay.

Children: Sono3D.

B.6.5 THE COMPARE PROCESS

******* cmpmain *******

Function: The compare process is used for graphical waveform comparison. It provides a ZOOM capability and a waveform sliding capability. This allows users to drag reference event waveforms and overlay them over current event waveforms to check common features. The user can create as many additional windows as he desires with the ZOOM feature.

Input: *.w waveform files.

Output: NONE.

Language: C *with* SQL.

IPC: This process uses the Unified Process Initiation Interface.

Parents: Top Level Spreadsheet Display.

Children: NONE.

B.6.6 THE IPC AND SYSTEM CONTROL PROCESSES

******* Dispatcher *******

Function: This is a program developed by SAIC which is used for IPC message routing. It provides an easy-to-use inter-process communications interface which allows any process to communicate with any other process in synchronous or asynchronous modes.

Input: NONE.

Output: log.[hostname] file which contains history of usage.

Language: C.

IPC:	It is the IPC.
Parents:	NONE.
Children:	NONE.

******* Monitor *******

Function:	This ISEIS process is used to start processes on the system where it is running and it also services GET_SYSTEM_LOAD messages by sending the system cpu loading statistic for its system to the requesting process. It also provides a mechanism for ISIS/Dispatcher communication.
Input:	NONE.
Output:	Temporary $W/[hostname]MON.stat file which contains redirected output of UNIX 'uptime' command.
Language:	C.
IPC:	Uses Dispatcher and ISIS interfaces.
Parents:	NONE.
Children:	*Started process with system() inherit the current environment.*

B.6.7 THE AUTOMATIC DISCRIMINATION PROCESSES

******* autoDS *******

Function:	This process controls the entire automatic processing for a given orid. The process is started from the Spreadsheet and this process first invokes autophase to generate phase picks and then it invokes incoherent beam processing from the doIBeam() interface. Once beams are available, the process starts the CLIPS *.DTA processes and then the *.EXE processes. The Combined ID *.DTA process is started after all *.EXE processes have been completed and the *.EXE Combined ID process is started after completion of the *.DTA process.autoDS starts all *.DTA processes simultaneously and waits for all to complete before starting the *.EXE processes.

Input:	NONE.
Output:	database updates.
Language:	C.
IPC:	This process uses the Unified Communications Interface.
Parents:	Spreadsheet Top Level Display.
Children:	IBeam, srauto, arauto, autophase and CLIPS processes.

******* autoISEIS *******

Function:	This process is used for offline periodic execution of automatic processing. It is designed to be run in a batch mode at regularly scheduled intervals.
Input:	NONE.
Output:	NONE.
Language:	C with SQL.
IPC:	This process uses the Unified Communications Interface.
Parents:	NONE.
Children:	IBeam, srauto, arauto, autophase and CLIPS processes.

******* autophase *******

Function:	This program is part of the automatic processing package. This process is used to pick seismic phases (time windows) using distance from event to station and group velocity parameters. It will attempt to define Pn, Pg, Lg, and Sn phases. These phases are needed for IBeam other subsequently executed automatic processes.
Input:	ARRIVAL relation.
Output:	STAPICK.
Language:	C with SQL.
IPC:	This interface uses the Unified Communications Interface.
Parents:	autoDS.
Children:	NONE.

APPENDIX C - DATABASE RELATIONS

This appendix contains descriptions of the new database relations required by the ISEIS system. These descriptions are presented in the form of table descriptions for the contents of each new relation. These descriptions are divided into the following groups:

<u>Discriminant Feature Relations</u>
- ampratio
- specratio

<u>Relations for Multiple Event Recognition System (MERSY)</u>
- ceppks
- cpdisc
- merstat
- pceppks
- respdisc

<u>Relations for Dynamic Time Warp (DTW)</u>
- dtw
- dtwstat

<u>Incoherent Beam and Phase Pick Relations</u>
- stadisc
- stapick

<u>Propagation Path and Discriminant Regionalization Relations</u>
- path
- refevent
- refregion

<u>Signal-To-Noise Ratio Relations</u>
- SNRfs
- SNRib

<u>Discriminant Results and Rule Relations</u>
- distech
- ruledisc

<u>Miscellaneous Signal Analysis Relations</u>
- cfsdisc
- cfkdisc

Discriminant Feature Relations

The following relations are generated by two programs which compute regional phase amplitude ratios and phase spectral ratios, generated by the ISEIS processes ARDISPLAY and SRDISPLAY, respectively.

The amplitude ratio is derived from the incoherent beams and the stapick relations. In essence, the amplitude ratios are the ratios of the maximum amplitudes or the average of the short-term-averages, *sta*, in user-specified windows on an incoherent beam.

ampratio				
attribute name	external type	storage positions	character format	attribute description
orid	i4	i8	1-8	origin id
numid	i4	i8	10-17	numerator sta arrival id
denomid	i4	i8	19-36	denominator sta arrival id
ratio	f4	e12.6	28-39	ratio value
name	c10	a10	41-50	name of ratio
freqmin	f4	f6.2	52-57	low frequency cutoff
freqmax	f4	f6.2	59-64	high frequency cutoff
method	c7	a7	66-76	amplitude ratio method
lddate	date	a17	78-94	load date

where *method* is one of the following values:
icbmax - maximum incoherent beam amplitudes
icbave - average incoherent beam amplitudes

The specratio relation contains the spectral-ratio feature, specified by ratios between two different spectral bands for a seismic phase.

specratio				
attribute name	storage type	external format	character positions	attribute description
orid	i4	i8	1-8	origin id
arid	i4	i8	10-17	arrival id
ratio	f4	f8.2	19-26	ratio value
phase	c8	a8	28-35	spectral phase
numfmin	f4	f8.2	37-44	numerator minimum frequency
numfmax	f4	f8.2	46-53	numerator maximum frequency
denomfmin	f4	f8.2	55-62	denominator minimum frequency
denomfmax	f4	f8.2	64-71	denominator maximum frequency
method	c7	a7	73-79	spectral ratio method
noise	c1	a1	81-81	y/n corrected spectra for noise
inst	c1	a1	83-83	y/n corrected spectra for inst resp
q	c1	a1	85-85	y/n corrected spectra for Q
q0	f4	f8.2	87-94	Q at frequency f0
zeta	f4	f8.2	96-103	freq dependence of Q
f0	f4	f8.2	105-112	reference frequency for Q0
lddate	date	a17	114-130	load date

where *method* is one of the following values:

MAX - maximum spectral value

RMS - root mean square spectral values different bands in the spectrum of a phase.

Relations for Multiple Event Recognition System (MERSY)

The following are the relations generated by the Multiple Event Recognition System (MERSY). MERSY generates cepstra and extracts features for the cepstra to be used for the identification of events which have time independent spectral modulations which, in turn, produce time independent cepstral peaks.

Ceppks contains a record of the consistent cepstral peaks across the phases of an event.

ceppks				
attribute name	storage type	external format	character positions	attribute description
orid	i4	i8	1-8	Origin Id
sta	c6	a6	10-15	station code
ptyp	c6	a6	17-22	consistent peak type code
pkamp	f4	e10.3	24-33	consistent peak - amplitude
pkqf	f4	e10.3	35-44	consistent peak - quefrency
lddate	date	a17	46-62	load date

where *ptyp* is the type of consistent peak. Valid consistent peak types are the following:

FC-PHS	consistent peak across Fourier cepstra of phases.
MC-PHS	consistent peak across Maximum Entropy cepstra of phases.
FC-NOI	significant Fourier noise peaks.
MC-NOI	significant Maximum Entropy noise peaks.
FC-ARY	consistent peak across Fourier cepstra of array stacks.
MC-ARY	consistent peak across Maximum Entropy cepstra of array stacks.

The cpdisc relation points to the cepstra which are stored as flat files on disk.

cpdisc				
attribute name	storage type	external format	character posititions	attribute description
arid	i4	i8	1-8	arrival id
fsrid	i4	i8	10-17	fourier spectrum recipe id
flen	f4	f15.3	19-33	frequency window
lfcut	f4	f6.2	35-40	low frequency cutoff
ceptyp	c6	a6	42-47	cepstrum type (e.g., FC-SNG; MC-SNG)
cpid	i4	i8	49-56	cepstrum ID
cprid	i4	i8	58-65	fc recipe ID
mxquef	f4	f9.4	67-75	maximum quefrency value
nquef	i4	i6	77-83	number of quefrency values
nmcoef	i4	i3	85-87	number of coefficients used for maxent cepstra
acoef	f4	e12.5	89-100	a coefficent for nonlinear trend
bcoef	f4	e12.5	102-113	b coefficent for nonlinear trend
ccoef	f4	e12 5	115-126	c coefficent for nonlinear trend
datsw	i4	i8	128-135	data switch
foff	i4	i8	137-144	byte offset in file
dir	c30	a30	146-175	cepstrum directory
dfile	c20	a20	177-196	cepstrum data file
lddate	date	a17	198-214	load date

The merstat relation contains statistical parameters from the spectrum and cepstrum of an arrival.

merstat				
attribute name	storage type	external format	character positions	attribute description
orid	i4	i8	1-8	Origin Id
sta	c6	a6	10-15	station code
cpid	i4	i8	17-25	cepstrum Id
ceptyp	c6	a6	27-32	cepstrum type code
snrdb	f4	f6.2	34-39	spectrum snr (db)
svar	f4	f6.3	41-46	detrended log spectral variance
skew	f4	f6.2	48-53	detrended spectral skew
skur	f4	f6.2	55-60	detrended spectral kurtosis
cvar	f4	f6.3	62-67	log cepstral variance
cskew	f4	f6.2	69-74	cepstral skew
ckur	f4	f6.2	76-81	cepstral kurtosis
lddate	date	a17	83-109	load date

where *ceptyp* is the type of cepstrum. Valid cepstrum types are the following:

FC-SNG	Fourier cepstrum - single phase

FC-STK	Fourier cepstrum - phase stack
MC-SNG	Maximum Entropy cepstrum - single phase
MC-STK	Maximum Entropy cepstrum - phase stack

Pceppks is a record of the significant cepstral peaks in each phase of an event. These differ from those in ceppks in that they are not necessarily consistent across two or more phases. Although MERSY produces this relation, it is not currently being used by ISEIS. It is envisioned that it may be used in a special cepstral depth discriminant, where these cepstral peaks may be an indication of depth phases.

pceppks				
attribute name	storage type	external format	character positions	attribute description
arid	i4	i8	1-8	Arrival Id
cpid	i4	i8	10-17	cepstral Id
ptyp	c6	a6	19-24	consitent peak type code
pkamp	f4	e10.3	26-35	consistent peak - amplitude
pkqf	f4	e10.3	37-46	consistent peak - quefrency
lddate	date	a17	48-64	load date

The respdisc relation points to the instrument response which MERSY needs to correct the *spectra prior to computation of the cepstra.*

respdisc				
attribute name	storage type	external format	character posititions	attribute description
net	c8	a8	1-8	netword name
nfreq	i4	i8	10-17	number of freq points in file
dfile	c20	a20	19-38	file name
dir	c30	a30	40-69	directory containing file
lddate	date	a17	70-86	load date

Relations for Dynamic Time Warp (DTW)

The following relations are generated by the dynamic time warp (DTW) incoherent-beam template matcher.

The dtw relation contains descriptive information for a Dynamic Time Warp (DTW) match of two STA incoherent beams.

dtw				
attribute name	external type	storage positions	character format	attribute description
bmidt	i4	i8	1-8	beam id for template
bmide	i4	i8	10-17	beam id for event
wrpmid	i4	i8	19-26	warp match id
wrprid	i4	i8	28-35	warp match recipe id
nxtwmid	i4	i8	37-44	next warp match id
prvwmid	i4	i8	46-53	previous warp match id
phsnrm	i4	i3	55-57	flag for phase used for normalization
tmpmax	f4	e12.6	59-70	max in normalization phase for template
evtmax	f4	e12.6	72-93	max in normalization phase for event
mxstrch	i4	i6	95-100	max stretch for match
taccdst	f4	e12.6	102-123	total accumulated distance for match
lddate	date	a17	125-141	load date

where *phsnrm* has the one of the following values:

1	-Pn phase
2	-Pg phase
3	-Sn phase
4	-Lg phase
5	-All phases

Dtwstat contains the results of the phase matching for a given DTW match.

dtwstat				
attribute name	storage type	external format	character positions	attribute description
wrpmid	i4	i8	1-8	warp match id
phsid	a6	a6	10-15	phase label (ie. Pn, Lg)
strttmp	i4	i8	17-25	start point of phase in template
npttemp	i4	i8	27-34	number of phase points in template
strtevt	i4	i8	36-43	start point event
nptevt	i4	i8	45-52	number of phase points in event
accdist	f4	e12.6	54-75	accumulated distance for phase
lddate	date	a17	77-93	load date

Incoherent Beam and Phase Pick Relations

These relations relate to the incoherent beams (IBEAM) and phase picks make on incoherent beams.

The stadisc relation points to incoherent beams, stored as flat files on disc, and contains the file header and descriptive information. These relations are produced by the ISEIS process IBEAM.

stadisc				
attribute name	external type	storage positions	character format	attribute description
ondate	i8	i4	1-8	date of event
time	f15.3	f8	10-24	epoch time for beam
incbmid	i8	f4	26-33	beam id
orid	i8	i4	35-40	origin id for event
sta	a6	a6	42-49	station code
incbmrid	i8	f4	51-56	beam recipe id
wrpmid	i8	i4	58-65	warp match id
filtid	i6	i4	67-71	filter id
filtlow	f6.2	f4	73-82	low frequency - filter band
filthgh	f6.2	f4	84-89	high frequency - filter band
twinlen	f6.2	f4	91-96	time in each window for beam
twshft	f6.2	f4	98-103	shift between windows for beam
nptbm	i8	i4	105-112	total number of beam points
datsw	i10	i4	114-123	data switch
foff	i10	i4	125-124	file offset
dir	a30	a30	126-155	directory containing beam file
dfile	a20	a20	157-176	name of beam file
lddate	date	a17	178-194	load date

Stapick marks a phase onset time, *time*, and the time duration, *tlen*, on an incoherent beam. It also contains estimates of the maximum and average amplitude values for the phase designated by *phsid* within the time window defined by the onset time and duration. Stapick relations are generated by user interactive phase selection functions in the ISEIS processes ARDISPLAY and DTWDISPLAY.

stapick				
attribute name	external type	storage positions	character format	attribute description
orid	i4	i8	1-8	origin id
incbmid	i4	i8	10-17	incoherent beam id
phsid	a6	a6	19-24	phase label (ie. Pn, Lg)
starid	i4	i8	26-33	sta arrival id
time	f4	f15.3	35-49	time for phase (from sta beam epoch)
tlen	f4	f15.3	51-65	duration of phase
pmax	f4	f15.3	67-81	Phase maximum value
pave	f4	f15.3	83-97	Phase average value
lddate	date	a17	99-115	load date

Propagation Path and Discriminant Regionalization Relations

The path relation contains anelastic attenuation parameters for the propagation path from a source region, indexed by a unique *regid* number, to the station, designated by *sta*. This relation is produced by user interaction in the spreadsheet.

path				
attribute name	storage type	external format	character positions	attribute description
regid	i4	i8	1-7	region id
sta	c6	a6	9-14	station code
phase	c8	a8	16-23	seismic phase
q0	f4	f8.2	25-32	Q at reference frequency
zeta	f4	f8.2	34-41	frequency dependence of Q
f0	f4	f8.2	43-50	reference frequency
gvel	f4	f8.2	52-59	group velocity
lddate	a17	date	61-77	load date

The following are the definitions of the relations for designating reference event regions and the reference events within them. Reference events are assigned to regions with the refevent relation, which associates a unique geographic geographic region index, *regid*, with any number of *orids* for reference events located within the region. The reference events are also assigned to specific stations or arrays, designated by *sta*. These relations are produced by user interaction with the ISEIS MAP process.

refevent				
attribute name	external type	storage positions	character format	attribute description
orid	i4	i8	1-8	origin id
regid	i4	i8	10-17	region id
sta	c6	a6	19-24	station code
lddate	date	a17	26-43	load date

The refregion relation defines a region centroid and ellipse about the centroid, indexed by a *regid.*

refregion				
attribute name	external type	storage positions	character format	attribute description
regid	i4	i8	1-8	region id
regname	c20	a20	10-29	region name
lat	f4	f9.4	31-39	region centroid latitude
lon	f4	f9.4	41-49	region centroid longitude
smaj	f4	f9.4	51-59	NS axis of region ellipse
smin	f4	f9.4	61-69	EW axis of region allipse
commid	i4	i8	71-78	comment id
lddate	date	a17	80-96	load date
latmin	f4	f5.2	98-92	minimum latitude
latmax	f4	f5.2	94-98	maximum latitude
lonmin	f4	f6.2	100-104	minimum longitude
lonmax	f4	f6.2	106-110	maximum longitude

Signal-To-Noise Ratio Relations

The following two relations contain signal-to-noise information, measured in two different ways.

The SNRfs relation contains signal-to-noise ratios measured in the frequency domain by the ISEIS process FSDISPLAY.

SNRfs				
attribute name	external type	storage positions	character format	attribute description
arid	i4	i8	1-8	origin id
snr	f4	f6.2	10-15	avg signal / avg noise
minf	f4	f6.2	17-22	minimum frequency
maxf	f4	f6.2	24-29	maximum frequency
lddate	date	a17	31-47	load date

The SNRib relation contains signal-to-noise ratio measurements made in the time domain on incoherent beams by the ISEIS process ARDISPLAY.

SNRib				
attribute name	external type	storage positions	character format	attribute description
orid	i4	i8	1-8	origin id
incbmid	i4	i8	10-17	incoherent beam id
sta	c6	a6	19-24	station
phase	c6	a6	26-31	phase name
snr	f4	f6.2	33-38	(max or avg) signal / (avg) noise
minf	f4	f6.2	40-45	minimum frequency
maxf	f4	f6.2	47-52	maximum frequency
typns	c1	a1	53-53	noise type (b=background, d=drag)
typsig	c3	a3	55-57	signal type (MAX,AVG)
strtns	f8	f15.3	59-73	noise epoch start time (if drag)
lenns	f4	f6.2	75-80	noise window length in seconds (if drag)
lddate	date	a17	82-98	load date

Discriminant Results and Rule Relations

The following relation contains information about a particular discriminant result. One of these must be defined for each discriminant on the spreadsheet.

The distech relation gives the results of event identification processing. Each discriminant technique is identified by means of a *techid*, a number which is associated with a technique name, or *technam*. The disctech relation is put out each time CLIPS rules are fired to determine status or when a discriminant is run on an event, identified by a unique *orid*.

This relation is described below:

disctech				
attribute name	external type	storage positions	character format	attribute description
orid	i4	i8	1-8	origin id
techid	i4	i4	10-13	technique id
technam	c20	a20	15-34	technique name
status	c6	a6	36-41	data status
model	c6	a6	43-48	model-based result
mconf	f4	f5.3	50-54	model-based result confidence
case	c6	a6	56-61	case-based result
cconf	f4	f5.3	63-67	case-based result confidence
edir	c64	a64	69-132	explanation directory
efile	c32	a32	134-165	explanation file
lddate	date	a17	167-183	load date

Data status, *status*, takes on one of the following values:
- c - complete, all necessary data is available for the discriminant
- i - incomplete, only a portion of the necessary is available for the discriminant
- nd - no data is available for the discriminant

The model-based result, *model*, takes on one of the following values:
- qb - quarry blast or mining explosion
- eq - earthquake
- me - marine explosion
- ex - other explosion
- unm - unidentified from model-based reasoning

The case-based result, *case*, takes on one of the following values:
- s - similar to the nearest reference events
- d - dissimilar to the nearest reference events
- unc - unidentified from case-based reasoning

Note that *mconf* is a confidence value, between 0.0 and 1.0, which is determined by model-based rules and is the confidence of the model-based event identification, *model*. The case-based confidence, *cconf*, is determined by computing the confidence of match of the discriminant feature to that of the nearest reference events and subtracting it from 1.0.

The ruledisc relation points to the discrimination decision rules, which are stored as CLIPS code in flat files on disk.

ruledisc				
attribute name	external type	storage positions	character format	attribute description
techid	i4	i8	1-8	technique id
dir	c128	a128	10-137	directory name
dfile	c128	a128	139-266	file name
lddate	c17	a17	268-284	load date
datype	a2	c2	286-287	decision type
technam	c20	a20	289-308	technique name
rsltfile	c64	a64	310-373	CLIPS rule file name
datyflg	i4	i8	375-384	decision type flag

Miscellaneous Signal Analysis Relations

These relations are generated by two processes: CFSDISPLAY, which generates continous spectra (sonograms), and CFKDISPLAY, which computes fk-measured velocity, azimuth, and f-statistic templates.

The cfsdisc relation points to sonograms stored on disk in the form of cfs flat files.

cfsdisc				
attribute name	storage type	external format	character positions	attribute description
orid	i4	i8	1-8	origin id
sta	c6	a6	10-15	station code
jdate	i4	i8	17-24	julian date
sgtime	f8	f15.3	26-40	first signal window epoch time
nstime	f8	f15.3	42-56	first noise window epoch time
sglen	i4	i8	58-65	signal spectrum window length (pts)
nslen	i4	i8	67-74	noise spectrum window length (pts)
shift	i4	i8	76-83	window time shift (pts)
nchan	i4	i8	85-92	number of channels used in averaging
mnfreq	f4	f8.4	94-101	minimum frequency
mxfreq	f4	f8.4	103-110	maximum frequency
nfreq	i4	i8	112-119	number of frequency points
noise	c1	a1	121	y/n noise correction
inst	c1	a1	123	y/n instrument correction
vsb	c1	a1	125	y/n VSB source correction
taper	f4	f6.2	127-132	percent window taper
smooth	i4	i8	134-141	number of Hanning window smooths
cfsdir	c60	a60	143-202	cfs file directory prefix name
cfsfile	c30	a30	204-233	cfs file prefix name
lddate	date	a17	235-251	load date
smprat	f8	f9.5	253-261	sampling rate

The cfkdisc relation points to continuous fk template flat files stored on disk which contain the values of fk measured f-statistic, velocity, and azimuth for successive time increments.

cfkdisc

attribute name	storage type	external format	character positions	attribute description
orid	i4	i8	1-8	origin id
sta	c6	a6	10-15	station code
jdate	i4	i8	17-24	julian date
time	f8	f15.3	26-40	first window epoch time
len	i4	i8	42-49	length of spectral windows (pts)
nwin	i4	i8	51-58	number of windows
shift	i4	i8	60-67	window shift (pts)
freqlow	f4	f8.4	69-76	minimum bb fk requency
freqhigh	f4	f8.4	78-85	maximum bb fk frequency
filtlow	f4	f8.4	87-94	low prefilter frequecy
filthigh	f4	f8.4	96-103	high prefilter frequency
dir	c60	a60	105-164	cfs file directory prefix name
cfile	c32	a30	166-195	cfs file prefix name
lddate	date	a17	197-213	load date
smprat	f4	f9.5	215-223	sampling rate

PPENDIX D - CLONING DISCRIMINANTS

In order to allow researchers to evaluate a multitude of options concerning a discriminant without losing the original set of rules, we have developed a method of "cloning" a discriminant. This basically involves duplicating all of the resource files and database entries for the existing discriminant and providing a copy of the rules which can be modified to test new theories without disturbing the original set of rules. The restriction on the "clone" is that the process must use the same set of processing routines and extract the same data from the database for evaluation. We have an example of this situation in our amplitude ratio process. Here we have three sets of rules that use, respectively, the maximum P/S ratio, the high frequency Pn/Sn ratio and the high frequency Pn/Lg ratio. Each of these discriminants uses the same discrimination software and data, but each is distinguished by its own technique id number **(TECHID)**. This TECHID allows us to invoke a different set of rules for the data which produces varied results. This appendix describes the software which is used to generate the discriminant clone. The operator interaction involved is described briefly as it contributes to the design of the software, but the details of the interaction are left to the operator's manual.

APPENDIX E - CREATING NEW DISCRIMINANTS

In Appendix D, we described the software for cloning a discriminant. This appendix describes the effort involved in adding a totally new discriminant to the system.

All discriminants implemented in the ISEIS system must be developed within the framework of the five processes described in Section 3.0 In essence, the discrimination will consist of the data assessment rules, the case-based rules, and the model-based rules, possibly accompanied by some specific data processing and parameter generation. The new discriminant developer must provide the data processing task as a separate UNIX task which can be invoked from the EXE process. The developer must then supply the rules for data assessment and discrimination processing. Once these elements are provided, the new discriminant can be added to the system by duplicating the software contained in the EXE, UPD, stat, cbid, and mbid directories contained in the discriminant template directory, and providing the appropriate information where indicated within the module templates. The developer must then add the appropriate entry to the **MachAlloc** file. Once competed, the new discriminant will then become available on the top level spreadsheet and will automatically be included in the composite discrimination identification.

APPENDIX F - KNOWN PROBLEMS AND SOLUTIONS

This appendix describes system deficiencies and suggestions for improvement.

F.1.0 STARDENT/SUN INCOMPATIBILITIES AND WORK AROUNDS

Stardent/SUN compatibility issues have created some significant problems during the course of the ISEIS project. Although the Stardent possesses superior performance, it lacks the UNIX SCCS utility for source code control; it currently lacks ORACLE and/or SQLNET; and its System V implementation only supports 14 character file names.

F.1.1 NO SOURCE CODE CONTROL SYSTEM (SCCS) ON THE STARDENT

The SCCS utility was very important for ISEIS configuration management. A few notes about the ISEIS directory hierarchy scheme will elucidate the problems we faced with SCCS and the Stardent.

Figure B-1 illustrates how the ISEIS directory hierarchy is designed. There is a 'common' subdirectory under ./iseis and then a 'stardent', 'sparc' and 'sun' subdirectory. To compile and link modules for a Stardent type computer, one logs on to a Stardent and invokes 'make' in the ./iseis/stardent directory. A similar procedure is followed to create executable code to run on Sun4 (sparc) and Sun3 (sun) platforms. Each subdirectory under 'common', 'sun', 'sparc', and 'stardent' has an SCCS directory in it which contains the change history of all files in the subdirectory. The SCCS directories in the 'sun', 'sparc' and 'stardent' are symbolically linked to the actual SCCS directories under 'common.' Thus, source code is actually only maintained under the ./iseis/common directory, but no executable modules are built under ./iseis/common. Since all systems have access to the complete hierarchy via NFS, code can be modified on any system by checking it out of SCCS, changing it and checking it back into SCCS. Once code is checked back in, the UNIX 'make' utility automatically determines what files have been updated in SCCS and extracts them prior to compilation. This scheme insures that changes are incorporated into each version of a particular executable on all systems.

However, since the Stardent has no SCCS utility, we had to define alias's which perform a remote shell on a Sun type system to do the SCCS extraction prior to compilation on the Stardent. This was very awkward, to say the least. However, once the login environment was configured, this work around was acceptable.

What was tiresome however, was the UNIX System V 14 character filename standard. A significant amount of ISEIS code was taken from previous projects and many of these modules had filenames in excess of 14 characters since the Berkeley UNIX standard supports 32 character filenames. Consequently, these filenames had to be shortened to 12 characters. Twelve character length file names were adopted because SCCS prepends two characters onto the filename when it is checked in. Thus, the last two characters are lost if a 14 character filename is checked into SCCS and then extracted. This formality introduced a lot of tedium into the project.

The moral of this story is that filenames should be kept as short as possible if code re-usability and portability are desired. PC-based DOS systems only support nine character file names. One can always change filenames when porting code to other platforms, but this necessitates Makefile modification as well. Incidentally, Makefiles had to be modified quite often during the course of the project and this task becomes quite burdensome as the number of Makefiles increases. More on this later.

F.1.2 NO ORACLE OR SQLNET AVAILABLE FOR STARDENT

The lack of ORACLE and/or SQLNET support on the Stardent significantly contributed to poor system performance. One of the central goals in the design of ISEIS involved intelligent allocation of resources. However, since about 90% of the process in ISEIS require ORACLE database access, these processes could not be run on the Stardent. In an attempt to harness some of the computational power of the Stardent, designers split their respective processes into subprocesses. A typical protocol goes as follows:

1. Start a process on a machine with database support to get pertinent data from the database and write it to a scratch file.

2. Start a computational process on the Stardent which utilizes the database parameters.

3. Create a scratch file with database parameters to store and then start a process on a machine with database support to update the database with the new information.

This scheme is very costly since three processes have to be started, whereas only one process would be needed if the Stardent had database support. It is doubtful whether any significant time was saved by splitting the load in such a manner. The overhead involved with scratch file creation and process initiation nullified any Stardent computational advantage. In informal benchmarks, the Stardent performed tasks about 25% faster than the Sun4 sparcstation. As of this writing, the Stardent corporation does not suppon ORACLE for their machine. A clean way to improve system performance with the existing architecture would involve the implementation of remote procedure calls (RPCs). Using this scheme, processes would access the database from a machine with ORACLE database support and they would invoke RPCs on the Stardent for computationally intensive tasks. An RPC invocation looks just like a standard C function call to the client process. A modification of this type should speed the production of incoherent beams, spectra, cepstra and DTW statistics. Ideally, every computational process in ISEIS should be profiled to pinpoint exactly where performance is lagging. We have found that most of our performance bottlenecks occur during database access, but algorithmic performance could possibly be enhanced as well.

F.1.3 INEFFICIENT USE OF THE DATABASE

A good portion of the ISEIS database interface code inefficiently accesses the database. In some cases, SELECTs are invoked repeatedly within loops where it would have been more efficient to use CURSORs. In other cases, CURSORs are DECLARED and invoked within other CURSORs. In most cases, only a single CURSOR may be used, but the SQL array FETCH feature is not utilized. It takes about as long to fetch an array of database values as it does to perform a single FETCH. Thus, much time can be saved when fetching large amounts of data from the database. This time saving is especially important when using ORACLE over a network via SQLNET.

Nested CURSOR FETCHes could perhaps be eliminated through use of the PREPARE statement which allows dynamic SQL statements. If the SQL IN() operator is used with a PREPARE, it may be possible to eliminate nested CURSOR accesses. The getbmdat.pc routine which is used by DTW is a likely place to start since DTWDisplay and DTWVws consume the most amount of time for database access.

ORACLE indices can also be used. We did not utilize indices very much in this project since our test database was rather small and any performance gain due to indexing would be negligible. However, we did index the WFDISC relation because there are a good number of waveforms in our database. Indexing should be tried first before any code is modified since it is the easiest thing to do.

A common irritation in ISEIS is the delay that occurs when the user selects an option from the Spreadsheet PROCESS menu and when the process's display actually appears. Often, the user will re-select the option thinking that he did not mouse correctly. This delay is usually caused by initial database access prior to window display. It would be better to display the window immediately, and then do the database access, when an option requiring data from the database is selected. This would reduce user anxiety caused by the delay. Currently, the Spreadsheet displays a status message indicating that a delay is expected, but immediate display of the process window would be preferable. Such a modification, however, would require moderate reworking of the display processes since status flags must be added indicating whether database data is available or not. Basically, each designer must shift database access from an initialization function to an event-driven function. This sort of scheme is currently implemented in the COMPARE process, but it does not yet exist in any of the other PROCESS menu display processes.

F.1.4 PROGRAM DEVELOPMENT ENVIRONMENTAL DIFFERENCES

There were a few strange differences between the Sun type and Stardent program development environments. C compilers were fairly compatible between the two systems, although the Stardent version was a little more verbose with warning messages. FORTRAN code ported fairly well from the Sun to Stardent environment, as well. However, executable modules which consisted of both C and FORTRAN code segments proved to be rather difficult to work with. Any C modules which will be called from FORTRAN on the Suns must end with a trailing '_', whereas these same modules on the Stardent must consist of all CAPITAL letters. This difference required the addition of #ifdef statements in all files which have FORTRAN functions.

All ISEIS designers complained vehemently about the Stardent source code debugger. It is awkward to use, the syntax is brutal and it tends to crash easily. On the Stardent, it was practically impossible to debug executable modules which consist of both FORTRAN and C software. Designers had to insert print statements to trace program execution The debugger somehow became confused by these programs. It is also hard to examine character strings with the Stardent debugger. The general practice required that the developer first obtain the address of the string and then display it via a hex dump. If you want to debug a module, you have to be in the directory where source file containing the module resides. ISEIS designers found this to be extremely annoying and inconvenient. There is a FILE debugger command which is supposed to get around this limitation, but it didn't work very well.

There were also problems with X window implementations between Sun and Stardent type systems. The Stardent was delivered with X Window Version II and the Suns use X Window Version 4.0. Since many X window version 4.0 toolkit resources are not available for Version 2.0, some Version 4.0 widgets could not be used on the Stardent. In particular, the SimpleMenu widget was not available on the Stardent. All ENSCO-developed widgets had to be written with X Window Version #ifdef statements since the CORE widget structures are defined differently between X Version 2 and Versions 3 and 4. About ten X toolkit widgets were developed on the ISEIS project. Stardent later came out with an X Window Version 3.0 package about halfway through the project. However, it proved to be too bug-ridden to be usable and was abandoned. A workable X Window 3.0 package would eliminate the need for the #ifdef statements. Stardent then came out with Version 3.0 of its operating system. However, the new awm X window manager crashed whenever attempts were made to draw arcs. Calls were made to the Stardent technical support staff to address the above problems. The X toolkit widgets seemed to function properly with the 3.0 system. The 3.X Version operating system is required for the faster P3 CPU boards which have not yet been installed. Stardent technical support will ship a 3.1 Version of the operating system toward the end of the project which is supposed to address some of the problems discussed above. The P3 CPU boards are supposed to boost CPU performance by a factor of two to three times. The new operating system and CPU boards will be installed and tested if time permits.

F.1.5 OTHER DIFFERENCES BETWEEN UNIX SYSTEM V AND BERKELEY

The main conflict between the Stardent (System V) and SUN (Berkeley) UNIX operating systems involved file name length conventions and these were discussed above. In addition to this problem, the System V commands sometimes accept different option flags than their Berkeley counterparts. The 'ps' command was used on the project by csh scripts to query program execution status. It was particularly used by startup scripts which first check to see if a program is running before attempting to start it. Due to system differences, the PSFLAGS environmental variable had to be created so that the proper flags would be used when the 'ps' command was used on different systems. This was not a serious problem, however. Most other commands were fairly consistent.

F.2.0 INTER-PROCESS COMMUNICATION ISSUES

The ISEIS system was developed using the older Dispatcher inter-process communication program. Most of the NMRD software however, uses the ISIS inter-process communications package. Ideally, ISEIS should use ISIS for all inter-process communication. Currently, there is an ISIS/Dispatcher interface built in to each of the system monitors.

F.2.1 TOO MANY INDIVIDUAL PROCESSES

The ISEIS system currently has too many individual processes and performance would probably improve if the number of processes were reduced. The lack of Stardent ORACLE support contributed to some of the extra processes, as described in the Database section of this appendix. There should be individual processes for the Spreadsheet, Map, Color Editor, Compare and each of the discriminant processes (*.DTA, *.EXE, *.UPD), at least. In addition, individual processes should exist for all of the processes which are started from the Spreadsheet PROCESS menu. This minimum set of processes promotes modularity and future system expansion (more discriminants). It is not worth executing a separate process if it will only require a few seconds to execute. Most of the processes invoked under the PROCESS menu could be merged. An ideal configuration would probably consist of a collection of processes which run on Sun type systems and that perform numerical computation on the Stardent system via RPC interfaces. RPCs are discussed in the database section. However, if Stardent were to provide an ORACLE or SQLNET package in the future, it would be preferable to purchase this package and install it on the Stardent.

F.2.2 CHANGE DISPATCHER INTERFACE TO ISIS INTERFACE

For compatibility, all ISEIS inter-process communications interfaces should be changed from Dispatcher interfaces to ISIS interfaces. This is a medium-size task. The changes should be rather localized, however. Most processes do not invoke dispatcher calls directly, but use them indirectly via the Unified Process Initiation Interface described in another section of the Maintenance Manual. However, processes which do perform dispatcher calls directly and those that process message arrival events via XtAddInput() would need to be modified. It would be preferable to use only one inter-process communications package for all seismic systems.

An additional future benefit is also possible from the exclusive use of an ISIS interface. Future releases of ISIS promise to deliver a network-wide resource allocation package. This package would intelligently select systems for process execution. The ISEIS Monitor processes, along with the Unified Process Initiation package, determine where to execute processes. The anticipated ISIS upgrade should eliminate the need for ISEIS Monitor processes, thus simplifying ISEIS. System load balancing and initiation could all be handled by ISIS. However, this upgrade is not available now.

One problem does remain which complicates ISEIS/ISIS conversion. If ISEIS were to use ISIS exclusively, the source code for ISIS would have to be ported to the Stardent. The Stardent version of ISIS would have to be thoroughly tested and licensing agreements would have to be worked out with Cornell. Assuming that a robust version of ISIS can be installed on the Stardent in a timely fashion, total conversion to ISIS for interprocess communication would be preferable.

F.3.0 IMPLEMENTATION OF SHARED LIBRARIES

The use of shared libraries would greatly reduce the amount of disk space required for the ISEIS system. All processes should bind the X11, X toolkit, ENSCO libraries, and SQL libraries as shared to save space, if possible. The ISEIS discriminant processes (*.DTA, *.EXE, *.UPD, *.mbid, *.cbid) processes, in particular, use a tremendous amount of space. Currently, each discriminant uses from seven to 10 megabytes of storage. This breaks down to about two megabytes for each discriminant process. If many more discriminants are added in the future, disk space could be a real problem.

F.4.0 IMPLEMENTATION OF THE IMAKE UTILITY

As the size of the ISEIS system increased, maintenance of Makefiles became increasingly time-consuming. Usually if one Makefile needed modification, most of the others needed to be modified as well. The practice of "cut and paste" can result in the rapid proliferation of undesirable Makefiles. Makefiles which are deemed undesirable are those which reference absolute UNIX path names and library names. Upon installation at another site, most Makefiles will probably need modification. The current set of ISEIS Makefiles reference environment variables specify path names and library names. The ISEIS system can thus, be compiled and linked at any site by simply changing the environmental variables in the ISEIS .cshrc file.

However, the IMAKE utility provides a way to automatically generate Makefiles which are tailored for a specific installation site. The developer creates an IMakefile and IMAKE generates a Makefile from it. The IMAKE scheme would reduce the number of environmental variables which are currently defined in the ISEIS shell. ISEIS designers have repeatedly complained about the large number of UNIX environmental variables and it would be preferable to remove as many as possible. Converting ISEIS to use IMakefiles instead of Makefiles is not complicated, but it is tedious and it would probably require one-to-two weeks. The IMAKE utility is packaged with the X window software.

F.5.0 MISCELLANEOUS

If the Interactive Map remains as a permanent part of the system, then it should be improved as well. The map widget does not always respond to expose events, which means that sometimes holes remain after a window (which had previously obscured the map) was moved or lowered. This also explains why the event symbol widgets have to be popped-down rather than moved off of the screen when the map is changed. We previously tried to move them off, but holes would sometimes be created. A programmer can specify an expose callback procedure for the HP widget where the map resides. This is currently not done. The installed expose procedure should probably copy the portion of the image which was exposed to the widget display. It is unclear why this wouldn't always have to be done, but the map display does seem to refresh itself when events are popped-down.

A fully scrollable map would be preferable to the new map loading scheme which is now used. However, scrolling would not be possible with the set of current maps since they are azimuthal projections. One large Mercator map would have to be created and placed into a viewport to enable smooth scrolling at a given zoom level. Such a modification would not be trivial however, and would involve some redesign of the map process.

Another problem is that signal processing designers used fixed two-dimensional arrays throughout their code. Such structures make assumptions about the maximum number of channels, stations, phases, etc. There is a multi-dimensional dynamic array allocation routine in one of the system utility libraries and it would be best if their code was changed to dynamically allocate arrays, rather than use fixed static arrays. This will cause nothing but trouble in the future.

Prof. Thomas Ahrens
Seismological Lab, 252-21
Division of Geological & Planetary Sciences
California Institute of Technology
Pasadena, CA 91125

Prof. Keiiti Aki
Center for Earth Sciences
University of Southern California
University Park
Los Angeles, CA 90089-0741

Prof. Shelton Alexander
Geosciences Department
403 Deike Building
The Pennsylvania State University
University Park, PA 16802

Dr. Ralph Alewine, III
DARPA/NMRO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Prof. Charles B. Archambeau
CIRES
University of Colorado
Boulder, CO 80309

Dr. Thomas C. Bache, Jr.
Science Applications Int'l Corp.
10260 Campus Point Drive
San Diego, CA 92121 (2 copies)

Prof. Muawia Barazangi
Institute for the Study of the Continent
Cornell University
Ithaca, NY 14853

Dr. Jeff Barker
Department of Geological Sciences
State University of New York
at Binghamton
Vestal, NY 13901

Dr. Douglas R. Baumgardt
ENSCO, Inc
5400 Port Royal Road
Springfield, VA 22151-2388

Dr. Susan Beck
Department of Geosciences
Building #77
University of Arizona
Tuscon, AZ 85721

Dr. T.J. Bennett
S-CUBED
A Division of Maxwell Laboratories
11800 Sunrise Valley Drive, Suite 1450
Reston, VA 22091

Dr. Robert Blandford
AFTAC/IT, Center for Seismic Studies
1330 North 17th Street
Suite 1450
Arlington, VA 22209-2308

Dr. G.A. Bollinger
Department of Geological Sciences
Virginia Polytechnical Institute
21044 Derring Hall
Blacksburg, VA 24061

Dr. Stephen Bratt
Center for Seismic Studies
1300 North 17th Street
Suite 1450
Arlington, VA 22209-2308

Dr. Lawrence Burdick
Woodward-Clyde Consultants
566 El Dorado Street
Pasadena, CA 91109-3245

Dr. Robert Burridge
Schlumberger-Doll Research Center
Old Quarry Road
Ridgefield, CT 06877

Dr. Jerry Carter
Center for Seismic Studies
1300 North 17th Street
Suite 1450
Arlington, VA 22209-2308

Dr. Eric Chael
Division 9241
Sandia Laboratory
Albuquerque, NM 87185

Prof. Vernon F. Cormier
Department of Geology & Geophysics
U-45, Room 207
University of Connecticut
Storrs, CT 06268

Prof. Anton Dainty
Earth Resources Laboratory
Massachusetts Institute of Technology
42 Carleton Street
Cambridge, MA 02142

Prof. Steven Day
Department of Geological Sciences
San Diego State University
San Diego, CA 92182

Marvin Denny
U.S. Department of Energy
Office of Arms Control
Washington, DC 20585

Dr. Zoltan Der
ENSCO, Inc.
5400 Port Royal Road
Springfield, VA 22151-2388

Prof. Adam Dziewonski
Hoffman Laboratory, Harvard University
Dept. of Earth Atmos. & Planetary Sciences
20 Oxford Street
Cambridge, MA 02138

Prof. John Ebel
Department of Geology & Geophysics
Boston College
Chestnut Hill, MA 02167

Eric Fielding
SNEE Hall
INSTOC
Cornell University
Ithaca, NY 14853

Dr. Mark D. Fisk
Mission Research Corporation
735 State Street
P.O. Drawer 719
Santa Barbara, CA 93102

Prof Stanley Flatte
Applied Sciences Building
University of California, Santa Cruz
Santa Cruz, CA95064

Dr. John Foley
NER-Geo Sciences
1100 Crown Colony Drive
Quincy, MA 02169

Prof. Donald Forsyth
Department of Geological Sciences
Brown University
Providence, RI 02912

Dr. Art Frankel
U.S. Geological Survey
922 National Center
Reston, VA 22092

Dr. Cliff Frolich
Institute of Geophysics
8701 North Mopac
Austin, TX 78759

Dr. Holly Given
IGPP, A-025
Scripps Institute of Oceanography
University of California, San Diego
La Jolla, CA 92093

Dr. Jeffrey W. Given
SAIC
10260 Campus Point Drive
San Diego, CA 92121

Dr. Dale Glover
Defense Intelligence Agency
ATTN: ODT-1B
Washington, DC 20301

Dr. Indra Gupta
Teledyne Geotech
314 Montgomery Street
Alexanderia, VA 22314

Dan N. Hagedon
Pacific Northwest Laboratories
Battelle Boulevard
Richland, WA 99352

Dr. James Hannon
Lawrence Livermore National Laboratory
P.O. Box 808
L-205
Livermore, CA 94550

Dr. Roger Hansen
HQ AFTAC/TTR
Patrick AFB, FL 32925-6001

Prof. David G. Harkrider
Seismological Laboratory
Division of Geological & Planetary Sciences
California Institute of Technology
Pasadena, CA 91125

Prof. Danny Harvey
CIRES
University of Colorado
Boulder, CO 80309

Prof. Donald V. Helmberger
Seismological Laboratory
Division of Geological & Planetary Sciences
California Institute of Technology
Pasadena, CA 91125

Prof. Eugene Herrin
Institute for the Study of Earth and Man
Geophysical Laboratory
Southern Methodist University
Dallas, TX 75275

Prof. Robert B. Herrmann
Department of Earth & Atmospheric Sciences
St. Louis University
St. Louis, MO 63156

Prof. Lane R. Johnson
Seismographic Station
University of California
Berkeley, CA 94720

Prof. Thomas H. Jordan
Department of Earth, Atmospheric &
Planetary Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139

Prof. Alan Kafka
Department of Geology & Geophysics
Boston College
Chestnut Hill, MA 02167

Robert C. Kemerait
ENSCO, Inc.
445 Pineda Court
Melbourne , FL 32940

Dr. Max Koontz
U.S. Dept. of Energy/DP 5
Forrestal Building
1000 Independence Avenue
Washington, DC 20585

Dr. Richard LaCoss
MIT Lincoln Laboratory, M-200B
P.O. Box 73
Lexington, MA 02173-0073

Dr. Fred K. Lamb
University of Illinois at Urbana-Champaign
Department of Physics
1110 West Green Street
Urbana, IL 61801

Prof. Charles A. Langston
Geosciences Department
403 Deike Building
The Pennsylvania State University
University Park, PA 16802

Jim Lawson, Chief Geophysicist
Oklahoma Geological Survey
Oklahoma Geophysical Obseervatory
P.O. Box 8
Leonard, OK 74043-0008

Prof. Thorne Lay
Institute of Tectonics
Earth Science Board
University of California, Santa Cruz
Santa Cruz, CA 95064

Dr. William Leith
U.S. Geological Survey
Mail Stop 928
Reston, VA 22092

Mr. James F. Lewkowicz
Phillips Laboratory/GPEH
Hanscom AFB, MA 01731-5000(2 copies)

Mr. Alfred Lieberman
ACDA/VI-OA State Department Building
Room 5726
320-21st Street, NW
Washington, DC 20451

Prof. L. Timothy Long
School of Geophysical Sciences
Georgia Institute of Technology
Atlanta, GA 30332

Dr. Robert Masse
Denver Federal Building
Bos 25046, Mail Stop 967
Denver, CO 80225

Dr. Randolph Martin, III
New England Research, Inc.
76 Olcott Drive
White River Junction, VT 05001

Dr. Gary McCartor
Department of Physics
Southern Methodist University
Dallas, TX 75275

Prof. Thomas V. McEvilly
Seismographic Station
University of California
Berkeley, CA 94720

Dr. Art McGarr
U.S. Geological Survey
Mail Stop 977
U.S. Geological Survey
Menlo Park, CA 94025

Dr. Keith L. McLaughlin
S-CUBED
A Division of Maxwell Laboratory
P.O. Box 1620
La Jolla, CA 92038-1620

Stephen Miller & Dr. Alexander Florence
SRI International
333 Ravenswood Avenue
Box AF 116
Menlo Park, CA 94025-3493

Prof. Bernard Minster
IGPP, A-025
Scripps Institute of Oceanography
University of California, San Diego
La Jolla, CA 92093

Prof. Brian J. Mitchell
Department of Earth & Atmospheric Sciences
St. Louis University
St. Louis, MO 63156

Mr. Jack Murphy
S-CUBED
A Division of Maxwell Laboratory
11800 Sunrise Valley Drive, Suite 1212
Reston, VA 22091 (2 Copies)

Dr. Keith K. Nakanishi
Lawrence Livermore National Laboratory
L-025
P.O. Box 808
Livermore, CA 94550

Dr. Carl Newton
Los Alamos National Laboratory
P.O. Box 1663
Mail Stop C335, Group ESS-3
Los Alamos, NM 87545

Dr. Bao Nguyen
HQ AFTAC/TTR
Patrick AFB, FL 32925

Prof. John A. Orcutt
IGPP, A-025
Scripps Institute of Oceanography
University of California, San Diego
La Jolla, CA 92093

Prof. Jeffrey Park
Kline Geology Laboratory
P.O. Box 6666
New Haven, CT 06511-8130

Dr. Howard Patton
Lawrence Livermore National Laboratory
L-025
P.O. Box 808
Livermore, CA 94550

Dr. Frank Pilotte
HQ AFTAC/TT
Patrick AFB, FL 32925-6001

Dr. Jay J. Pulli
Radix Systems, Inc.
2 Taft Court, Suite 203
Rockville, MD 20850

Dr. Robert Reinke
ATTN: FCTVTD
Field Command
Defense Nuclear Agency
Kirtland AFB, NM 87115

Prof. Paul G. Richards
Lamont-Doherty Geological Observatory
of Columbia University
Palisades, NY 10964

Mr. Wilmer Rivers
Teledyne Geotech
314 Montgomery Street
Alexandria, VA 22314

Dr. George Rothe
HQ AFTAC/TTR
Patrick AFB, FL 32925-6001

Dr. Alan S. Ryall, Jr.
DARPA/NMRO
3701 North Fairfax Drive
Arlington, VA 22209-1714

Dr. Richard Sailor
TASC, Inc.
55 Walkers Brook Drive
Reading, MA 01867

Prof. Charles G. Sammis
Center for Earth Sciences
University of Southern California
University Park
Los Angeles, CA 90089-0741

Prof. Christopher H. Scholz
Lamont-Doherty Geological Observatory
of Columbia University
Palisades, CA 10964

Dr. Susan Schwartz
Institute of Tectonics
1156 High Street
Santa Cruz, CA 95064

Secretary of the Air Force
(SAFRD)
Washington, DC 20330

Office of the Secretary of Defense
DDR&E
Washington, DC 20330

Thomas J. Sereno, Jr.
Science Application Int'l Corp.
10260 Campus Point Drive
San Diego, CA 92121

Dr. Michael Shore
Defense Nuclear Agency/SPSS
6801 Telegraph Road
Alexandria, VA 22310

Dr. Matthew Sibol
Virginia Tech
Seismological Observatory
4044 Derring Hall
Blacksburg, VA 24061-0420

Prof. David G. Simpson
IRIS, Inc.
1616 North Fort Myer Drive
Suite 1400
Arlington, VA 22209

Donald L. Springer
Lawrence Livermore National Laboratory
L-025
P.O. Box 808
Livermore, CA 94550

Dr. Jeffrey Stevens
S-CUBED
A Division of Maxwell Laboratory
P.O. Box 1620
La Jolla, CA 92038-1620

Lt. Col. Jim Stobie
ATTN: AFOSR/NL
Bolling AFB
Washington, DC 20332-6448

Prof. Brian Stump
Institute for the Study of Earth & Man
Geophysical Laboratory
Southern Methodist University
Dallas, TX 75275

Prof. Jeremiah Sullivan
University of Illinois at Urbana-Champaign
Department of Physics
1110 West Green Street
Urbana, IL 61801

Prof. L. Sykes
Lamont-Doherty Geological Observatory
of Columbia University
Palisades, NY 10964

Dr. David Taylor
ENSCO, Inc.
445 Pineda Court
Melbourne, FL 32940

Dr. Steven R. Taylor
Los Alamos National Laboratory
P.O. Box 1663
Mail Stop C335
Los Alamos, NM 87545

Prof. Clifford Thurber
University of Wisconsin-Madison
Department of Geology & Geophysics
1215 West Dayton Street
Madison, WS 53706

Prof. M. Nafi Toksoz
Earth Resources Lab
Massachusetts Institute of Technology
42 Carleton Street
Cambridge, MA 02142

Dr. Larry Turnbull
CIA-OSWR/NED
Washington, DC 20505

Dr. Gregory van der Vink
IRIS, Inc.
16116 North Fort Myer Drive
Suite 1440
Arlington, VA 22209

Dr. Karl Veith
EG&G
5211 Auth Road
Suite 240
Suitland, MD 20746

Prof. Terry C. Wallace
Department of Geosciences
Building #77
University of Arizona
Tuscon, AZ 85721

Dr. Thomas Weaver
Los Alamos National Laboratory
P.O. Box 1663
Mail Stop C335
Los Alamos, NM 87545

Dr. William Wortman
Mission Research Corporation
8560 Cinderbed Road
Suite 700
Newington, VA 22122

Prof. Francis T. Wu
Department of Geological Sciences
State University of New York
at Binghamton
Vestal, NY 13901

AFTAC/CA
(STINFO)
Patrick AFB, FL 32925-6001

DARPA/PM
3701 North Fairfax Drive
Arlington, VA 22203-1714

DARPA/RMO/RETRIEVAL
3701 North Fairfax Drive
Arlington, VA 22203-1714

DARPA/RMO/SECURITY OFFICE
3701 North Fairfax Drive
Arlington, VA 2203-1714

HQ DNA
ATTN: Technical Library
Washington, DC 20305

Defense Intelligence Agency
Directorate for Scientific & Technical Intelligence
ATTN: DTIB
Washington, DC 20340-6158

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314 (2 Copies)

TACTEC
Battelle Memorial Institute
505 King Avenue
Columbus, OH 43201 (Final Report)

Phillips Laboratory
ATTN: XPG
Hanscom AFB, MA 01731-5000

Phillips Laboratory
ATTN: GPE
Hanscom AFB, MA 01731-5000

Phillips Laboratory
ATTN: TSML
Hanscom AFB, MA 01731-5000

Phillips Laboratory
ATTN: SUL
Kirtland, NM 87117 (2 copies)

Dr. Michel Bouchon
I.R.I.G.M.-B.P. 68
38402 St. Martin D'Heres
Cedex, FRANCE

Dr. Michel Campillo
Observatoire de Grenoble
I.R.I.G.M.-B.P. 53
38041 Grenoble, FRANCE

Dr. Kin Yip Chun
Geophysics Division
Physics Department
University of Toronto
Ontario, CANADA

Prof. Hans-Peter Harjes
Institute for Geophysic
Ruhr University/Bochum
P.O. Box 102148
4630 Bochum 1, GERMANY

Prof. Eystein Husebye
NTNF/NORSAR
P.O. Box 51
N-2007 Kjeller, NORWAY

David Jepsen
Acting Head, Nuclear Monitoring Section
Bureau of Mineral Resources
Geology and Geophysics
G.P.O. Box 378, Canberra, AUSTRALIA

Ms. Eva Johannisson
Senior Research Officer
National Defense Research Inst.
P.O. Box 27322
S-102 54 Stockholm, SWEDEN

Dr. Peter Marshall
Procurement Executive
Ministry of Defense
Blacknest, Brimpton
Reading FG7-FRS, UNITED KINGDOM

Dr. Bernard Massinon, Dr. Pierre Mechler
Societe Radiomana
27 rue Claude Bernard
75005 Paris, FRANCE (2 Copies)

Dr. Svein Mykkeltveit
NTNT/NORSAR
P.O. Box 51
N-2007 Kjeller, NORWAY (3 Copies)

Prof. Keith Priestley
University of Cambridge
Bullard Labs, Dept. of Earth Sciences
Madingley Rise, Madingley Road
Cambridge CB3 OEZ, ENGLAND

Dr. Jorg Schlittenhardt
Federal Institute for Geosciences & Nat'l Res.
Postfach 510153
D-3000 Hannover 51, GERMANY

Dr. Johannes Schweitzer
Institute of Geophysics
Ruhr University/Bochum
P.O. Box 1102148
4360 Bochum 1, GERMANY

Best Available Copy