LA-UR-98-1264
C.3

DOCUMENTATION OF PIDAS CODE VERSION 2.0 FOR TRACKING AND TRENDING ALARMS

```

class CPidasDoc : public CDocument
{
protected: // create from serialization only
    CPidasDoc();
    DECLARE_DYNCREATE(CPidasDoc)

// Attributes
public:

// darby start
public:
    CAlarmList m_theAlarmList; // an alarm list
    CAlarmList m_theSectorList; // a sector list
    // with point ignored
    CStartDlg sDlg; // the start dialog
    CListThere bDlg; // dialog to check for existing list
    CRawDlg fileDlg; // dialog to get the raw data file
    void ProcessAnInputFile(CString);
    CTime m_earliestHistoryEndTime; // if a batch's end time exceeds this time that
    // batch is included in the trend, if not that batch is not included in the trend
    // this is set by the user in OnTrendall()
    CTime m_HistoryStartTime; // the start time for the earliest batch in the history, determined
    // in OutputAlarmHistory() and in OutputSectorHistory() in OntrendAll()
    CString m_theIA55SpecificHistory; // used in ProcessAnInputFile()
    CString m_theIA18SpecificHistory; // used in ProcessAnInputFile()
    BOOL bSpecificHistoryCreated; // set to TRUE in ProcessAnInputFile and used in OnSaveDocument();
    // initialized to FALSE in CPidasDoc() and in OnNewDocument()

// NEW for trending
    CHistAlarmList m_historicalList[2]; // historical list of alarms
    // element 0 is for IA55 facility element 1 is for IA18 facility
    CHistSectorList m_histSectorList[2]; // historical list of sectors
    // element 0 is for IA55 facility element 1 is for IA18 facility

```

John Darby and Dean Sanzo

Probabilistic Risk and Hazard Analysis Group
Technology and Safety Assessment Division
Los Alamos National Laboratory

MARCH 1998

LOS ALAMOS NATIONAL LABORATORY
3 9338 00418 3652

JUN 25 1998

SCANNED

CONTENTS

	Page
LIST OF FIGURES.....	ii
LIST OF TABLES.....	iii
1.0. INTRODUCTION	1
1.1. Reason for Development.....	1
1.2. Design Requirements for the Code.....	1
1.3. Design Decisions for the Code	1
1.4. Framework Decisions for the Code	1
2.0. USER'S MANUAL.....	2
2.1. Overview of Analysis	2
2.1.1. Preparation.....	2
2.1.2. Processing.....	3
2.1.3. Evaluation.....	3
2.1.4. Trending.....	3
2.1.5. Saving the Data.....	3
2.1.6. Summary.....	4
2.2. Typical Session	4
2.2.1. Processing.....	4
2.2.2. Evaluation.....	8
2.2.3. Trending.....	10
2.2.4. Saving the Data.....	14
2.2.5. Summary of a Typical Session.....	15
2.3. Output Files for Viewing and Printing	15
2.3.1. Viewing and Printing Evaluation Results.....	17
2.3.2. Viewing and Printing Trending Results.....	18
2.3.3. Viewing and Printing Graphical Plots.....	23
2.3.4. Viewing and Printing for a Typical Session.....	27
2.4. Optional Features	27
2.4.1. Opening an Existing *.lst File	27
2.4.2. Errors in a *.raw File.....	27
2.4.3. Options For Evaluation and Trending.....	30
2.4.4. Display Options.....	33
2.4.5. Plot Options.....	35
2.4.6. Displaying and Modifying the Master List of Sectors and Alarm Points.....	36
2.4.7. Backing Up Files and Deleting Files.....	39
3.0. CODE DATA STRUCTURES	40
3.1. Data Structures for Tracking Alarms.....	41
3.2. Data Structures for Tracking Sectors.....	41
3.3. Data Structures for Trending Alarms	41
3.4. Data Structures for Trending Sectors	42
3.5. Serialization	44
4.0. RESULTS OF TESTING	45
4.1. Errors in the *.raw Files.....	45
4.2. Unanticipated Fatal Runtime Errors.....	45
5.0. SOURCE CODE	45
REFERENCE.....	45

LIST OF FIGURES

	Page
Fig. 2-1. About pidas beta window.....	2
Fig. 2-2. Summary of the five steps.....	4
Fig. 2-3. PIDAS window after execution.....	5
Fig. 2-4. Main window for PIDAS code.....	6
Fig. 2-5. Dialog for selecting *.raw file to process.....	7
Fig. 2-6. Dialog for changing master list of sectors and alarm points.....	8
Fig. 2-7. Dialog shown after processing is complete.....	9
Fig. 2-8. Dialog for 'Evaluate by alarm and/or sector'.....	10
Fig. 2-9. Dialog for 'Evaluate by cause'.....	11
Fig. 2-10. Dialog for 'Evaluation complete'.....	11
Fig. 2-11. Trend by alarm and/of sector dialog.....	12
Fig. 2-12. Trending complete dialog.....	13
Fig. 2-13. File operations on the *.lst file.....	15
Fig. 2-14. Viewing results.....	16
Fig. 2-15. Current list dialog.....	17
Fig. 2-16. Cause type dialog for current list.....	18
Fig. 2-17. Example portion of a view of a *.dat file (dummy data).....	19
Fig. 2-18. History dialog.....	20
Fig. 2-19. Cause type dialog for history.....	21
Fig. 2-20. Example portion of a view of a *.his file (dummy data).....	22
Fig. 2-21. Plot dialog.....	23
Fig. 2-22. Plot choice dialog.....	24
Fig. 2-23. Plot file description dialog.....	25
Fig. 2-24. Example view of a *.plt file (1 of 3 pages, dummy data).....	26
Fig. 2-25. Pidas main window with *.lst file loaded.....	28
Fig. 2-26. Notification of errors in the *.raw file.....	29
Fig. 2-27. Old dialog.....	31
Fig. 2-28. error.dat file in PIDAS main window.....	32
Fig. 2-29. Processing dialog window with bad records removed.....	33
Fig. 2-30. Files from evaluation by sector only for all causes.....	34
Fig. 2-31. Dialog window for 'Ancient'.....	35
Fig. 2-32. Master sector list dialog.....	36
Fig. 2-33. List of sectors for TA-18.....	37
Fig. 2-34. Dialog for missing sectors.....	39
Fig. 2-35. Dialog for more points hit than in sector list.....	39
Fig. 2-36. ...pidas beta subdirectories.....	40
Fig. 3-1. Data structures for trending.....	43

LIST OF TABLES

	Page
TABLE 3-1 Data Requirements for Trending	42

DOCUMENTATION OF PIDAS CODE VERSION 2.0 FOR TRACKING AND TRENDING ALARMS

1.0. INTRODUCTION

A computer code was developed for tracking and trending alarms from the Perimeter Intrusion Detection and Assessment System (PIDAS) portion of the security system at Los Alamos National Laboratory. The code is named PIDAS to reflect its use for exterior alarms.

'Tracking' and 'Trending' have special connotations in the code. 'Tracking' refers to the analysis of one batch of data over one specific time period, that is, the analysis of one set of raw input data. 'Trending' refers to the analysis that considers many batches of data, that is, the historical analysis of data from many sets of raw input data.

1.1. Reason for Development

The code was developed to fulfill a commitment to the Department of Energy (DOE) to track and trend false and nuisance alarms. The PIDAS code fulfills this commitment for exterior alarms; current plans call for the development of a similar code to track and trend interior alarms.

Ogden Environmental and Energy Services Company, Inc., developed a tracking and trending computer code for exterior alarms in 1994 (Ogden Exterior Code). However, it does not meet current Los Alamos needs; for example, it handles exterior alarms by sectors and not by individual alarms, and the Laboratory needs to evaluate individual alarms.

1.2. Design Requirements for the Code

We imposed a number of requirements on the code.

- The code must track and trend by both alarms and sectors (a sector is a collection of alarms).
- The code must handle data from two facilities: TA-55 and TA-18.
- The code must handle the format of the raw data input files seamlessly.
- The code must handle large input files (many megabytes in size).
- The code must track and trend by false alarm rate (FAR) and by nuisance alarm rate (NAR).

1.3. Design Decisions for the Code

A number of design decisions were made and incorporated into the code.

- The code should track by all causes, not just by FAR and NAR.
- The code should handle input data over an unspecified, variable time duration.
- The trending capability in the code should be 'invisible' to the user of the code.
- The code should provide a hierarchy of output data from detailed to summary.
- The top summary-level output data should be graphical in form.
- The textual output data should be in ASCII (text) format to facilitate use in standard word processing and spreadsheet programs.
- The code should flag and record format errors in the raw data input files.

1.4. Framework Decisions for the Code

Based on the factors listed in Secs. 1.2 and 1.3, we decided to write the code in Visual C++ 5.0. Visual C++ encapsulates the Windows Application Programming Interface (API) into the Microsoft Foundation Classes (MFC), thereby greatly reducing the time required to write object-oriented code under the Windows operating system. Object-oriented code is the natural choice for quickly modeling complex applications because the use of objects enables the code to model reality in a clear, encapsulated manner. The Visual C++ framework works under both Windows 95 and Windows NT 4.0, and C++, and its predecessor C, were designed to handle large applications by using separate compilation of the modules within a program.

2.0. USER'S MANUAL

This discussion is current for Version 2.0 of the PIDAS code, which is dated January 21, 1998. Version 2.0 differs from Version 1.0 in the following respects.

- The earlier version did not highlight sectors in which alarms actually occurred (in the *.raw input files) that were not in the master list of sectors used by the code to calculate the average number of alarms per sector. The new version automatically lists such sectors and allows for revisions to the master list of sectors to update the master list. Similarly, the new version highlights those sectors for which more alarm points actually occurred (in the *.raw data files) than indicated in the master list of sectors; the master list can be updated to reflect the correct number of alarm points in these sectors. This new capability is not part of alarm tracking and trending *per se*, rather it provides quality assurance for the master list of sectors maintained by FSS.
- The earlier version automatically trended over all batches within 1 yr back from the last chronological batch. The new version allows the user to select the duration for the trend up to a 1-yr duration. This allows trending to be done on a variable time scale so that trending can 'start' at a specified time, which is useful if major changes are made to the alarm system in the field that make prior data invalid.
- Numerous minor changes were made to improve the trapping of errors in the input data files and the stability of the code.

Version 2.0 is identical to Version 1.0 in all other respects and completely replaces Version 1.0. The current version of the code can be displayed in the 'About pidas beta' submenu item of the 'Help' menu item in the PIDAS main window, as shown in Fig. 2-1.

2.1. Overview of Analysis

In application, the PIDAS code uses five steps.

1. Preparation
2. Processing
3. Evaluation
4. Trending
5. Saving the Data

In the following discussion, a 'batch' of data refers to one file from FSS that contains alarm data for one facility (either TA-18 or TA-55) over a specified time period. A 'history' refers to a collection of many discrete batches of data for a particular facility. 'Evaluation' refers to the analysis of one batch of data; 'trending' refers to the analysis of a history of data (many batches of data). Within the code, tracking is performed by the Evaluation step and trending is performed by the Trending step.

2.1.1. Preparation. 'Preparation' is the copying of a file of raw data into the appropriate subdirectory of the code. The raw data file must be in ASCII format, and it must conform to the format used by FSS for the raw data files as of December 23, 1997. If the format of the raw data files is changed, the code cannot accurately process the data in step 2.

It is highly recommended that the FSS raw data files be checked for spurious characters in the last record (line) in the file. Many of the raw data files, especially for TA-55, have a spurious last record, and this must be deleted for the code to process the file properly. The code expects every line in the raw data file to begin with a date, and a spurious line at the end of the file will cause the code to not function properly because it tries to read spurious

Fig. 2-1. About pidas beta window.

characters as a date. A spurious last record in the file should be deleted completely using either a word processor or the Visual C++ Integrated Development Environment (IDE) to edit the text file.

To be found by the program, the raw data files must have a '.raw' file extension, and these files must reside in the input subdirectory of the pidas beta directory (...\\pidas beta\\input). The name of the file indicates the facility to which the data applies (TA-18 or TA-55). For example, L1870115 refers to TA-18 and L5570115 refers to TA-55. The L stands for 'log'. This file name format is that used by FSS to label the text file of raw data downloaded from the security system data banks. The numbers after the facility number in the file name refer to the date of the file; for example, L1870115 indicates that the file contains data ending on January 15, 1997. The files as received from FSS do not have a '.raw' extension; they have a '.txt' extension. For use in the code, the text files have to be copied into the input subdirectory and renamed with a .raw extension; this can be done using either a word processor or the Visual C++ IDE. For example, the file L1870115.txt as received from FSS must be saved as a text file with the name L1870115.raw in the input subdirectory.

2.1.2 Processing. 'Processing' is the conversion of raw input data into data structures for further analysis. When the Process portion of the code is activated, a list of raw data files is provided from which one is selected for processing. After the raw data file is selected for processing, data from that file are read in, analyzed, and saved in data structures inside the code appropriate for subsequent Evaluation and Trending. All of the actual operations on the input data are performed in the Processing step.

After the raw data are processed, the Evaluation and Trending analyses should be performed. Conceptually, it is important to know that Evaluation and Trending do not perform any operations on the actual input data; these steps interrogate the information that was created in the Processing step and provide various outputs of this information organized as specified by the user.

2.1.3 Evaluation. 'Evaluation' assesses the alarm data for the current batch. After the Evaluation portion of the code is activated, text output files are produced that track the alarm data by cause. For example, a file is produced that lists all alarms ranked in descending order by FAR, another file is produced that lists all sectors ranked in descending order by average NAR, and so on.

All of the files produced as a result of an Evaluation are ASCII files with the file extension '.dat'; these files are saved in the output subdirectory of the pidas beta directory (...\\pidas beta\\output).

2.1.4 Trending. 'Trending' assesses the history of all the data that have been processed. In practice, trending 'adds' the current batch of data to all the prior batches of data and calculates time-weighted averages over all the batches. Trending provides a 'running' average of the alarm data. After the Trend portion of the code is activated, text output files are produced that sort the alarm data by average historical FAR and NAR. For example, a file is produced that lists all the alarms ranked in descending order by average historical FAR, including a list of the FAR for each batch in the history for each alarm.

Trending produces a set of ASCII files with the file extension '.his' that are saved in the output subdirectory of the pidas beta directory (...\\pidas beta\\output). Trending also produces text files with the extension '.plt' in the output subdirectory. These files are used by the code to generate output graphical plots. Trending also produces binary files with the extension '.cfile'. These files are used to save the historical information for future trending by the code. They are used internally by the code and are not of primary interest to the user.

2.1.5 Saving the Data. To allow for future analysis or a review of the results of a prior analysis of a batch of data, the user should save the results of processing/evaluating/trending the raw data in a file with an extension '.lst'. The code automatically prompts the user to do this before exiting the program. It is highly recommended that the name of the .lst file correspond to the name of its associated .raw file; for example, the results of processing the L1870115.raw file should be saved as L1870115.lst. (The .lst file is actually the Visual C++ document's data serialized to disk. The extension .lst was chosen because the document's data structures consist of many dynamic doubly linked lists.)

Conceptually, the .lst file can be considered as a .raw data file that the code has processed and reconfigured for use by the code. After a *.raw file has been Processed and saved as a *.lst file, Evaluation and Trending can be performed on the *.lst file without reprocessing the associated *.raw file.

Fig. 2-2. Summary of the five steps.

2.1.6 Summary. Figure 2-2 summarizes the five steps in the analysis (Preparation, Processing, Evaluation, Trending, and Saving the Data). The various input/output files used by the code also are shown in this figure.

Figure 2-2 indicates that *.pts files also are used by the code. These are the files that contain the master list of sectors and alarm points in the sectors used to compute the average FAR and average NAR by sector. The user should never have to use these files directly; the way in which the code uses these files is described in Sec. 2.4.6.

2.2. Typical Session

This section is a walk-through of a typical session with the PIDAS code. The walkthrough is for a standard application of the code; optional features are discussed in Sec. 2.4.

The walkthrough covers four of the five steps discussed in Sec. 2.1, specifically, Processing, Evaluation, Trending, and Saving the Data. The first step, Preparation, is not discussed because it is done from outside the PIDAS code itself, and this step was covered fully in Sec. 2.1.1.

2.2.1 Processing. To run the PIDAS application, the pidas beta.exe file is executed either from the Windows command line using the run command or from within the Visual C++ framework using the Execute command from the IDE Build menu item. The executable file is in the ...\\pidas beta\\debug directory.* When the code is executed, the dialog window shown in Fig. 2-3 appears.

*The Visual C++ framework provides two configurations: Debug and Release. The debug configuration provides on-line debugging support; the release configuration does not provide debugging support but runs faster. The current version of the PIDAS code (Version 2.0) uses the Debug configuration to assist in solving problems during application by FSS should any arise.

Fig. 2-3. PIDAS window after execution.

This modal dialog window states that either of two choices should be made.

- 'Process' a raw data file (*.raw)
- 'Open' an existing processed file (*.lst)

This dialog window is for information only; no code actions are performed after closing this dialog window by clicking 'OK'.

The usual choice is to Process a raw data file into data structures for further analysis. The second choice is to open a *.lst file to view the results of prior analysis; the *.lst file is the result of saving the data from an earlier analysis. This walkthrough discusses the first option, 'Process a .raw data file.' The second option, 'Open a .lst file,' is addressed in Sec. 2.4. The user must click the 'OK' button. When this is done, the code is ready to Process raw data.

After the opening dialog window is closed, the main PIDAS window is ready to receive user-selected commands. The main PIDAS window is shown in Fig. 2-4. Three toolbar buttons are highlighted in this figure: Process Raw Data, Evaluate Results, and Trend. These toolbar buttons cause the code to perform the three steps, Processing, Evaluation, and Trending, described in Sec. 2.1. These three commands also are provided as submenu items in the Analyze menu as shown in Fig. 2-4.

To Process the raw data, the user clicks the process toolbar button (or selects Process from the Analyze menu). A dialog window then is displayed listing all the *.raw files available for processing. These are all files with an extension .raw in the ...\\pidas beta\\input subdirectory. This dialog window is shown in Fig. 2-5. The user clicks on the name of the *.raw file to be processed and then clicks on 'Process File'. (To abort processing, the user clicks on 'DO NOT Process File'.)

When the 'Process File' button is clicked, the code identifies the facility to which the raw data file applies—TA-18 or TA-55—by searching the second and third characters in the *.raw filename that was selected to be processed. A facility identification dialog window is displayed asking the user to confirm that the correct facility has been identified. Assuming that the naming convention for *.raw files discussed in Sec. 2.1.1 has been followed, the user merely confirms that the code has selected the correct facility; if an unusual file name has been selected, the user can force the facility to be either TA-18 or TA-55 by following the directions in the facility identification dialog window.

After the facility has been confirmed, the code allows the user to change the master list of sectors and alarm points. The dialog windows shown in Fig. 2-6 are displayed for this purpose; changing the master list of sectors and alarm points is discussed in Sec. 2.4.6. To accept the existing master list of sectors and alarm points, the user clicks 'Cancel' button.

The code then processes the raw data file. This may take some time as some of the files may be as large as 30 Mbytes. While the raw data file is being processed, the current record (line) number that is being read into the

Fig. 2-4. Main window for PIDAS code.

Fig. 2-5. Dialog for selecting *.raw file to process.

Fig. 2-6. Dialog for changing master list of sectors and alarm points.

PIDAS code from the raw data file is displayed in the main PIDAS dialog window. Approximately 30 min is required to process a 30-Mbyte file (on a 166-MHz Pentium Pro; debug configuration).

After processing is complete, the dialog window shown in Fig. 2-7 is displayed. This window indicates the number of records (lines) in the raw data file that were processed and whether errors were noted in the raw data file. Assuming no errors were detected in the raw data file, the dialog box indicates no errors as shown in Fig. 2-7 and states that the Evaluation step should be performed next.

Errors in the raw data file are discussed in Sec. 2.4. This section should be consulted before further analysis is performed if the dialog window indicates that there are errors in the *.raw file.

The user clicks 'OK' to close the dialog window, indicating that the Processing is complete.

If the data file that was processed contains either (a) sectors that are not in the master list of sectors and alarm points or (b) sectors with more alarm points than are in the master list of sectors and alarm points, additional dialog boxes are displayed showing the missing sectors and/or those sectors with more points than in the list. This is discussed in Sec. 2.4.6.

The code performs no further actions automatically; it is ready to accept further user commands.

2.2.2 Evaluation. To evaluate the data, the user clicks on the 'Evaluate Results' toolbar button (shown in Fig. 2-4) in the PIDAS main window. (Selecting 'Evaluate' in the Analyze submenu has the same effect.) It is important to note that Evaluation provides output for the current batch of data (one batch of the entire history); Trending provides output based on the entire history.

The code then displays the dialog window shown in Fig. 2-8. This window allows to user to specify if the evaluation will be done by alarm, sector, or both. The default option is to evaluate by alarm and sector. This choice determines the type of output *.dat files that will be generated. Evaluating by both alarm and sector will generate files giving results by individual alarm points and individual sectors (a sector is a group of alarm points). Choosing either alarm or sector will generate results only by alarm or sector, respectively. As indicated in this dialog, Evaluation does not perform any operations on the *.raw data file; it takes the Processed *.lst file and produces output *.dat files from the data in the *.lst file. To implement the default evaluation option, the user clicks the 'OK' button.

Fig. 2-7. Dialog shown after processing is complete.

Fig. 2-8. Dialog for 'Evaluate by alarm and/or sector.'

Another dialog window, shown in Fig. 2-9, then is displayed. This dialog window asks the user to specify by which causes the data will be evaluated; the default option is to evaluate by FAR and NAR. Another option allows the data to be evaluated by all causes; this option is discussed in Sec. 2.4. To implement the default option, the user clicks the 'OK' button. In response to this selection, the code evaluates the data and displays a message to that effect while data are being evaluated. When the code has completed the evaluation, the dialog window shown in Fig. 2-10 is shown. This dialog window indicates that the results of the Evaluation can be viewed by selecting Display. Viewing results is discussed in Sec. 2.3. The next step in the analysis process is to perform Trending.

2.2.3. Trending. PIDAS automatically creates an updated history considering all the batches every time a *.raw data file is Processed into a *.lst file; the resulting *.lst file can be trended to provide average results based on the overall history. A more complete description of how the code automatically creates an updated history is provided in Sec. 3.4 of this report.

It is important to note that Trending provides output based on the entire history, and Evaluation provides output for the current batch of data (one batch of the entire history). To perform trending, the user clicks on the Trend toolbar button in the PIDAS beta main window (shown in Fig. 2-4). (Selecting Trend in the Analyze submenu has the same effect.) In response to this selection, the dialog window shown in Fig. 2-11 is displayed. This dialog prompts the user to select whether trending is by alarm, sector, or both alarm and sector. The default is to trend by alarm and sector. The choice determines the type of output *.his and *.plt files that will be generated. Choosing both alarm and sector will generate files giving results by individual alarm points and individual sectors (a sector is a group of alarm points). Choosing either alarm or sector will generate results only by alarm or sector.

This dialog also contains an edit box in which the duration of the trend in days can be specified; the duration can be set from 1 to 365 days. Trending is performed using all batches whose end date is greater than the 'duration start date.' The 'duration start date' is calculated to be that date which is X days back from the end date of the latest batch, where X is the number of days entered into the edit box. The default value for X is 365 (days). The trend will always be performed on at least one hatch of data, the latest batch, because X cannot be set to be less than 1. One reason for setting X to 1 is so that the *.plt files will produce graphical results for one and only one batch of data, specifically the latest batch of data. Trending can be performed as many times as desired with different

Fig. 2-9. Dialog for 'Evaluate by cause.'

Fig. 2-10. Dialog for 'Evaluation complete.'

Fig. 2-11. Trend by alarm and/of sector dialog.

durations; each time, a new set of *.his and *.plt output files are created for the duration specified, but the actual data in the *.lst file is unaffected because Trending merely operates on that data to produce the desired output.

As indicated in this dialog, Trending does not perform any operations on the *.raw data file; Trending takes the processed *.lst file and produces various output *.his and *.plt files from the data in the *.lst file.

To implement the default option, the user clicks the 'OK' button. In response to this selection, the code updates the entire history of all prior batches of data with the current batch of data and calculates running averages for FAR and NAR based on the time-weighted FAR and NAR calculated for each discrete batch of data in the history.

The code displays a message in the PIDAS main window indicating that Trending is in process. When the trending analysis is complete, the dialog window shown in Fig. 2-12 is displayed. This dialog window indicates that the results of the Trending can be viewed by selecting Display. Viewing results is discussed in Sec. 2.2.4.

Three special features are built into Trending. First, the trend can extend back in history any number of days up to 365 (1 yr) from the end date of the last chronological batch in the history. (Batches with end dates earlier than 1 yr back from this date are deleted automatically from the batches considered in the history during the Processing step of the analysis. This prevents the history from becoming too large to analyze efficiently as years of data are collected. The *.lst file always contains all batches within 1 year of the latest batch in the history. Trending uses a

Fig. 2-12. Trending complete dialog.

user-specified duration to interrogate the entire history in the *.lst file and only trend those batches later than the 'duration start date.'

Second, the *.raw data files do not have to be analyzed (Processed/Evaluated/Trended) in chronological order. That is, if a raw data file is missed, it can be analyzed after files with newer dates have been analyzed; the code automatically inserts the batch of data into the proper chronological position in the history. **Although *.raw files can be analyzed out of chronological order, it is highly recommended that this be the rare exception and that files for each facility be processed in chronological order.** There are two main reasons that Processing out of order is not recommended.

- The naming scheme for files produced during an analysis can cause ambiguity for processing out of order. For example, as discussed in Sec. 2.3.2, the *.his files produced as a result of Trending have the 'Start Date' and 'End Date' of the time period that the history covers. The start date is the start date from the earliest batch of data in the history and the end date is the end date from the last batch in the history. If a batch with dates between the last history's start and end dates is trended, the file names of the *.his files are not changed because of the start and end dates of the history have not changed, but the data in the *.his files have changed because of the insertion of an intermediate batch of data in the overall history.
- If the codes are not processed in chronological order, the 'latest batch' is not the 'current batch.' The 'latest batch' is the batch with the latest end date; the 'current batch' is the batch that is being Processed. These two batches are one and the same only if the *.raw data files are Processed in chronological order. If the files are processed out of order, Evaluation provides results for the current batch, and Trending provides results up to the latest batch; this is confusing.

The code allows for processing out of order for two reasons.

1. A *.raw data file may be skipped inadvertently because of human error.
2. It may be required to reprocess an old *.raw data file using a different master list of sectors and alarm points in the sectors.

However, the preferred approach is to process the files in chronological order and only process out of order for either of these two cases.

The code considers each facility (TA-18 and TA-55) separately, so the order of processing *.raw file for **different** facilities is not of concern. That is, as long as *.raw data files for a given facility are processed in chronological order, Processing of files for the different facilities can be performed in any order. For example, consider the following four *.raw data files to be processed: two for TA-18 and two for TA-55.

1. L1870115.raw
2. L1870124.raw
3. L5570329.raw
4. L5570505.raw

The earlier file for TA-18, L1870115.raw, should be processed before the later file for TA-18, L1870124.raw. The earlier file for TA-55, L5570329.raw, should be processed before the later file for TA-55, L5570505.raw. However, the order of processing between facilities is not important; that is, the files can be processed in any of the following sequences and the 'latest batch' will be the 'current batch' for each separate facility.

- 1, 2, 3, 4
- 1, 3, 2, 4
- 3, 1, 2, 4
- 3, 4, 1, 2.

The following Processing sequences will result in the 'latest batch' not being the 'current batch' for the indicated facility.

• 2, 1, 3, 4	TA-18 files processed out of order, TA-55 file processed in order
• 4, 3, 2, 1	TA-18 files processed out of order, TA-55 file processed out of order
• 1, 2, 4, 3	TA-18 files processed in order, TA-55 file processed out of order.

Third, if the same *.raw data file is analyzed more than once, it is not double-counted in the history; the code automatically prevents this from occurring by deleting the existing batch and overwriting it with the new batch. The reason that the 'repeated' batch is overwritten rather than ignored is that the user may be reprocessing intentionally the batch using different data in the master list for sectors and alarm points.

The next step in the analysis process is to Save the Data.

2.2.4. Saving the Data. When the raw data file has been Processed and Evaluation and Trending have been performed, the data structures should be saved. The code uses the Visual C++ framework to save the data in a file with the extension '.lst'. Saving the data in a *.lst file in effect means that the data can be used again at a future date without reprocessing the raw data file. The *.lst file is the *.raw file preprocessed and ready for direct use by the rest of the functions in the code. (If the processed data are saved as a *.lst file without performing Evaluation and/or Trending, this *.lst file can be reopened later and Evaluation and Trending can be performed without reprocessing the associated *.raw file. This is because all the information from the *.raw data file needed for subsequent Evaluation and/or Trending is stored in the *.lst file. If the Processed data are not saved as a *.lst file, Evaluation and Trending cannot be performed later without reprocessing the associated *.raw data file. If the data are to be analyzed using a different set of data in the master list of sectors and alarm points, the *.raw data file must be Processed again with the new master list.)

Figure 2-13 shows the submenu items on the File menu of the PIDAS main window. These are the standard Windows file-related commands common to many Windows programs, except that for the PIDAS code, the file commands relate to a *.lst file whose data are the results of processing a raw input file.

Once a raw data file has been Processed, Evaluated, and Trended, the user should use the Save submenu item in the File menu to save the data structures. As discussed in Sec. 2.1.5, the framework assigns the default file extension '.lst' to these files, and it is highly recommended that the name of the .lst file be the same as the name of the .raw file which it represents (except of course for the different file extension.) The *.lst files are saved in the ...\\pidas beta directory itself, not in a subdirectory of pidas beta.

After the user has saved the data in a .lst file, the New command from the File menu should be selected to create a new document to accept processing of a new raw data file. Then, Processing of the new data file can proceed as discussed in Sec. 2.2.1.

If the user fails to save the data in a .lst file, the framework prompts the user to do so.

The 'Open' command is used to open a previously saved *.lst file; the use of 'Open' is discussed in Sec. 2.4.1.

The 'Print' command is used to print out results that are displayed on the screen; viewing and printing are discussed in Sec. 2.3.

Fig. 2-13. File operations on the *.lst File.

2.2.5 Summary of a Typical Session. A typical working session of the code as discussed in Secs. 2.2.1 through 2.2.4 is proceeds as follows.

- Execute pidas beta.exe in the ..\pidas beta\debug subdirectory.
- Click the **Process Raw Data** toolbar button shown in Fig. 2-3 and select a raw data file to process.
- Click the **Evaluate Results** toolbar button shown in Fig. 2-3 and accept the default choices.
- Click the **Trend** toolbar button as shown in Fig. 2-3 and accept the default choices
- Select **Save** from the File menu and save the data in a *.lst file.

The results of an analysis can be viewed and printed out; Sec. 2.3.4 discusses viewing and printing for a typical working session.

2.3. Output Files for Viewing and Printing

PIDAS produces numerous output files, many of which can be viewed and printed by the user to see the results of an analysis. As a result of Processing/Evaluation/Trending as discussed in Sec. 2.2, the code automatically produces the following ASCII files in the ...\\pidas beta\\output subdirectory that can be viewed and printed.

- Files with the '.dat' extension.
- Files with the '.his' extension.
- Files with the '.plt' extension.

The code also automatically produces files with the '.cfile' extension, but these are binary files used by the code for Trending and are not for viewing or printing. Section 3.5 discusses how the code uses these files.

All of these files (*.dat, *.his, *.plt) are produced whether or not the data structures for the current batch of data are saved in a *.lst file. However, saving the data structures in a .lst file enables the code to find these output files

for a prior analysis by 'Opening' a saved .lst file; this is discussed in Sec. 2.4.1. The *.dat files document the results of an Evaluation of a batch of data. That is, the *.dat files document the tracking of the current batch of data. The *.his and *.plt files document the results of Trending a batch of data.

The *.dat and *.his files are intended to be viewed and printed directly from within the code. They also can be viewed and printed from any other application that accepts text files, such as a word processing program. The *.plt files are used within the code to produce graphical output for viewing and printing. The code does not provide the capability to directly view and print the *.plt files; rather, the code provides the capability to view and print the graphical plots produced within the code using these files. The graphical plots can be viewed and printed only from within the code. The *.plt files, being text files, can be viewed in any application that accepts text files, but these files are merely lines of data used internally by PIDAS for producing graphical output and therefore provide no useful information for documenting results.

Figure 2-14 shows the PIDAS main window with user interface elements for viewing specifically noted. Views can be displayed using the three tool bar buttons.

- Current List
- History
- Select Plot

The Current and History submenu items in the Display menu serve the same purpose as the Current List and History toolbar buttons, respectively. There is no Display submenu item corresponding to the Select a Plot toolbar button. Also, the Display submenu contains two additional items, Old and Ancient, that are discussed in Sec. 2.4.4.

The discussion in the remainder of this section assumes that the Processing, Evaluation, and Trending steps have been completed using the default options as discussed in Sec. 2.2.

Fig. 2-14. Viewing results.

2.3.1. Viewing and Printing Evaluation Results. If the 'Current List' toolbar button is clicked, the dialog window shown in Fig. 2-15 is displayed. This dialog window lists the *.dat files that were produced as a result of the Evaluation step. Each file name has the appropriate 'Facility' that was Processed and the 'Start Date' and 'End Date' of the date from the raw data file that was processed. Also, each file name has 'Cause Type' and 'Alarm/Sector' information indicated. The Cause Type indicates the tracking results in the file; for example, FAR indicates that the file tracks the current batch of data by FAR and NAR indicates that the file tracks the data by NAR. The Alarm/Sector identifier indicates whether the file tracks by alarm points or by sectors.

The 'Dump' designator in a file name indicates that the file dumps the data as they were Processed without ordering by FAR, NAR, or any other cause type.

The set of files displayed in the dialog box for 'Current List' are the tracking results based on the options chosen during Evaluation. As discussed in Sec. 2.2.2, the default options for Evaluation are to track by FAR and NAR by both alarms and sectors. However, many cause types besides FAR and NAR can be tracked; these are discussed in Sec. 2.4.3.

The button titled 'Show More Information on Cause Types in File Names' provides more descriptive information on the acronyms used in the file names for Cause Type. If this button is clicked, the dialog window in Fig. 2-16 is shown and contains the descriptive information.

To view desired results, the user selects a file from the list and clicks the 'Show File' button in the Current List dialog window. For example, to display the tracking results for TA-18 from June 6, 1997, through June 12, 1997, ranked in descending order by FAR for each alarm point, the file 'TA18 6_9_1997 to 6_12_1997 FAR Alarm.dat' should be selected. Or to display the tracking results for TA-18 from June 6, 1997, through June 12, 1997, ranked in descending order by NAR for each sector, the file 'TA18 6_9_1997 to 6_12_1997 NAR Sector.dat' should be selected. PIDAS implements both vertical and horizontal scrolling to view all the data in the file that is displayed on the screen as indicated in Fig. 2-14.

Fig. 2-15. Current list dialog.

Fig. 2-16. Cause type dialog for current list.

Figure 2-17 is an example of how a view of a *.dat file appears on the screen. This example uses dummy data, not data from an actual analysis.

To print desired results when a view is displayed on the screen, the 'Print' item in the 'File' menu as shown in Fig. 2-13 is clicked. For *.dat files, 'Print Preview' in the File submenu merely displays a message indicating that the text on the screen will be printed out.

2.3.2. Viewing and Printing Trending Results. If the 'History' toolbar button is clicked, the dialog window shown in Fig. 2-18 is displayed. This dialog window lists the *.his files that were produced as a result of the Trending step. Each file name has the appropriate 'Facility' that was Processed and the 'Start Date' and 'End Date' of the time period that the history covers. The start date is the start date from the earliest batch of data in the history, and the end date is the date from the last batch in the history. Also, each file name has 'FAR/NAR,' '50/All,' and 'Alarm/Sector' information indicated. The 'FAR/NAR' indicates whether the file contains historical information for FAR or for NAR. The '50/All' indicates whether the file contains information for the top 50 alarms or sectors or for all alarms or sectors for which an alarm for any cause was recorded in any time batch in the history. The 'Alarm/Sector' specifies whether the file results are by alarm point or by sector.

The set of files displayed in the dialog box for 'History' is the trending results for all batches of data with end dates after the 'duration start date' through the latest batch, which is the current batch that was just evaluated assuming the files are processed in chronological order. Trending is always performed by both FAR and NAR, but as discussed in Sec. 2.2.3, it can be performed by alarm, sector, or both alarm and sector. The default option is to trend by both alarm and sector.

Fig. 2-17. Example portion of a view of a *.dat file (dummy data).

Fig. 2-18. History dialog.

For a history, the 'running' average FAR and NAR are computed and used to rank-order results. The running average is the time-weighted average over all batches in the history. The running average FAR for n total batches is computed by the following equation:

$$FAR_{\text{running average}} = \frac{\sum_{i=1}^n (FAR_i * T_i)}{\sum_{i=1}^n T_i}, \quad \text{Equation (1)}$$

where i is one of the n batches in the history, FAR is the false alarm rate for the i^{th} batch, and T is the duration of the i^{th} batch (raw data end time minus raw data start time). The average uses time weighting because the longer the time period for a given batch, the greater the importance of the FAR for that batch. The running average NAR is computed in a similar fashion.

The '50' and the 'All' files contain similar but different information. The files for the top '50' alarms or sectors provide detailed information for the top 50 alarms or sectors, including the FAR and NAR for every alarm or sector for every batch in the history; where a batch FAR or NAR is not provided, the FAR or NAR for that batch was 0.0. (Even if the FAR or NAR was 0.0 for a batch, the batch is still in the history if an alarm occurred for any cause.) The files for 'All' the alarms or sectors provide information for all the alarms or sectors that ever had an alarm for any cause, but the FAR and NAR for each constituent batch in the history are not provided. Thus, the

'50' files contain more detailed information on the top 50 alarms or sectors, whereas the 'All' files contain less detailed information for all the alarms or sectors.

The button titled 'Show More Information on History Types in File Names' provides more descriptive information on the acronyms used in the file names. If this button is clicked, the dialog window shown in Fig. 2-19, which contains the descriptive information, is shown.

To view desired results, the user selects a file from the list and clicks the 'Show File' button in the Current List dialog window. For example, to display the trending results for TA-18 from January 3, 1997, to June 23, 1997, ranked in descending order by average FAR for the top 50 alarm points by alarm point, the file 'TA18 1_3_1997 to 6_23_1997 FAR 50 Alarm.his' should be selected. Or to display the trending results from January 3, 1997, to June 23, 1997, ranked in descending order by average NAR for all alarms by sector, the file 'TA18 1_3_1997 to 6_23_1997 NAR ALL Sector.his' should be selected. PIDAS implements both vertical and horizontal scrolling to view all the data in the file that is displayed on the screen, as indicated in Fig. 2-14.

Figure 2-20 is an example of how a view of a *.his file appears on the screen. This example contains dummy data, not real data from an actual analysis.

Fig. 2-19. Cause type dialog for history.

Fig. 2-20. Example portion of a view of a *.his file (dummy data).

To print desired results when a view is displayed on the screen, the 'Print' item in the 'File' menu (as shown in Fig. 2-13) is clicked. For *.his files, 'Print Preview' in the File submenu merely displays a message indicating that the text on the screen will be printed out.

2.3.3. Viewing and Printing Graphical Plots. If the 'Select Plot' toolbar button is clicked, the dialog window shown in Fig. 2-21 is displayed. This dialog provides two options for the plots: 'Plot Historical Trend' and 'Plot Last Data Batch in the History.' 'Plot Historical Trend' is the default option and enables the actual plot subsequently selected to plot a running average FAR or NAR by alarm or sector, depending on which specific plot is chosen. 'Plot Last Data Batch in the History' is described in Sec. 2.4.5.

Assuming the default option is selected, the dialog in Fig. 2-22 is displayed after the 'Show Plot Choices' button is clicked. If the 'Show More Information on Acronyms in File Names' is clicked, the dialog shown in Fig. 2-23 is displayed, which provides more descriptive information on the acronyms used in the *.plt file names. This dialog lists the specific plots available. Four plots choices are available: FAR by alarm, FAR by sector, NAR by alarm, and NAR by sector. All the plots provide up 50 alarms or sectors rank-ordered by the running average FAR or NAR.

An example view of one of three pages of a plot for alarm points by running average FAR is shown in Fig. 2-24. The plot contains dummy data, not actual data resulting from a true analysis.

Several interesting features are designed into the plot's format. The abscissa has the top 50 alarms or sectors rank-ordered by decreasing average FAR or NAR, so the alarms or sectors with the worst (highest) average FAR or NAR are to the left end of the plot. The ordinate is scaled automatically so that the maximum value on the ordinate is the average FAR or NAR for the worst alarm or sector; all other average FAR and NAR values are scaled accordingly between this maximum value and zero at the plot's origin. The numbers plotted for each alarm or sector are the running average FAR or NAR rounded off to the closest integer; that is, the running average FAR or NAR for

Fig. 2-21. Plot dialog.

Fig. 2-22. Plot choice dialog.

Fig. 2-23. Plot file description dialog.

Fig. 2-24. Example view of a *.plt file (1 of 3 pages, dummy data).

any one of the top 50 alarms or sectors is the number plotted for that alarm or sector. The plot does not plot zero running FAR and NAR, but the alarms or sectors with zero FAR or NAR (if any) are those to the right end of the plot with no numbers plotted. If the history contains fewer than 50 alarms or sectors, the right end of the abscissa has no alarm points or sectors noted past the last one in the historical data.

The 'Print Preview' item in the File menu shown in Fig. 2-13 shows the plot on the screen as it will be printed.

'Print' on the File menu will print the plot; the plot is designed to be printed on 8.5-in. by 11-in. paper in the landscape mode, so the printer should be configured to this page size and orientation. (When the 'Select Plot' toolbar button is clicked, a message is shown stating that the printer should be set to landscape mode.)

2.3.4. Viewing and Printing for a Typical Session. After Processing, Evaluation, and Trending have been performed for a typical session as described in Sec. 2.2, it is recommended that the following results be viewed and printed.

- The 'Current List' toolbar button should be clicked, and the alarm and sector files for FAR and NAR should be viewed and printed.
- The 'History,' toolbar button should be clicked, and the alarm and sector files for FAR and NAR for the top 50 alarms and sectors should be viewed and printed.
- The 'Select Plot' toolbar button should be clicked. The default option for viewing the history, not the last batch of data, should be accepted. All four of the available files should be viewed and printed; these files plot the top 50 alarms and sectors by running average FAR and running average NAR.

Taken together, these printouts provide hard-copy documentation of the analysis (Processing, Evaluation, and Trending) of a batch of data.

2.4. Optional Features

The documentation provided thus far in Sec. 2 has addressed a typical working session in which all default options are accepted. This section addresses other options that may be selected in the code.

2.4.1. Opening an Existing *.lst File. The results of an analysis can and should be saved in a *.lst file as discussed in Sec. 2.2.4. Saving the results in a *.lst file allows the results to be used at a later time without reprocessing a raw data file. The *.lst file is the *.raw file preprocessed and ready for use by the code.

The most common use of a *.lst file is viewing and printing the results of an earlier analysis. To do this, select 'Open' from the File menu (see Fig. 2-13). From the file dialog that is displayed, select the *.lst file to be displayed. The selected file is loaded and the title bar of the PIDAS main window displays the selected file as shown in Fig. 2-25. With the file loaded, viewing and printing of results associated with the prior analysis of the batch of data corresponding to that *.lst file can be performed, if the associated *.dat, *.his, and *.plt files have not been deleted from the ...\\pidas beta\\output subdirectory.

When a *.lst file has been opened, it is possible to re-evaluate and retrend the data, but this typically should not be necessary unless output files have been deleted and cannot be reloaded.. However, it is not possible to reprocess because the *.lst file has already been processed.

2.4.2. Errors in a *.raw File. During code testing, it was noted that some of the *.raw files have 'garbage data' in a few places for the alarm point and sector. There were so few of these occurrences that the statistical effect on the overall results is insignificant. However, it is possible that the garbage data can appear in view and print results because the code reads in alarm points and sectors as strings and will process the garbage data. Therefore, it is recommended that such data be removed from the *.raw data files.

It is not practical to manually search the *.raw files for garbage data as some of the files contain hundreds of thousands of records (lines). PIDAS searches for errors in the *.raw file during Processing and alerts the user to the presence of garbage data. After the Processing step is complete, the code displays a dialog window that shows the number of records processed and whether any errors were detected during Processing. Figure 2-7 shows this dialog for the case in which the *.raw file does not contain garbage data. If errors are encountered, the dialog window that appears after Processing is complete is similar to that shown in Fig. 2-26.

Fig. 2-25. Pidas main window with *.lst file loaded.

Fig. 2-26. Notification of errors in the *.raw file.

If the error.dat file is selected and the 'Show File' button is clicked, the errors trapped during Processing are displayed on the PIDAS main window as shown in Fig. 2-28. The particular results in Fig. 2-28 indicate that the alarm point value is garbage data in records 19,297 and 127,746 in the *.raw data file. When this occurs, it is recommended that further analysis (Evaluation and Trending) not be performed. The user should exit PIDAS without saving the results in a *.lst file. Then the *.raw data file should be opened, either in a word processor or in the Visual C+ IDE, and the records with the garbage data should be stripped out.

04/26/97 18:45:07 901-piÅ Clear Maint SECTOR-PpÅPpiÅPpiÅPpiÅPpiÅPpiÅPp
05/01/97 11:48:06 901-piÅ Alarm Maint SECTOR-PpÅPpiÅPpiÅPpiÅPpiÅPpiÅPp

Record (line) 127,746

Record (line) 19,297

Errors such as those just discussed do not prevent the code from operating properly. One error in the *.raw file that will prevent the code from operating properly is a spurious last record. This error and how to fix it were discussed in Sec. 2.1.1.

After a *.raw data file is Processed and the 'Evaluate Results' toolbar button is clicked (see Fig. 2-4), the dialog window shown in Fig. 2-8 is displayed. This dialog box allows tracking results to be produced by alarm, sector, or both alarm and sector. The default option is alarm and sector, which produces files with both alarm and sector information as indicated in Fig. 2-15 in which both '.... Alarm.dat' and '...Sector.dat' files have been produced as the result of an Evaluation. If the user chooses to evaluate by only alarm or only sector in the dialog window shown on Fig. 2-8, then only '.... Alarm.dat' or '...Sector.dat' files are created during an Evaluation and only these files appear in the dialog window shown in Fig. 2-15.

- All (sum of all the following causes in this list)
- Maintenance (no compensatory measures)
- Maintenance (compensatory measures)

Fig. 2-27. Old dialog.

Fig. 2-28. error.dat file in PIDAS main window.

Fig. 2-29. Processing dialog window with bad records removed.

- Walktest
- Patrol Access
- CAS Closure
- Unknown
- Authorized Intrusion
- Weather
- Wildlife
- Unauthorized Intrusion
- Adjacent Activity

FAR is equivalent to Unknown; NAR is the sum of: Weather, Wildlife, and Adjacent Activity.

If the Evaluation is performed for only sectors ('Evaluate Sectors Only' in the dialog in Fig. 2-8) and for all causes ('Evaluate for All Causes' in Fig. 2-9), then the files produced as a result of the Evaluation when 'Current List' is selected (Fig. 2-14) are displayed in the dialog shown in Fig. 2-30. (This is the same dialog box shown in Fig. 2-15 with different options selected for Evaluation.) As indicated in this dialog window, the evaluation produced files for all causes only by sector.

For Trending, the default option is to trend by both alarm and sector, as indicated in Fig. 2-11. If desired, the trending can be performed by only alarm or only sector. Trending is done for both FAR and NAR but not for the other causes. Therefore, Evaluation can be performed for all causes, but Trending is restricted to FAR and NAR.

2.4.4. Display Options. As indicated in Fig. 2-14, there are two additional items on the Display submenu: 'Old' and 'Ancient'. There are no corresponding toolbar buttons for these items.

The 'Old' item displays the dialog box shown in Fig. 2-27. This dialog lists every *.dat file in the ...\\pidas beta\\output subdirectory, including the error.dat file previously discussed. Therefore, this item allows any *.dat file

Fig. 2-30. Files from evaluation by sector only for all causes.

to be viewed and printed, not just those associated with the current batch of data that was just Evaluated. That is, results of prior evaluations can be viewed and printed with 'Old' if the files have not been deleted from the output subdirectory.

'Ancient' performs a similar service for Trending as 'Old' does for Evaluation. Specifically, 'Ancient' displays a dialog box listing every *.his file in the ...\\pidas beta\\output subdirectory. This allows histories from prior Trending analyses to be viewed and displayed, assuming they have not been deleted from the output subdirectory. Figure 2-31 shows the dialog window that is displayed as a result of selecting 'Ancient.'

2.4.5. Plot Options. When 'Select Plot' is clicked (Fig. 2-14), a message telling the user to set the printer to landscape mode is displayed, then the dialog window shown in Fig. 2-21 is displayed. The default option for this dialog is 'Plot Historical Trend' and this option was addressed in Sec. 2.3.3. If the other option, 'Plot Last Data Batch in the History,' is selected, a different set of plots is produced. The plots still provide the top 50 alarms and sectors along the abscissa ordered by running average FAR or NAR, but the ordinate is now the **FAR or NAR for the last chronological batch of data in the history** instead of the running average FAR or NAR over all batches in the history. That is, the plot shows the FAR or NAR for the last batch of data in the history in the **same order** as for the running average FAR or NAR. This plot shows the FAR or NAR for the top 50 alarms or sectors ordered by decreasing running average FAR or running average NAR for a single batch of data. This batch is the

Fig. 2-31. Dialog window for 'Ancient'.

last chronological batch of all the batches in the history. The plot can be used to see how the latest batch of alarm data compares with the historical average. Because the FAR or NAR values are for the **last chronological batch ordered by decreasing running historical FAR or NAR**, the values are not always in descending order from left to right as is the case when the running averages are plotted (the default option); this is by design because ordering by the running historical order immediately shows when an alarm or sector in the last batch has a different priority than the historical average for that alarm or sector.

Plots produced with 'Plot Last Data Batch in the History' provide similar but different information to the files produced during Evaluation. First, the plots are always for the last batch in the history, whereas the Evaluation files are for the batch just evaluated, and these are different if the batch just evaluated is not the last chronological batch in the history. Typically, data are Processed/Evaluated/Trended in chronological order, so this is not an issue. However, even when the current batch is the last chronological batch in the history, the data provided in the plots and in the Evaluation files differ unless the trend is over such a short duration that only the latest batch is included in the trend. In fact, if the files are Processed in chronological order and the duration for the trend is set at 1 day, the plots graphically show the results for the current batch of data only because the current batch is the only batch in the trend and the current batch is the latest batch. This feature can be used to provide graphical output for the current batch.

The files produced from an Evaluation (e.g., Fig. 2-15) rank-order results by the data in the current batch. The plots produced using 'Plot Last Data Batch in the History' rank-order results by running historical averages. Therefore, the order of the data may be different. For example, suppose that a particular alarm has the highest FAR for the current batch of data; this alarm will be at the top of the FAR file for alarms, which was created by the Evaluation. However, suppose that the running historical average FAR for that particular alarm—even after including the current batch in the history as a result of Trending the batch—is not the alarm with the highest running average FAR, rather it is number 10 in terms of decreasing running average FAR. On the plot, that alarm will be at position 10, not at position 1.

2.4.6. Displaying and Modifying the Master List of Sectors and Alarm Points. As mentioned in Sec. 2.2.1, the user has the option of changing the master list of sectors and alarm points, and during Processing, the code asks if changes to the list are to be made for Processing of the current *.raw file.

The master list of sector and alarm points is a list of sectors and the number of alarm points in each sector for both TA-18 and TA-55. The list can be displayed and, if necessary, modified by clicking on the '# Points' toolbar as shown in Fig. 2-14. (The list also can be displayed using the 'Points' submenu item in the 'Sector' menu item.) This results in the dialog window shown in Fig. 2-32 being displayed.

Fig. 2-32. Master sector list dialog.

The sector list for either TA-18 or TA-55 can be displayed by selecting the desired facility and clicking 'Show Sectors'. As shown in Fig. 2-33, this gives a list of all the sectors for the selected facility and allows the list to be modified by

- deleting a sector,
- adding a new sector, or
- changing the number of alarm points for an existing sector.

The master lists of sectors and alarm points for the two facilities (TA-18 and TA-55) are stored in the following *.pts files in the ...\\pidas beta subdirectory: TA18PointsInSector.pts and TA55PointsInSector.pts. These two text files are updated automatically whenever the user uses the dialog shown in Fig. 2-33 to change the list(s).

The master sector list is used to compute the average number of hits per alarm point in a sector for a given cause type. The code Processes the *.raw data file and calculates the number of hits for a specific alarm point and the number of hits for a specific sector, which is the sum of the number of hits for all alarm points in that sector. The code also calculates the average number of hits for an alarm point for a specific cause in each sector. The average number of hits for an alarm point in the sector is the total number of hits for the specified cause in the sector, divided by the total number of alarm points in the sector. The master list of sectors and alarm points provides the number of alarm points in each sector used in calculating the average number of hits per alarm point in a sector.

For example, suppose that Processing of a *.raw data file indicates that for Sector 811-02, 3 different alarm points in that sector were hit for any cause and, in total, 14 hits were recorded for the cause 'Wildlife' for all 3 alarm points that were hit for any cause. Suppose that the master list for sectors and alarm points contains 5 as

Fig. 2-33. List of sectors for TA-18.

the total number of alarm points in sector 811-02. The average number of hits for cause 'Wildlife' per alarm point in sector 811-02 for this batch of data is: $14/5 = 2.8$. Note that the average number of hits per alarm point in the sector is the average overall alarm points in that sector whether or not all alarm points were actually hit. That is, for the previous example, the total number of hits is divided by 5 (the total number of alarm points in the sector), not by 3 (the total number of alarm points actually hit in the sector for any cause).

The master list of sectors and alarm points changes with time as the configuration of the security system changes. This is the reason that the code allows changes to be made to the master list, and also requires that the user either accept or change the existing master list every time a *.raw file is Processed.

After Processing a *.raw data file, the code automatically highlights the following two situations.

- The *.raw data file contains sectors that have alarm points that were hit that are not in the master list of sectors.
- The *.raw data file contains sectors that are in the master list but for which more alarm points were hit than listed in the master list for those sectors.

If either of these two situations is encountered after Processing is finished, the code displays either or both of the dialog windows shown in Figs. 2-34 and 2-35. These dialog windows are displayed, as necessary, after the dialog box shown in Fig. 2-7 is closed. The missing sectors are recorded in the 'missing.pts' text file in the ...\\pidas beta subdirectory and the sectors with fewer points listed that actually hit are recorded in the 'wrong_points.pts' text file in the ...\\pidas beta subdirectory.

If either of the two dialog boxes shown in Figs. 2-34 and/or 2-35 is displayed, the user has two options:

1. correct the master list of sectors and re-process the *.raw data or
2. ignore the discrepancies and Evaluate/Trend/Save the Processed data.

Fig. 2-34. Dialog for missing sectors.

Fig. 2-35. Dialog for more points hit than in sector list.

The first option is preferable. However, if an old batch of data is being processed, it may be difficult to reconstruct the actual master sector list that was accurate at the time the data were collected. In that case, option 2 can be used. If option 2 is used, the code computes the average number of hits for an alarm point in the sector using the total number of hits for all alarm points in the sector for any cause for all sectors flagged in the dialog boxes as shown in Figs. 2-34 and 2-35. This is in contrast to the usual case previously described, where the average number of hits per alarm point in a sector is computed using the total number of alarm points in the sector from the master list of sectors and alarm points. The calculation is done differently because the master list of sectors and alarm points is not correct for those sectors; by using the actual number of hits, a reasonable approximation for the sector is made and this approximation is guaranteed to be conservative because the actual number of alarm points in the sector, although not known, must be greater than or equal to the number of alarm points hit for any cause in the sector.

For example, suppose that Processing of a *.raw data file indicates that for Sector 811-02, 6 different alarm points in that sector were hit for any cause and, in total, 14 hits were recorded for the cause 'Wildlife' for all the 6 alarm points that were hit for any cause. Suppose that the master list for sectors and alarm points contains 5 as the total number of alarm points in sector 811-02. The average number of hits for cause 'Wildlife' per alarm point in sector 811-02 for this batch of data is computed as $14/6 = 2.3$, not as $14/5 = 2.8$. Note the difference between this case and the usual case described previously.

Whenever option 2 is used, the *.dat output files for the current batch explicitly identify those sectors for which the average per alarm point in the sector was computed using the total number of alarm points hit.

2.4.7. Backing Up Files and Deleting Files. The PIDAS code is in the ...\\pidas beta subdirectory. The location of this subdirectory depends on where the code was installed on the user's disk. For example, the examples presented in this document were produced with the code located in the following path: D:\\darby C++\\pidas beta.

Within the ...\\pidas beta subdirectory, there are four subdirectories: debug, res, output, and input as shown in Fig. 2-36.

The Debug and res subdirectories **should never be tampered with**. The Debug directory contains the debug configuration files, and the res directory contains the graphical resources for the code. (If a Release subdirectory also is present, then a Release configuration has been created in addition to the Debug configuration. The Release subdirectory also **should never be tampered with**.)

Fig. 2-36. ...\\pidas beta subdirectories.

The input subdirectory is where all *.raw files from FSS to be Processed should be placed as discussed in Sec. 2.1.1. The output subdirectory is where the results of an analysis are placed as discussed in Sec. 2.3. In application, the code searches the input subdirectory for *.raw files to process and places the results of analyses into the output subdirectory; this is transparent to the user. The code uses C++ functions to locate the input and output subdirectories regardless of where the user places the parent ...\\pidas beta subdirectory.

After each session, the entire ...\\pidas beta subdirectory should be copied to a backup medium, such as a Jaz drive. The backup medium should have significant storage capacity (at least 1GB) because of the size of the files including the *.raw data files.

Any or all of the files in the input or output subdirectories can be deleted without affecting operation of the code. However, if a *.raw data file is deleted from the input directory, it is no longer available for (re) Processing, and if a file is deleted from the output, it is no longer available for viewing or printing, although its associated *.lst file can be Evaluated and/or Trended easily and quickly to produce the desired output. It is recommended that all the input and output files be backed up before deletion from the working subdirectories. As the working input and output subdirectories become cluttered with too many files over time, it is recommended that the older files in these subdirectories be deleted. Deletion of any file in the output subdirectory does not affect the ability to Trend because special historical *.cfile binary files are saved in the Debug or res subdirectories to effect Trending; this is discussed in Sec. 3.5.

The *.lst files are saved in the ...\\pidas beta subdirectory itself. As *.lst files build up over many analyses, these files can be deleted after they have been backed up. However, the *.lst files are not large, and it is recommended that they be deleted only as required by available system storage limitations because when a *.lst file is deleted, its associated *.raw file must be reprocessed for subsequent Evaluation and/or Trending. **No other files should ever be deleted from the ...\\pidas beta subdirectory.**

3.0. CODE DATA STRUCTURES

This discussion is current for Version 2.0 of the code.

The code uses dynamic data structures, specifically doubly linked lists on the heap, to efficiently handle data of varying size. The dynamic linked lists are implemented as instances of descendants of the MFC COBList class.

The code makes use of the ability of MFC to automatically serialize objects to and from disk storage; this is effected by deriving pertinent classes from the MFC CObject class.

The code uses the Document/View framework built into Visual C++ 5.0.

'Tracking' and 'Trending' have special connotations in the code. 'Tracking' refers to the analysis of one batch of data over one specific time period; that is, the analysis of one set of raw input data. 'Trending' refers to the analysis that considers many batches of data; that is, the historical analysis of the data from many sets of raw input data.

3.1. Data Structures for Tracking Alarms

A public data member in the document class is used to track alarms. The identifier for this data member is `m_theAlarmList` which is an instance of the class `CAlarmList`. `CAlarmList` has a private data member `m_listOfAlarms`, which is an instance of `CObList`. `m_listOfAlarms` is the dynamic linked list that stores the results of processing the raw data input file.

The raw data input file is processed into `m_theAlarmList` by the document's public member function `ProcessAnInputFile(CString)`.

`ProcessAnInputFile(CString)` adds an alarm point along with its sector to the list that it creates if that alarm occurs for any cause. Alarm points that do not alarm are not added to the list. The number of 'hits' on a given alarm is calculated as follows. After a sector is CLOSED by specifying a specific cause (e.g., WILDLIFE), the very next 'hit' for any alarm in that sector is counted as a 'hit' for that alarm because of the cause for which the sector containing that alarm is ultimately closed out; only the first 'hit' after a sector is closed is counted. The coding within PIDAS to effect this calculation from data in the *.raw data file is complicated and tricky, for two main reasons: (1) alarm hits are recorded by alarm point and closed out by sector and (2) the number of records between a hit on an alarm point and a close out of the sector for that alarm point is variable depending on how many other alarm point hits occur in the meantime. The source code is heavily annotated with comments to explain exactly how this is accomplished. Basically, the following points are pertinent.

- Alarms come in by alarm point.
- Alarms are closed out by cause for each sector, which closes all alarms in the sector.
- Only the first 'hit' for an alarm point after its sector is closed out is counted; subsequent hits are not counted until the sector is ultimately closed out again, after which the next first hit is counted.

3.2. Data Structures for Tracking Sectors

A public data member in the document class is used to track sectors. The identifier for this data member is `m_theSectorList`, which is an instance of the class `CAlarmList`. `m_theSectorList` is similar to `m_theAlarmList` except that certain data in the base class `CAlarmList` are changed to roll-up alarms within the same sector.

The sector list is used to tabulate the average number of alarms by cause for the sector (average denoting average for the specific alarm points in the sector). As discussed in Sec. 2.3.4, the average number of hits for an alarm point in the sector is the total number of hits for the specified cause in the sector, divided by the total number of alarm points in the sector. The master list of sectors and alarm points provides the number of alarm points in each sector used in calculating the average number of hits per alarm point in a sector.

3.3. Data Structures for Trending Alarms

The trending of alarms is maintained by the document's public data member `m_historicalList[2]`, an array of two instances of `CHistAlarmList`. In the remainder of this discussion, `m_historicalList[i]` refers to either one of the two elements of the array, so *i* can be either 0 or 1. There are two elements in the array, one to trend TA-55 and the other to trend TA-18. The constituent data structures for Trending are complicated and are best described by a figure. Table 3-1 shows the data required for trending in a tabular form; Fig. 3-1 shows the data structures that implement the structures indicated by Table 3-1.

TABLE 3-1
DATA REQUIREMENTS FOR TRENDING

Batches of Data → Alarm Histories ↓	1	2	3	4	5	6	New Batches Added as Needed →
A	x ¹		x	x	x		
B		x	x	x	x	x	
C	x	x		x		x	
D		x	x	x	x	x	
E	x	x	x	x			
F	x	x	x	x	x		
New Histories Added As Needed ↓							

¹Data are FAR and NAR per alarm per batch. Absence of data implies both FAR and NAR are 0 for that alarm for that batch.

m_HistoricalList[i] consists of two dynamic linked lists, one containing instances of CATimeBatch and the other containing instances of CHistoricalAlarm. Each instance of CATimeBatch represents one time batch of data: the 1, 2, 3,... batches of data in Table 3-1. Each instance of CHistoricalAlarm represents a set of data for one alarm: the A, B, C, ... alarm histories of Table 3-1.

CHistoricalAlarm also contains an inner dynamic linked list of instances of COneAlarmList: the x data sets of Table 3-1.

The FAR and NAR data created during Trending consists of running historical average values as defined in Sec. 2.3.2.

3.4. Data Structures for Trending Sectors

The trending of sectors is performed with data structures similar to those used for trending alarms, except that different (but similar) classes are used. The trending of alarms is maintained by the document's public data member m_histSectorList[2]; two instances of CHistSectorList.

For sectors, CHistoricalSector is analogous to CHistoricalAlarm for alarms. For sectors, COneSectorList is analogous to COneAlarmList for alarms.

The code automatically updates the history as batches of data are processed. Even if batches are processed out of chronological order, the history maintains all batches within 1 yr of the latest batch. The latest batch is the batch with the latest end date; the current batch is the batch being processed. These batches are the same only if the files are processed in chronological order. The history for the current batch will contain batches past the current batch if the current batch is not the latest batch. It is possible to reset the history by replacing the *.cfile binary files in the ...\\pidas beta subdirectory with specific *.cfile binary files from the output subdirectory, as described in Sec. 3.5 of this report. However, if this is done, the prior history used automatically by the code is lost, having been replaced by a different history.

For example, suppose that batches have been processed with end dates running from 1/5/97 through 11/23/97; the automatic history feature of the code includes all these batches. If a batch with an end date of 6/5/97 is processed, the history will be updated so that it includes the current batch ending 6/5/97, but the history end date is still through the latest batch ending 11/23/97 and all batches through this latest batch are in the history. The only

^a1,2,3,...,A,B,C,..., and the x's refer to Data Needs as Noted in Table 3.1.

Fig. 3-1. Data structures for trending.

way to run the 6/5/97 file out of chronological order and have the history stop at this batch is to overwrite the *.cfile binary files in the ...\\pidas beta subdirectory with the specific histories for the last batch that was processed before the 6/5/97 data, for example, the 5/3/97 batch, as described in Sec. 3.5. Then, because the automatic history files have been rewritten to only contain batches through 5/3/97, subsequent processing of the 6/5/97 batch will result in the new updated history going through 6/5/97. However, this new history applies to all subsequent processing so that if the next file processed has an end date of, for example, 12/25/97, the history will 'jump' over batches from 6/5/97 through 11/23/97 because the master history files were overwritten to exclude these batches. To create a history only through the 6/5/97 batch but then update the history for all the subsequent batches, each batch from 6/5/97 on must be reprocessed after the master history files were replaced with the history through 5/3/97. **The files should be processed in chronological order or things become very confusing.**

If it is desired to reprocess all files starting with no prior history, then both of the *.cfiles in the ...\\pidas beta subdirectory should be deleted. The history will be rebuilt, and these files will be recreated to maintain that history as the files are subsequently processed.

Note that selecting a duration for Trending as described in Sec. 2.2.3 of this report has no effect on the automatic history feature of the code because Trending merely provides output of the history for the selected duration and in no way changes the actual data in the history.

3.5. Serialization

For tracking, serialization is automatic per the standard MFC document/view framework.

For trending, custom serialization to/from two C binary files (instances of CFile) is used. The two files are: TA18 total history.cfile and TA55 total history.cfile; these two files retain data for TA-18 and TA-55, respectively, from prior analyses of batches of data so that trending including the current batch of data can be performed. These two files are located in the ..\\pidas beta subdirectory

These files contains the results of all alarm and sector histories for all batches—through the last batch—whose end dates are not older than 1 yr from the end date of the last batch. Data older than 1 yr are deleted from the files automatically.

The *.raw data files do not have to be analyzed (Processed/Evaluated/Trended) in chronological order. That is, if a raw data file is missed, it can be analyzed later after files with newer dates have been analyzed; the code automatically inserts the batch of data into the proper chronological position in the history.

If the same *.raw data file is analyzed more than once, it is not double-counted in the history; the code automatically prevents this from occurring by deleting the earlier data and overwriting it with the newer data.

To provide a recovery capability in case problems arise during application of the code, two additional instances of CFile are created for each current batch of data that is analyzed. These files are named for the current batch of data; for example, if the current batch start and end dates are 6_3_1997 and 7_5_1997, the two files are: TA18 6_3_1997 to 7_5_1997.cfile and TA55 6_3_1997 to 7_5_1997.cfile. These files are located in the output subdirectory.

After the analysis of a current batch of data is saved, the data in the CFiles are redundant; specifically, TA18 total history.cfile and TA18 6_3_1997 to 7_5_1997.cfile have identical data, and TA55 total history.cfile and TA55 6_3_1997 to 7_5_1997.cfile have identical data. The reason for having two sets of files is for recovery. Suppose the next batch of data that is analyzed has start and end dates of 7_7_1997 and 7_15_1997. After analysis and saving, TA18 total history.cfile and TA55 total history.cfile have been updated to reflect the new data (also, TA18 7_7_1997 to 7_15_1997.cfile and TA55 7_7_1997 to 7_15_1997.cfile with the same data have been created). Suppose there is a problem with this last analysis—bad data, problems in the application, corrupt files, or whatever. Furthermore, suppose that many batches of data were included in the last 'good' analysis, that being analysis of the 6_3_1997 to 7_5_1997 batch of data. If only the TA18 total history.cfile and TA55 total history.cfile files were saved the entire set of batches would have to be reprocessed because these two history files may be corrupt and no prior history files would be available. However, because for each analysis of a specific batch of data specific history files also are saved, e.g., TA18 6_3_1997 to 7_5_1997.cfile and TA55 6_3_1997 to 7_5_1997.cfile, the last set of 'good' history files can be used to overwrite the TA18 total history.cfile and TA55 total history.cfile files and Processing need only be repeated for the set(s) of data past the batches that were included in these files.

In summary, the code uses two binary disk files—TA18 total history.cfile and TA55 total history.cfile—to maintain the current histories for the two facilities. The code also produces, but does not use, two history files specific to each analysis. These latter files are for recovery purposes only, or for resetting the history as previously discussed in Sec. 3.4.

Also, each batch of data is unique to one facility, either TA18 or TA55, so each analysis of a batch of data only updates one history file, either TA18 total history.cfile and TA55 total history.cfile as appropriate. However, in application the code re-writes both history files after the analysis; the file that is not associated with the facility for the current analysis is rewritten with no additional data added. The same is true for the batch specific history files,

e.g., TA18 6_3_1997 to 7_5_1997.cfile and TA55 6_3_1997 to 7_5_1997.cfile; two files are created for each analysis although only the file corresponding to the facility for the current batch of data has new data appended to it.

4.0. RESULTS OF TESTING

The authors of this report tested the PIDAS code on about 400 Mbytes of actual data for TA18 and TA55 provided by FSS. This section documents the results of that testing. Aside from the types of problems discussed in the following two subsections, the code performed well and produced accurate results.

4.1. Errors in the *.raw Files

During testing, two types of errors were noted in the *.raw data files.

- Garbage data in a few records
- A spurious final record in the file

These two errors and how to correct them are discussed in Secs. 2.4.2 and 2.1.1, respectively. The first type error is not a fatal error for the code; however, garbage data will be carried through the analyses. The second error is a fatal runtime error for the code as explained in Sec. 2.1.1.

It is highly recommended that these two types of errors in the *.raw data file be fixed and processing be redone before performing Evaluation or Trending.

4.2. Unanticipated Fatal Runtime Errors

During the analyses of some of the *.raw data files using Version 1.0 of the code, the Processing and Evaluation steps were performed, but the Trending step was not performed. When *.lst files were saved for these analyses in which Trending had not been performed, then later the *.lst files were opened, Trending was performed, and the *.lst file was resaved and a fatal error occurred. This is believed to be a result of attempt to serialize different objects to the same output file. In one case, this type of error necessitated rewriting the main two *.cfiles with the last correct results before the code would function properly (see Sec. 3.5).

This error was isolated and fixed in Version 2.0 of the code. The fix was to modify the OnSaveDocument() function in the CPidasDoc class so that data are not rewritten to existing *.cfile files in the output directory if these existing *.cfile files cannot be opened.

In actual application, if the code crashes for some unknown reason and cannot be restarted, we recommend that the current *.cfiles in the ...\\pidas beta subdirectory be overwritten with the latest correct *.cfiles from the output subdirectory, for both the TA18 and the TA55 *.cfiles. This procedure was discussed in Sec. 3.4.

5.0. SOURCE CODE

The source code for the program is too voluminous to attach as hard copy. About 20,000 lines of custom code were written for this application. The Visual C++ framework also generates much 'standard' code to which the custom code was appended and inserted. An Iomega Zip drive disk (100-Mbyte capacity) is available from the authors with the entire program source code.

REFERENCES

Ogden Exterior Code. 'Los Alamos National Laboratory FSS-10 PIDAS FAR/NAR Database Operating Instructions,' Rev. 1 (December 9, 1994), Ogden Environmental and Energy Services Company, Inc.