

Unified Logging System

Antony Mapfumo (02786923)

24th December 2004

Course: EE74 Master of Engineering Science (Computer & Communications)

Subject: EEP301 Project

Supervisor: Dr. Chandran

Abstract

Australia Post operates optical character recognition based mail sorting machines. Occasionally they have faults which vary in severity. These faults are logged and sent to a central server for analysis. This project focussed on developing tools to allow technicians and managers to view and analyse these faults in an intuitive manner with the aim of using this information to improve mail delivery service and reduce costs along the way. A unified logging system allows for the definition of a common log file format with which the various kinds of mail sorting machines need to adhere to with the aim of analysing faults from a central point.

A log analysis tool prototype was already in place. Several enhancements and additions were made to it. The user interface was improved to make it more intuitive, search facilities were enhanced, and new search capabilities were introduced. The ability to automatically email reports to users was introduced as was machine comparison analysis. Improvements in log file storage and caching resulted in the time required to generate reports being improved. A simple but powerful user guide was developed as was the detailed design documentation.

Statement of original authorship

The work contained in this project report has not been previously submitted for assessment at any other tertiary educational institution. To the best of my knowledge and belief, the project contains no material previously published or written by another person except where due reference is made.



Antony Mapfumo

Date: 29th December 2004

Statement of project completion

The minimum requirements as specified in the project proposal have been met and where a requirement has not been met a satisfactory reason has been given.

Queensland University of Technology supervisor

 Date: 4/1/05
-ATPROF. V. CHANDRAN,

Australia Post supervisor

 Date: 4/1/05
FOR R. LUNDON

Acknowledgement

I would like to acknowledge and thank the following people for their assistance and patience during the completion of this project:-

- Robert Lunnon
- Dr. Vinod Chandran
- The staff at Australia Post, Network Engineering Unit, who helped me in finding my way around and who made nice barbeques at the end of every month whilst I was there which helped in keeping me relaxed.
- Suzanne Garvey for helping with cooking and proof reading my thesis

Contents

- Abstract** 1
- Statement of original authorship** 2
- Statement of project completion** 2
- Acknowledgement** 3
- Table of figures** 5
- 1 Introduction** 6
- 2 Application of OCR to mail sorting**..... 8
- 3 Problems associated with mail sorting** 10
 - 3.1 Event logging 11
- 4 Mail sorting challenges** 12
- 5 Project Objectives** 13
 - 5.1 Improvements to the original prototype 14
 - 5.2 New features added to the prototype 16
 - 6.1 Anticipated value of the project 18
 - 6.2 Actual project outcome 18
 - 6.2.1 Technical improvements to enhance the accuracy of the FMOCR data 18
 - 6.2.2 Log file information restructuring to align to 6AM-6AM day 19
 - 6.2.3 Implementing hierarchical relationship analysis 19
- 7 Research Methodology** 20
- 8 Project outcome, functionality and value**..... 21
 - 8.1 Project Value 21
 - 8.2 Project Functionality 21
- 9 Lessons Learned** 27
 - 9.1 Problems faced 27
- 10 Future improvements**..... 28
- 11 Conclusion**..... 29
- 12 Glossary**..... 30
- 13 References** 31
- 14 Appendix A – User guide** 32
- 15 Appendix B – Design documentation** 59
- 16 Appendix C – Source code**..... 82

Table of figures

Figure 1 - Event logging and central log server relationship	11
Figure 2 – New user interface	22
Figure 3 - Sample email report.....	25
Figure 4 - Machine Comparison analysis.....	26
Figure 5 - RIS main web page snapshot.....	33
Figure 6 - FMOCR main web page snap shot.....	34
Figure 7 - FMOCR trend form	34
Figure 8 - Query by Event Text	36
Figure 9 - Query by Event Type.....	36
Figure 10 - Query by Module.....	37
Figure 11 - Query by Module ID	37
Figure 12 - Query by duration.....	38
Figure 13 - Query by criticality.....	38
Figure 14 - Query by impairment.....	39
Figure 15 - Query by time	39
Figure 16 - Query by watch list.....	40
Figure 17 - Multi-field search [2 fields].....	41
Figure 18 - Multi-field search [3 fields].....	41
Figure 19 - Multi-field search [4 fields].....	42
Figure 20 - Sample trend output (Page top)	42
Figure 21 - Sample trend output (Page middle)	43
Figure 22 - Sample trend output (Page bottom).....	43
Figure 23 - Do an FMOCR Log Report	44
Figure 24 - FMOCR Log Reporting.....	44
Figure 25 - Top 10/20 Report for yesterday.....	45
Figure 26 - Quick watch list report for yesterday	45
Figure 27 - Quick watch list report for any day	46
Figure 28 - Quick watch list report for any day sample results	46
Figure 29 - Watch list registration.....	47
Figure 30 - Registration confirmation page	48
Figure 31 - Watch list editing login page snapshot.....	48
Figure 32 - Watch list editing introduction page snapshot.....	49
Figure 33 - Creating a watchlist	50
Figure 34 - Adding watch lists to your profile	51
Figure 35 - Email reports (Top events)	52
Figure 36 - Email reports (watchlist comparison).....	53
Figure 37 - Email reports (history graph).....	54
Figure 38 - Daily quick summary	54
Figure 39 - Stop events by duration snapshot	57
Figure 40 - Comparison Graphs	57

1 Introduction

Australia Post is one of Australia's top ranking companies. They handle more than 19 million articles of mail each working day or over 4.7 billion articles per year and provide service to the one million customers who visit their retail store outlets each day. The aim of this project was to integrate event information from its Flat Mail Optical Character Reader (FMOCR) automated mail sorting machines to a central repository, and to develop tools to search and collate this information in an easy and intuitive way. Flat Mail Optical Character Readers machines sort large letter mail (known as Flats) such as magazines and A4 envelopes. As these machines incorporate pattern recognition a brief insight into optical character recognition will help in understanding their functionality. Optical character recognition is the recognition of printed or handwritten text by a computer system so that the text can be stored in computers as characters instead of images [6]. At the heart of every OCR system is a scanner for reading text and software for analysing images.

The first step involves using a scanner to acquire an image, as a bitmap, containing the text you are interested in. The next step involves using software based on complicated image processing techniques to identify the individual characters in the document. To improve accuracy, modern OCR programs make use of multiple algorithms of neural network technology to analyse the text in a document [7]. They can better analyse the stroke edges and distinguish text from the background. To cater for irregularities of printed ink on paper the algorithms averages the dark and light regions along a stroke to make a better decision as to what character it is. The main advantage of an OCR system is that it makes it easier and quicker to read text from printed or handwritten material [1]. Better hardware and image processing algorithms have improved OCR accuracy over the years.

Using a good scanner is the first step in getting better results. A high resolution scanner will ensure that there are more distinct colours or more distinct shades of grey in the case of black and white text. High resolution scanning also improves recognition of very small type especially those under 6 points. In the past acquiring high resolution images meant more time was required to read the image which would be undesirable in a mail sorting operation which handles more than 10, 000 pieces of mail per hour. However, recent parallel and dedicated hardware are faster at

acquiring images for character recognition. Some OCR systems occasionally find it hard to distinguish similar characters like **l** and **1**, or **0** and **o**. This is especially true in recognising hand written text. Some systems overcome this problem by training the OCR system to handle noisy input and similar looking characters. The more training the system gets the more accurate will be the character recognition over time.

2 Application of OCR to mail sorting

To understand the benefits of using optical character recognition techniques for mail sorting let's have a quick overview of how postal mail centres use to sort mail in the past. Mail and parcels were collected from post boxes and post offices and then transported to mail processing centres. Employees then looked at the address of each and every item and passed it to the appropriate section for further processing by other employees [8].

As the volume of mail processed at mail centres around the world increased it was quickly realised automating the postal sorting process was necessary if mail was to continue to be delivered in time [3]. Research into the use of optical character recognition for postal sorting started in the 1950s but it was not until 1982 that the first OCR based machine was utilised by The United States Postal Service [4]. Before OCR was used previous mail sorting automation machines were Multi-Position Letter Sorting Machines (MPLSM) whose operations were largely manual and required 17 operators. By the end of 1984 more than 250 OCR based mail sorting machines were being used the US Postal Service and they were processing more than 20,000 mail pieces per hour. At this stage the machines could not process handwritten mail pieces. Over the years improvements have been made to the accuracy of OCR based systems as well as the ability to handle hand written mail pieces.

On average an Australia Post mail sorting employee can process 1,000 mail items per hour whilst optical character based mail sorting has a peak performance of around 28,000 items per hour. From these figures, it is clear that a mail sorting machine can process the same amount of mail 24 employees (around 4 people are needed to operate the FMOCR machines) can process per hour. Whilst these are rough figures they give a good indication of the savings and efficiency associated with OCR based machines to sort mail. Mail sorting machines at Australia post make use of optical character recognition technology to achieve on time delivery of mail. The FMOCR machines are up to 82 m long and incorporate up to 480 terminal stations (most of these terminals are output terminals that do not require human intervention). They are capable of automatically sorting up to 28,000 large letters an hour including plastic wrapped items. These machines scan the address block on each A4 sized mail item to determine the postal and the delivery point information. They then verify the scanned

information against an internal database. Like any other OCR based technology FMOOCR machines are susceptible to bad hand writing (at Australia Post handwritings are recognised about 65% of the time and printed materials are recognised 85% of the time), however the problem is not so bad because the majority of the mail processed is in printed form. A video character recognition system (VCR) together with employees provide backup to the FMOOCR machines. Once a mail item is successfully scanned it is then bar-coded for further processing by barcode sorters which are very fast mail sorting machines. If a mail item is already bar-coded the whole optical scanning and recognition is skipped and the mail item is transferred directly to the bar code sorter.

3 Problems associated with mail sorting

The FMOCR mail sorting machines are very big, complicated and consist of large networks of programmable logic devices (PLCs) monitoring and helping with the mail sorting process. Some of the PLCs monitor the thickness of mail items whilst others do tasks like counting mail items. The following paragraphs explain some of the errors which can occur. Some errors require the machines to be stopped and the sorting operation cannot proceed until the problem has been sorted. These events are called impair or stop events depending on the duration of the event. Some events are just warnings. Whilst they do not require the machines to be stopped, they are a good indication that an impair event will occur soon if they are not attended to.

The machines require mail to be of certain maximum dimensions (about A4 size). When mail pieces larger than the stipulated size are put into the machine a couple of errors will be generated when the mail item jams sections of the machine. For example, once it blocks the entry into the feeder all other mail items cannot go through and the conveyor belt will be automatically stopped until an operator removes the blocking item. A "feeder jam" error together with a "conveyor stopped" messages might be generated. Each time an errors occurs it is logged into a log file including the duration of that event.

For safety reasons operators or any other object are not allowed within certain sections of the machines (they have proximity sensors in those areas). When those proximity violations occur the machine stops and logs an event to the log file. The other major problem which affects these machines is sequencing errors. Sequencing errors occur when an actuator fails to receive a mail item within a specified amount of time. Sometimes more than one mail item try to go past the optical scanner. By measuring the thickness (items are not allowed to exceed certain thickness) and weight of the "item" the machine can easily determine that there is more than one item. Other errors happen when the printer happens to be offline when the mail item is about to be bar-coded. The FMOCR machines have more than 200 different types of errors some of which are subsets or combinations of the ones already mentioned.

3.1 Event logging

The FMOCR mail sorting machines are interfaced to personal computers running Microsoft Windows NT. Events are logged to the host computer through an RS232 serial port "as they happen". When there is a mail jam, for example, a jam related event is sent to the computer for logging including the duration of that event.

Every morning the mail control computer (MCC) connects to the central log server using the file transfer protocol (FTP) and uploads the previous day's log file. It is convenient to upload the log files in the morning because that is the time when the machines are usually idle.

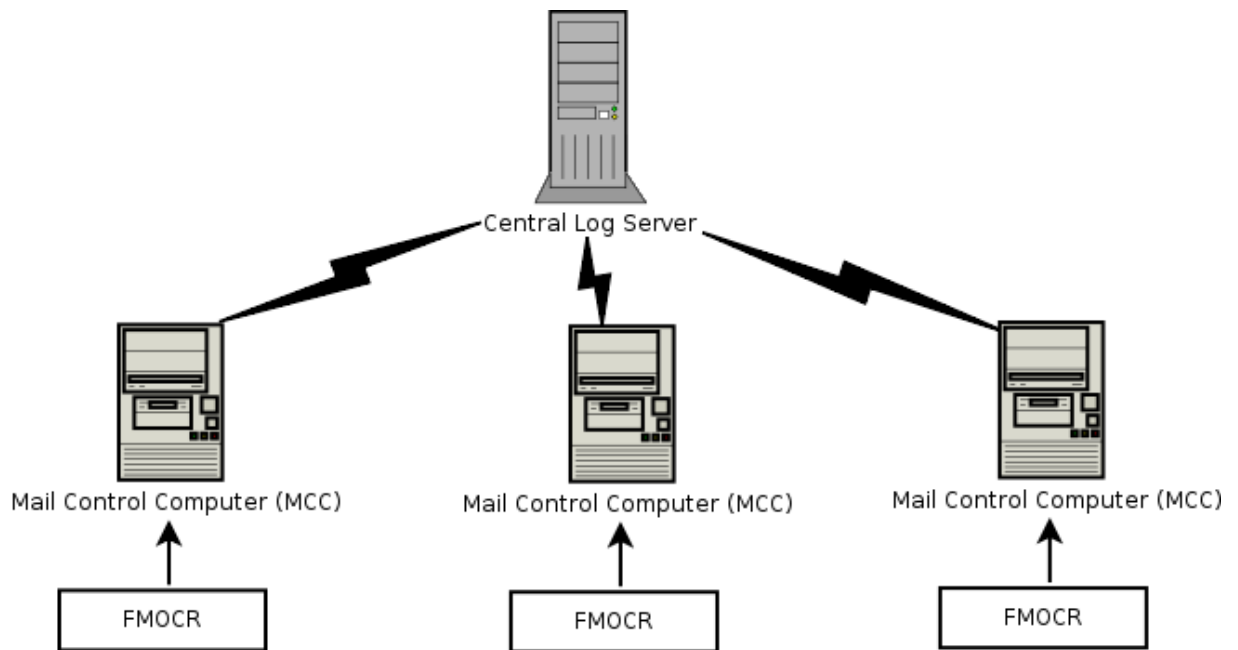


Figure 1- Event logging and central log server relationship

There are plans for the FMOCR events to be logged in "real time" to a remote location, the central log server, in the future. This will enable events to be included in reports as they arrive instead of waiting for the mail control computers to send events in batches each morning.

4 Mail sorting challenges

The operation of mail sorting centres brings about many challenges arising mainly from the volume of mail processed and the need to deliver it on time. Australia Post is under government obligation to deliver mail on time 94% of the time. External auditors (KPMG) enforce this requirement by sending mail at random times from random locations to see if it arrives on time. Malfunctioning mail sorting equipment can delay mail delivery and affect businesses that depend on mail being delivered on time. Jammed sorting machines can cause backups in mail centres and require labour intensive hand sorting to clear the backlog.

The main challenge is to figure out how to make better use of all the log files generated by the mail sorting machines in order to better anticipate problems before they happen thereby increasing efficiency and saving costs at the same time. Sometimes technicians might want to know what stopped a machine at a certain date and time. They might also want to know what other events happened before the event that impaired the machine. Without log files it would be very difficult leaving them having to rely on machine operators having to remember what happened that day.

Another challenge is that even if the log files are available they are not of much use if they are not properly managed and if there is no infrastructure in place for you to quickly and easily get the information you are after. For example, if it takes an hour to go through the log files to find the information you are after the machine may have to remain idle during that period bearing in mind that some of these machines have to process more than 28,000 pieces of mail per hour. The aim is to spend more time repairing the machine than to finding out what went wrong hence optimising downtime. The mail sorting machines cost hundreds of millions of dollars so from a management point of view they need to find out if they are being fully utilised. One way to find that out is to compare the uptime versus downtime. Without appropriate event logging and a mechanism to analyse that information this becomes an enormous challenge. Management are also interested in knowing which mail centres might need more sorting machines to handle the ever increasing amount of mail they handle. Without proper statistics this again becomes a difficult task to achieve.

5 Project Objectives

Australia post prototyped its infrastructure on its Flats Multi-line Optical Character Recognisers (FMOCR) and wrote a parser to bring FMOCR Event data into the ULS data-store and several web based tools for undertaking simple analysis of that data. Before this project started this infrastructure was in its infancy and needed more work done on it.

The following is the list of project objectives which needed to be achieved:-

- Improve or re-write the log trend reporting tool
- User Interface improvement
- Several enhancements to search facilities
- Technical improvements to enhance the accuracy of the FMOCR data in particular
- Improvements in data store to improve web server response times
- Addition of several new analyses and improvements to presentation
- Architectural and detailed design documentation
- Design and implementation of user help and user manual
- Log file information restructuring to align to 6AM-6AM day
- Capability to directly request a graph for a particular date using the trend form
- Cache data on web server to eliminate network latency
- Upgrade web server speed to run analysis scripts faster
- Data quality improvement
- Include new analysis methods including having log reports by criticality
- Implement hierarchical relationship analysis
- Implement email reports and alerts
- Implement machine comparison analysis

New features introduced by this project are discussed in section 5.2 whilst improvements to the prototype are discussed in section 5.1

5.1 Improvements to the original prototype

Improve or re-write the log trend reporting tool

The original trend tool (known as *fmocrtrend* in the prototype) is the main program responsible for reading the log files, analysing them and finally utilising a graph plotting utility (Gnuplot) to graph the data. To implement the required enhancements to requires re-writing or modifying the prototype tools. These enhancements include the ability to accept multiple search strings, reading search information from a file and the ability to accept search strings with spaces in them.

User Interface Improvement

The user interface needs improvement to make it easier to select trends from the log report page. There should be a way to present to the user choices of key values applicable for the field selected (on the trend page). Improvements should be made to the FMOCR tools web side to make access to information as straight forward as possible.

Enhancements to search facilities

Some of the enhancements include multi keyword searches and the ability to type in a more descriptive search rather than a one word search. For example, it is desirable for the users to type in "Shuttle Jam" instead of just "Shuttle". Users should also be able to use multi-field searches. For example, they should be able to search by event module, event time and event date. There should be quick links for searches that users usually require. A link to top 10 stop events for yesterday or last week is a good example of such a link that would enhance the functionality. Users should also be able to define a collection of queries and put them in a file that can be processed as a batch.

Make the section Index more intuitive

The objective is to allow the user to enter embedded spaces in trend query key values. The query "Shuttle Jam" is more intuitive than just "Shuttle" and the user should be able to get more accurate results that way.

Technical improvements to enhance the accuracy of the FMOCR data

This involves sorting the problem which currently happens when a "stop time" overlaps into the "start time". The idea is to mark such log entries as unreliable so that it will be up to the user to decide to incorporate them or discard it.

5.2 New features added to the prototype

Architectural and Detailed Design Documentation

There was a need for detailed documentation of the log tools design and implementations including the current ULS prototype with the aim of making it easier to understand, add more features as necessary and correct errors if they arise.

Design and implementation of user help and user manual

A simple and intuitive help manual was to be written. It was also desirable to have a simple search facility in the help manual together with a link to frequently asked questions and a quick guide.

Capability to directly request a graph for a particular date using the trend form

After enhancing or re-writing the trend tool it should be possible for users to search for a graph from a particular day. Prototype implementation currently allow users to request trend graphs for current day, week, fortnight, month and two minutes only.

Data Quality Improvement

Prime parser with previous day's data and determine open events at the end of the previous day. Allow durations to be calculated for events open at the end of the previous day.

Reduce ambiguity of event data

Currently, bad records are indicated by an exclamation mark in the duration field but the duration is specified to be a floating point numeric field in the universal log file format specification. There needs to be another way to flag an incompletely written record other than violating the file format specification.

Upgrade Web Server Speed to run analysis scripts faster and cache data on web.

The web server was to be moved to a computer with faster processing speed and the log files were to be cached on the local web server instead of the current file server for faster execution. This was to ensure that only new files were copied from the file server when the tools are executing thereby reducing network traffic.

New Analyses

The objective is to have a report by criticality and count or duration and specific operationally targeted reports, for example "Feeder Jam Reports". Users should also be able to define their own report requests.

Machine Comparisons

It was suggested that a daily comparison for all events that meet a suitable criteria should be automatically generated everyday. This was already being done manually but needed to be done here as either an "on demand" report or a daily pre-generated one. The criteria for events to make it to this report were to require input from the users.

Email reports

It was suggested that we could produce a standard set of reports (the log report for yesterday, for example) each morning and e-mail to users. In addition to the daily standard reports, users needed the ability to define specific events they are interested that would also be emailed to them.

Email alerts

It has been suggested that if during morning processing we detect that something significant has happened that we send an email alert to a site nominee. This is much more difficult than e-mail reports because it is necessary to establish thresholds for deciding when something is significant.

6AM-6AM alignment

It was suggested that the tools align log processing to the 6AM-6AM reporting periods. This has the undesirable side-effect that reports can not be generated until at least 6AM local time. The problem was apparent when getting log files from Perth where the log files could be as late as 9AM in arriving at the log server during daylight saving in the eastern states. Such a change could adversely affect many users. Maintenance and headquarters users will be affected the most by such a time alignment.

6 Anticipated project outcomes

The following were the original anticipated project outcomes.

- Highly responsive and easy to use logging system
- High quality log information
- Ability to predict future problems based on current log information
- Detailed Design Documentation thereby enabling other machines to be easily incorporated into the Unified Logging System

6.1 Anticipated value of the project

The overall objective of the project was to improve the reliability of the mail sorting machines. It was hoped that the completed project would result in quality log information being readily available. This information could be used for a number of purposes but mainly to improve the efficiency of Australia Post's mail sorting machines. The data from the files could be used for planning purposes including trying to predict when problems might occur so that in the eventuality of such a case the right people will be well prepared to deal with the problems.

6.2 Actual project outcome

All of the objectives were met except for three. The explanation is given directly below for these three. The functionality of the rest of the objectives which were met will be discussed in chapter 8, *Project Outcome and Functionality*.

6.2.1 Technical improvements to enhance the accuracy of the FMOCR data

The solution required the cooperation of the manufacturer of the FMOCR mail sorting machines as it requires a review of how the data is logged. Some of the machines are still under warranty which means only the manufacturer can rewrite PLC (programmable logic device) programs to address the issue. We could not arrange the cooperation of the manufacturer in a timely manner.

6.2.2 Log file information restructuring to align to 6AM-6AM day

The idea of aligning log file information between 6am and 6pm was abandoned because of logistic and practical reasons. Most of the time technicians do maintenance work very early in the morning well before log files arrive from mail sorting centres.

6.2.3 Implementing hierarchical relationship analysis

Sometimes machine errors are related, certain events happen just before a series of other events happen. The idea of implementing a hierarchical relationship was to automatically link events when the user does a query so that they are presented with related events that happened before the event they selected. The FMOCR machines can generate more than 200 different error messages. It was quickly realised that there are too many permutations. Furthermore, information regarding the relationship between all linking events was not readily available. It still could be done, but as a separate project. Perhaps this was being too ambitious.

7 Research Methodology

The research involved fully understanding the current prototype and how the mail sorting machines work in general. The other part of the research involved a thorough comprehension of C/C++, GNU/Linux and Bash Shell programming literature in order to evaluate how best it could help with the implementation of the logging system since the prototype was based on these. Since the whole project was to be implemented using open source tools more research was put into them particularly the use of *Gnuplot* used for graph plotting and curve fitting. Research was also done on the use of using the g++, the GNU/C++ compiler, particularly on how to optimise programs written in C++. Program profiling was researched with the aim of finding bottlenecks in algorithm execution and improving on those. Securing and optimising the Apache web server was also researched.

Extensive use was made of the Internet ranging from research on mail sorting to optical character recognition. The research also focussed on visiting the actual mail centre to see the FMOCR machines in action. It also involved interacting with Australia Post mail sorting employees with the aim of acquiring more knowledge of the mail sorting process and typical problems the mail sorting machines encounter. Finally the FMOCR machines come with extensive documentation on compact disc which was also used as part of the research.

8 Project outcome, functionality and value

8.1 Project Value

Australia Post can now use the information retrieved from the log analysis tools created by this project to improve its mail delivery across Australia. It now has a better ability to anticipate certain problems thereby reducing down time, cutting costs and improving the mail delivery service along the way. Australia Post now has an infrastructure in place to store and retrieve mail sorting machines' error messages in an easy and productive way.

8.2 Project Functionality

The project consists of programs written in C/C++ which interact with the user through web pages. A tool called **fmtrend** accepts user queries then reads appropriate log files, plots a trend graph and then displays, to the user, the total number of records found, their duration, the unique records found, the total number of files included, the time it took to generate the report and more importantly a trend graph. Examples of such a graph will be shown in the sections below.

Fmreports was developed specifically for generating reports that will be sent to users by email. It takes a number of arguments including the ability to read user specific watch lists. Watch lists are lists of error messages the user is interested in. Other tools are "helper" tools to the above mentioned tools (fmtrend and fmreports). For example, a tool called **fmproc** takes the name value pairs from a form query and split it into its constituent parts. Most of the project functionality can be better understood by reading the **user guide** in the appendices section of this document. The following sections are summaries of what the specific tools developed for this project do. A technical description of the tools is in the **detailed design section** of the appendices.

Improve or re-write the log trend reporting tool

The original prototype did a good job but in order to implement the required functionality it had to be rewritten completely because of the limitations it had. For example, it could not handle phrases as search strings very well (it was limited to using single word queries). Furthermore it had no ability to handle multiple search strings. The new tool is faster than the prototype. Large queries which used to take closer to 120 seconds can now be performed well under 15 seconds. The tool now performs queries on all different aspects of the log file (date, duration, event text, module, etc) and these fields can be combined to help produce better results. The tool now handles invalid data and illegal input better with the ability to explain to the user if there are errors with their input. A good example is the ability to report to the user that a log file is missing and therefore the events in that file will not be included in the report. Empty search fields are now being handled appropriately.

User Interface improvement

The user interface was improved and it now allows the user to enter more than one search string. The diagram below shows the new interface for the *fmtrend* tool.

The screenshot shows a web-based user interface for the *fmtrend* tool. At the top, there are three dropdown menus: 'Machine Number' set to '701', 'Period' set to 'DAY', and 'Plot Interval' set to 'DAILY'. Below these are three identical search rows. Each row consists of a 'Field' dropdown menu (all set to '7:Event Text') and a 'Search String' text input field (all containing 'NONE'). Underneath the search rows, there are two text input fields: 'User Name' containing 'global' and 'Watch List' containing 'NONE'. At the bottom of the form, there is a bold instruction 'Click submit to generate' and a 'SUBMIT' button.

Figure 2 – New user interface

It is now easier for the user to select the machine they are after, the period (day, fortnight, month or as far back as they system has log files for) and the search fields they want. A detailed explanation of how to use the new interface is explained in the help section of the appendices.

Enhancements to search facilities

Besides allowing multiple search strings to be included in the query as shown in Figure 2 the system now allows the user to use phrases and watch lists. Instead of querying by a single word like "Shuttle" a user can now specify that they want shuttle jam errors by typing in "Shuttle Jam". This allows the user to use more descriptive search strings and they also get more accurate results that way. By meeting this objective, the project objective "**make the section index more intuitive**" was also met.

Addition of several new analyses and improvements to presentation

The reports and trend graphs are now presented in a clear form. For example, the trend graphs now include a moving average line (see user guide). The reports are also being more clearly formatted with the aim of making them easy to pick up essential information. For example, the grand totals on machine comparisons analysis are shown in bold. Histograms are used to give a graphical comparison of machine performances. New analysis includes the ability to search all events within time or duration periods. For example, you can now search for all events whose duration is between 20 seconds and 56 seconds inclusive or all events that happened between 2334HRS and 2356HRS.

Architectural and Detailed Design Documentation

The appendix section "detailed design" contains a copy of the design. C++ classes and their methods are clearly explained. Source code is heavily commented so as to form part of the documentation as well. The design documentation should make it easier to extend the functionality of the log analysis tools. There are a lot of reusable functions. These will be useful when the other machine types like the *Spectron* machines (used to sort smaller mail pieces) are added to the unified logging system.

Design and implementation of user help and user manual

The user guide was developed as per "User Guide" appendix. An html version was also developed and deployed on the web server so as to have instant access. The user guide was deliberately made easier to allow users with limited computing knowledge to easily follow the instructions. The help guide consists of several graphical illustrations that users can relate to whilst learning how to use the system. The html version of the user guide consists of an easy to use navigation structure. A copy of the html guide is on the compact disc accompanying this project.

Cache data on web server to eliminate network latency

The web server from which the log files tools run from is not on the same computer used to deposit the log files each day. The log files are now cached on the GNU/Linux web server as compared to being accessed from the log file server. A typical search can include 30 files each of about 3 megabytes totally to about 90 megabytes. This amount of traffic can have a big impact on the time it takes to return search results. By meeting this objective, the "**Improvements in data store to improve web server response times**" objective was also met.

Capability to directly request a graph for a particular date using the trend form

The *fmtrend* tool can accept files from any date and is therefore able to perform a trend for any date. As per the user guide, all the user has to do is to select the date field from the drop down list. Getting a trend graph is useful for a couple of situations especially when trying to predict future problems.

Upgrade Web Server Speed to run analysis scripts faster

The web server from which the unified logging system tools run from was successfully migrated from a modest Intel Pentium 75MHz processor based computer with 128Mb of RAM to a more powerful computer with an Intel Pentium 2.8GHz processor and 1Gb of RAM running the GNU/Linux operating system. When performing queries which involve large amounts of log files and multiple search strings a lot of RAM improves performance. All the files from the old web server were successfully copied to the new system and the web server response time was very significantly improved. More users can now perform complex queries at the same time without a significant performance degradation.

Include new analysis methods including having log reports by criticality

Events can now be analysed by criticality by selecting the criticality field from the drop down menu as per user guide instructions. Users can also define the critical events they are interested in and put them in a file called a watch list and have the system email them reports based on their watch lists. There is also a quick link for obtaining the top 10 or top 20 critical events for any day. A machine comparison analysis can also be performed on these critical events.

Implement email reports

Users are now getting reports mailed to them. The user guide contains instructions explaining how to request these reports. The figure below shows a sample email message generated by the email reporting tool.

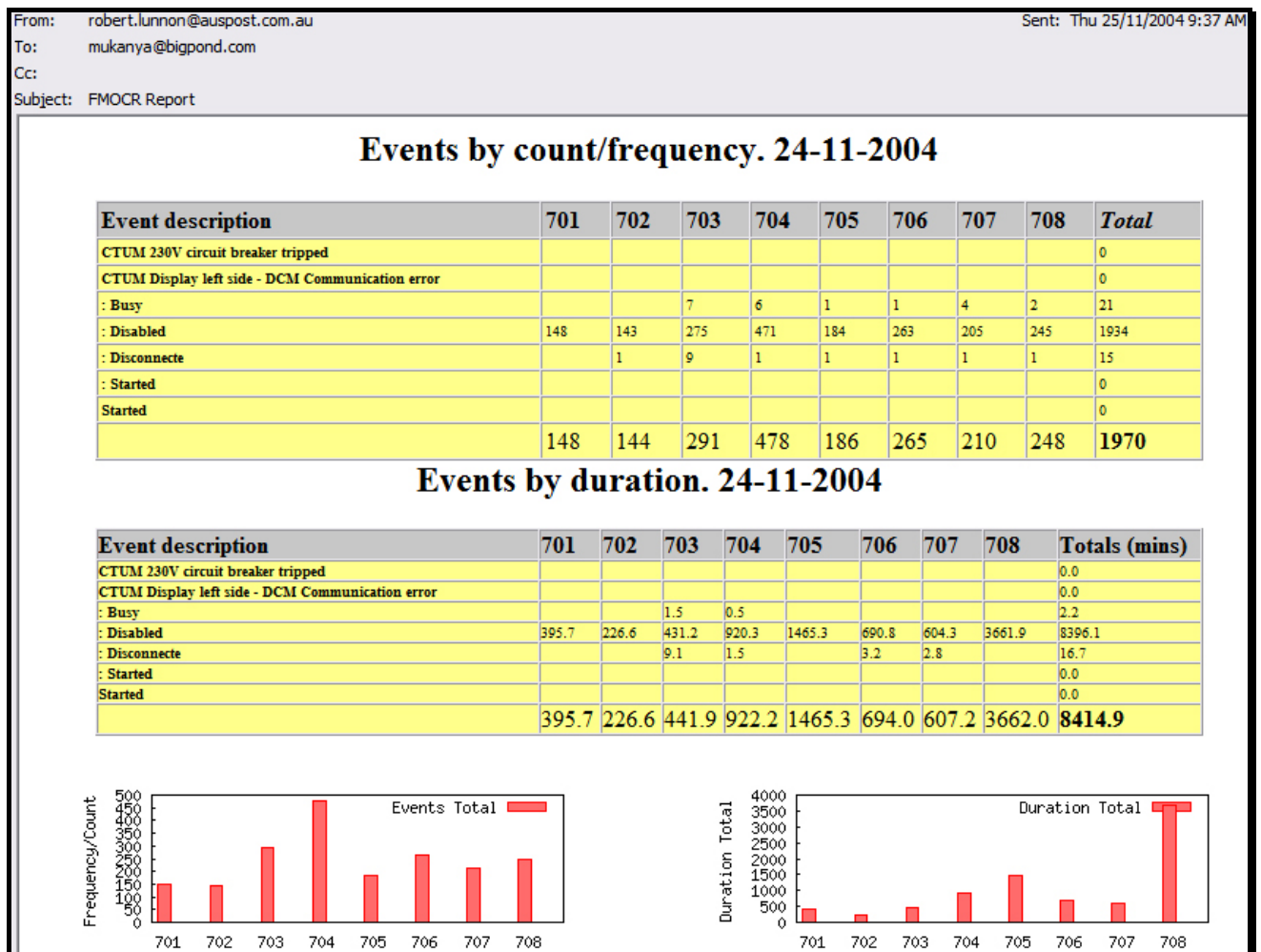


Figure 3 - Sample email report

In the figure above the user had defined the events they are interested in, as per the event description column, and put them in a watch list. The email report consists of a

two tables, one for the events frequency and other for the events durations totals, and the corresponding machine comparison graphs.

Implement machine comparison analysis

Machine by machine comparison is now possible. The following figure shows a typical result.

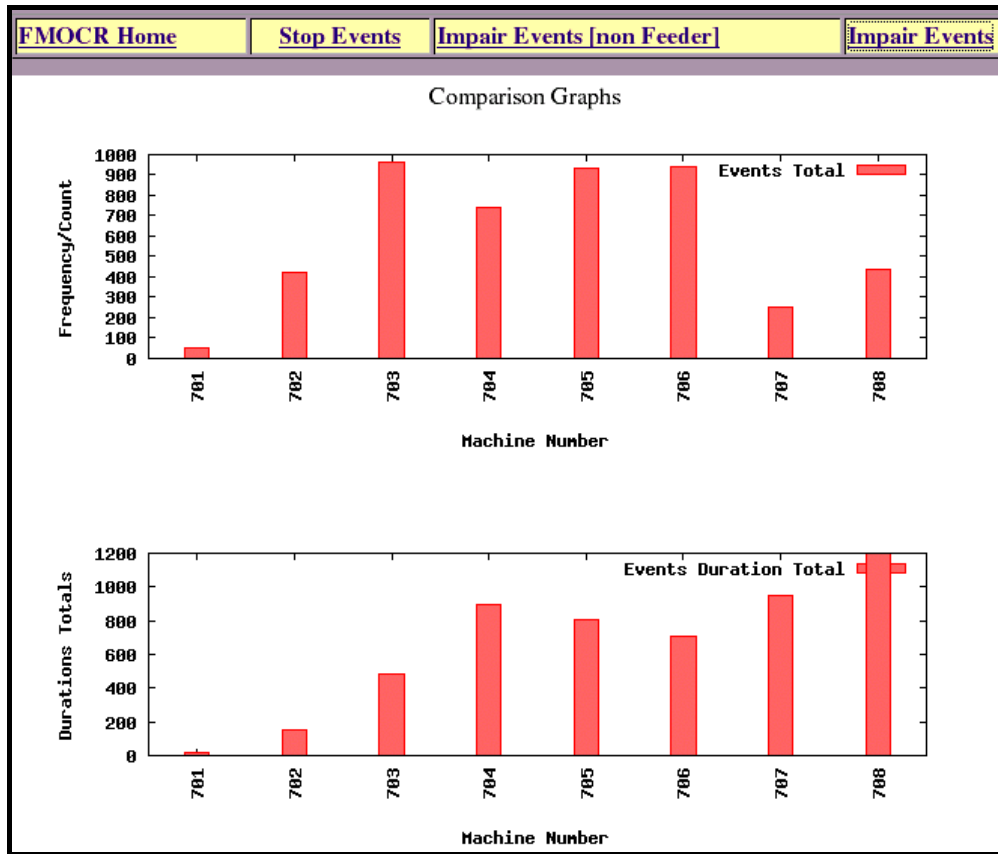


Figure 4 - Machine Comparison analysis

The above graphs show a comparison of each of the mail sorting machines by the number of errors and also by the duration of the errors. For example, a quick look at graphs reveals that machine 701 was the best performer on this day. It can also be seen that although machine 708 had relatively few errors, its duration totals' are the highest. The user guide has further information about these comparisons. The tools also show the figures used to generate this report.

9 Lessons Learned

When I started this project I had a pretty good idea of what to expect but I did not expect it to be this entertaining. The most fascinating thing was to realise that my solution was improving mail delivery throughout Australia. I learned how to implement an industry project from getting the specifications through to implementing the solution. I encountered a lot of problems, particularly the way specifications are always changing, and I learned how to adapt.

I learned how to improvise when computing resources are limited. I especially learned that a computing solution does not always need more powerful hardware to be thrown at the problem but rather more efficient algorithms and planning can achieve the same if not better results.

I improved my technical skills. I can now use C/C++ to solve practical problems better than I did when I commenced the project. I also improved my knowledge of GNU/Linux and open source software as the project solution was implemented on GNU/Linux and utilised open software utilities like *Gnuplot*. I also learned how to be both an active team member and also work independently.

9.1 Problems faced

The main problems faced were the fact that the project had to be completed in short period of time and I had to work on a production server. The limitation meant that I could not do as much testing of the tools I was developing as I wanted and it also meant that I could not make the user interface as presentable as I wanted. Working on the production server instead of a development computer meant that I had to be extra careful which I meant I could not experiment with other implementations freely since any mistake could have taken the server offline. Finally a minor problem I faced is the lack of Internet connectivity I had during the day (at night I had connectivity at home) which limited my research capability especially the inability to use Internet search engines to solve small problems.

10 Future improvements

There are a couple of areas that have potential for improvement. Firstly, I recommend moving the log file information to a relational database. This would result in a small performance overhead (it is quicker to read the log files directly than through a database) but the advantages include having an easy backup system and the ability to do advanced queries since relational databases were designed for queries.

It would be useful to have the amount of mail that each mail sorting centre processes to be included when doing machine comparisons. At the moment the comparisons only tell you the number of faults and their total durations. To do a fair comparison of mail centres the total quantity of mail processed during that time should be included in the equation as well since some centres handle more mail than others. Finally, it would be helpful to have a technician select an error message and suggest how to solve the problem presented to them. This would require help from someone who has considerable experience with mail sorting machines.

11 Conclusion

Mail delivery is a complex and challenging business which is constantly meeting new challenges as the amount of mail to be delivered increases. Australia Post technicians and management can now use the valuable information from the mail sorting machine log files to improve mail delivery in Australia. Information about mail sorting machines' faults can now be used in an easy and intuitive way.

Management can now have a quick overview or detailed information about machine performances with the aim of improving mail delivery and cutting costs. I am confident that the work I have done will help Australia Post to continue and to improve its "on time mail delivery" policy.

12 Glossary

OCR: Optical Character Recognition.

FMOCR: Flat Mail Optical Character Reader

RAM: Random Access Memory.

Mb: Megabytes

GB: Gigabytes

CGI: Common Gateway Interface

More glossary terms are in appendices sections of this report.

13 References

- [1] [Rangachar Kasturi, Lawrence O'gorman, Venu Govindaraju], "Document image analysis: A primer" Vol. 27, Part 1, February 2002
- [2] Homayoon S.M. Beigi, "An overview of handwriting recognition", Proceedings of the World Congress on Automation, Montpellier, France, May. 27-30, 1996.
<http://www.internetserver.com/~beigi/homayoon/conference.html>
- [3] Giovanni Garibotto, "Computer Vision in Postal Automation"
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ECVNET/Postal.Automation.html
- [4] "History of the U.S. Postal Service, 1775-1993 Postal Mechanization/Early Automation" http://www.usps.com/history/his3_5.htm
- [5] "Optical character recognition"
http://www.webopedia.com/TERM/o/optical_character_recognition.html
- [6] "OCR"
http://searchsmallbizit.techtarget.com/sDefinition/0,,sid44_gci214132,00.html
- [7] "OCR Basics"
<http://www.ocr-systeme.de/englisch/ocrallg.htm>
- [8] "MQC - Chapter 7 Non-automation Mailings"
http://pe.usps.gov/mpdesign/misc_docs/mqcHTML/mqc_7.htm
- [9] "Designing Letter and Reply Mail"
<http://pe.usps.gov/text/Pub25/Pub25ch1.htm>

14 Appendix A – User guide

1.0 Introduction	33
2.0 Summary of what can be done	33
3.0 The RIS web site	33
4.0 FMOCR Trends	34
4.1 How to get a trend?	34
4.1.1 Select machine number	34
4.1.2 Select trend period.....	35
4.1.3 Select plot interval.....	35
4.2 Single query trend	35
4.2.1 Event Text trend	36
4.2.2 Event Type trend	36
4.2.3 Query by module.....	36
4.2.4 Query by module ID.....	37
4.2.5 Event Duration trend	37
4.2.6 Event criticality trend	38
4.2.7 Impairment trend	38
4.2.8 Event time trend	39
4.3 Watch list query	40
4.4 Points to note.....	40
4.5 Multiple queries trend	40
4.5.1 Examples	41
4.6 Sample trend graphs.....	42
5.0 FMOCR Log Report	44
5.1 Web Reports.....	44
5.1.1 Top 10/20 Report	44
5.1.2 Watchlist Report for Yesterday or any day.....	45
6.0 Watch lists	47
6.1 Registering for watch lists.....	47
6.2 Creating or editing watch lists.....	48
7.0 Email Reports	52
7.1 Email reports – Top Events	52
7.2 Email reports – Watchlist Comparison	53
7.3 Email reports – Watchlist Graph.....	53
8.0 Daily FMOCR Reports - Histograms	54
9.0 Glossary	58
2.1 Example FMOCR log entry	60
6.1 Fmtrend Usage summary:	62
Makefiles	80
Testing	81
Glossary	81

1.0 Introduction

Each day the Network Engineering Unit (NEU) collects the HMI log files from all FMOCR machines nationally and processes them into an Excel compatible form suitable for undertaking analysis work. The system has been recently enhanced to provide simple but very powerful analysis work including automatic daily emails, trend graphs, top 10 or top 20 events for any day, machine to machine comparison histograms. The interface is deliberately very simple and is accessible for any web browser within the Australia Post Intranet.

2.0 Summary of what can be done

- Set up a watch list of events that are of interest to you and have a report emailed to you daily
- Get trend graphs based on your chosen search criteria including criticality, event text, impairment or durations
- View daily summaries of stop events, impair events (non feeder) and impair events (feeder) from the FMOCR web site for all the machines including comparison histograms
- View yesterday's (or any day for that matter) top 10 or 20 stop, warning and impair events for any machine or all the machines. You can also choose to sort events by frequency or by duration. You can also use watch lists you have set up as your search criteria instead of stop, warning or impair.

3.0 The RIS web site

The Reliability Improvement Section (RIS) web site found at <http://dpu/neu/ris/RISIndex.html> is your first point of contact with the FMOCR log tools. From that page select the FMOCR link as shown by the snapshot below in figure 1.

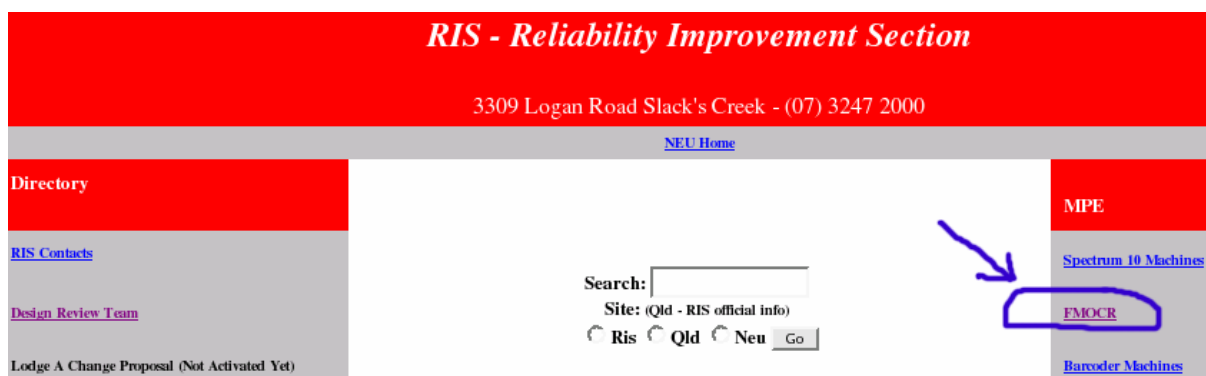


Figure 5 - RIS main web page snapshot

After selecting the FMOCR link you will be presented with further options, frequently asked questions, FMOCR log reporting, FMOCR trend graph, Daily bar graph comparisons, FMOCR documentation and Information. The figure below shows a snapshot of the FMOCR web page.

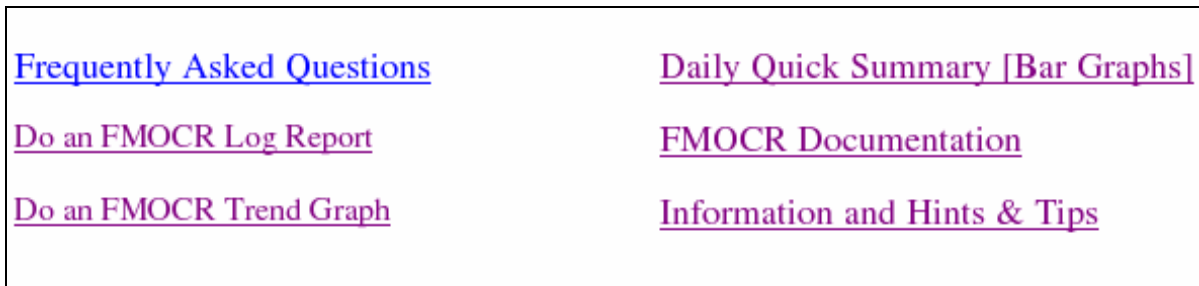


Figure 6 - FMOCR main web page snap shot

The rest of this document illustrates how to use each of the options described above.

4.0 FMOCR Trends

The FMOCR trend tool allows you get a quick snap shot of a machine's activity from today going back to a specified period of time. You get a graph illustrating events per hour or per minute, together with the list of the actual events which happened during that time. You also get to know the durations of those events.

4.1 How to get a trend?

4.1.1 Select machine number

The first step in getting a trend is to select the machine you want the trend for. Currently the machine numbers are 701, 702, 703, 704, 705, 706, 707 & 708 respectively. Another option is to include all the machines in the trend by selecting the option "ALL". Figure 3 shows the typical form you will see on the FMOCR web site. In this case, to select the machine number just click on "701" and you will be presented with further options.

 A screenshot of the FMOCR trend form. The form is set against a light yellow background and contains the following fields:

- Machine Number:** A dropdown menu with "701" selected.
- Period:** A dropdown menu with "DAY" selected.
- Plot Interval:** A dropdown menu with "DAILY" selected.
- Field:** Three identical dropdown menus, each with "7:Event Text" selected.
- Search String:** Three text input fields, each containing "NONE".
- User Name:** A text input field containing "global".
- Watch List:** A text input field containing "NONE".

 Below the fields, there is a text label "Click submit to generate" and a "SUBMIT" button.

Figure 7 - FMOCR trend form

4.1.2 Select trend period

After selecting the machine number you then have to select the trend period. You can do a trend for a single day (yesterday's trend), week (last seven days), fortnight (last 14 days), two minutes, month (last 30 days) or "ALL". "ALL" means you want to include all the available files. Depending on the number of files currently on the server this could be as far back as 3 months. As per Figure H1 above click on "DAY" and you will be presented with all the available options

Please be patient when you select the options "MONTH" and "ALL" since the program will have to read between 28 and 20 000 large files.

4.1.3 Select plot interval

The plot interval can either be "DAY" or "AUTO". The plot interval allows you specify how frequent you want to see the changes on the graph. For example, if you had previously selected a period of "MONTH" the plot interval of "DAY" will allow you to see the machines performance per each day more clearly. Leaving the default value of "AUTO" is usually sufficient. For obvious reasons, using a plot interval of "DAY" when you had previously selected a trend period of "DAY" will result in the system defaulting to "AUTO", which is a two minutes interval in this case.

4.2 Single query trend

The program allows you to do a simple search by first selecting the search field and then typing in your query. For example, to search for all events involving "Jam" simply select field "7: Event Text" from the drop down box as shown on Figure H1 above. You can even include a phrase as in "Shuttle Jam". Also see the section on full phrases later in this document to find out more. You can do simple things like finding all events which exceed a certain duration by first selecting "12: Event Duration" and then typing in "10+" in the search string field, for example to find all events which exceed 10 seconds in duration. Table 1 below shows a summary of the search criteria. Each search field will be discussed further in the later section of this document.

Search field	Description	Example search string
1:Date	The date the event happened	01-11-2004
2:Time	Time interval for the events you are after	2:30-20:23
3:UID	Query by event unique identifier	715387989
4:Machine Type	Distinguishes FMOCR from other machines (currently not necessary)	7
5:Machine ID	Same as Machine Number. Not necessary at this stage	701
6:Event Type	Common event grouping	Jam
7:Event Text	Description of an event	Shuttle Jam
10:Module	Module name	Stacker Feeder
11:Module ID	Module Instance ID	3
12:Event Duration	Duration of event in seconds	10+ or 156- or 23:88
13:Event Criticality	Indicator of criticality of event	Impair
14:Impairment	Estimate of machine impairment as a percentage	33

Table 1 - Possible search criteria summary (shaded fields are currently not necessary)

4.2.1 Event Text trend

An "Event Text" is a machine specific text relating to an event. Examples include "BCP1: Image-Printer sends unknown data 1;0" and "ETS-IU4: Conveyor light barrier error: conveyor segment 1". To query by event text you can type in a single keyword from the event text or to get even more accurate results you could type in the whole phrase as in "Shuttle Jam" instead of just "Shuttle". The figure below illustrates this procedure.

Machine Number: 701 Period: DAY Plot Interval: AUTO

Field: 7: Event Text Search String: jam feeder

Field: 7: Event Text Search String: NONE

Field: 7: Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 8 - Query by Event Text

4.2.2 Event Type trend

An event type is a grouping of common events as in "Jam", "Timing" or "Power". Make sure you select "6: Event Type" from the search field and then type in the event type you are after in the search string field. Figure 5 below shows an example.

Machine Number: 708 Period: MONTH Plot Interval: DAILY

Field: 6: Event Type Search String: Timing

Field: 7: Event Text Search String: NONE

Field: 7: Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 9 - Query by Event Type

4.2.3 Query by module

This is for use with module names like "Reader", "BarcodePrinter", "TLM" or "ACS-Shuttle". The query procedure is similar to the ones already discussed above. Make sure you select "10: Module" from the search field first and then type in the module you are after in the search string field. Figure 6 below shows an example.

Machine Number: 708 ▾ Period: DAY ▾ Plot Interval: AUTO ▾

Field: 10:Module ▾ Search String: TLM

Field: 7:Event Text ▾ Search String: NONE

Field: 7:Event Text ▾ Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 10 - Query by Module

4.2.4 Query by module ID

This is a number representing the instance of a particular module. For example, a module called TLM might event instances 1, 2 and 3 as in TLM 1, TLM 2 and TLM 3 respectively. Make sure you select "11: Module ID" from the search field and then type in the module ID you are after in the search string field. Figure 7 below shows an example.

Machine Number: 701 ▾ Period: DAY ▾ Plot Interval: DAILY ▾

Field: 7:Event Text ▾ Search String: NONE

Field: 11:Module ID ▾ Search String: 3

Field: 7:Event Text ▾ Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 11 - Query by Module ID

4.2.5 Event Duration trend

This is the event duration in seconds. You can search for events which exceeded a certain time, were shorter than a certain time or between certain intervals. To search for all events that exceeded 10 seconds (inclusive) you simply type **10+** in the query string field. Similarly, for all events which were shorter than 48 seconds you simply type in **48-** (inclusive). Finally to find all events whose duration was between 10 and 48 seconds inclusive simply select field "12: Event Duration" and type in **10:48** in the search string as shown in Figure 8 below.

Machine Number: 701 Period: DAY Plot Interval: DAILY

Field: 7:Event Text Search String: NONE

Field: 12:Event Duration Search String: 10:48

Field: 7:Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 12 - Query by duration

4.2.6 Event criticality trend

This is a textual indicator of the criticality of the event as in "Safety", "Impair", "Warning" or "Stop" just to name a few. Select the field "Event Critic." Type in the criticality you are after in the search string text box. Figure 9 below shows an example.

Machine Number: 701 Period: DAY Plot Interval: DAILY

Field: 7:Event Text Search String: NONE

Field: 13:Event Critic. Search String: Stop

Field: 7:Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 13 - Query by criticality

4.2.7 Impairment trend

"Impairment" is an estimate of how this event impaired machine performance and is expressed as a percentage. Select the last field from the drop down list, "14: Impairment" and type in the required value in the search string. To do a trend of events with an impairment of say, 33% see Figure 10 below.

Machine Number: 701 Period: DAY Plot Interval: AUTO

Field: 7:Event Text Search String: NONE

Field: 14:Impairment Search String: 33

Field: 7:Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 14 - Query by impairment

4.2.8 Event time trend

This query enables you to find events which happened between certain times. For example, it might be useful to find all events that happened between 9pm and 11pm. This trend is even more useful when used in conjunction with other queries (see the section "Multiple queries" for more information). The query format is **<start time>-<end time>** where start or end time is in the format **hour:minute**. To find events between 9pm and 11pm you type **21:00-23:00** Figure 11 below illustrates how to do a time trend.

Machine Number: 701 Period: DAY Plot Interval: AUTO

Field: 7:Event Text Search String: NONE

Field: 2:Time Search String: 21:00-23:00

Field: 7:Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 15 - Query by time

4.3 Watch list query

The last single query trend describes the watch list trend. A watch list is a collection of events you are interested in. See the section "Watch lists" for information on how to register yourself for watch lists and how to create a watch list. To do a watch list trend enter your username and the name of the watch list. To illustrate how this works lets have an example. A user with username called "Anthony" has already registered and has a watch list called "mywatches" which contains the following contents:-

ctum
shuttle Jam
impair

All they have to do is enter "Anthony" in the username and "mywatches" in the watch list section. What will happen is that those three items in their watch list (ctum, shuttle jam & impair) will be included in their trend. Figure 12 shows how to accomplish this.

Machine Number: 701 Period: DAY Plot Interval: AUTO

Field: 7:Event Text Search String: NONE

Field: 2:Time Search String: NONE

Field: 7:Event Text Search String: NONE

User Name: anthony Watch List: mywatches

Click submit to generate

SUBMIT

Figure 16 - Query by watch list

4.4 Points to note

- ✓ You can use spaces in your search strings (i.e. short phrases like "Shuttle Jam").
- ✓ You can use any of the search field/search string combinations from the three shown on the form (you don't necessarily have to type in the top one)
- ✓ Leave the fields you are not using as "NONE" or empty. Anything else will be included in your query and may give incorrect results
- ✓ Time queries should be in 24HR notation

4.5 Multiple queries trend

Sometimes you may not only want to find certain events but also within a certain duration or impairment. Multiple queries allow you such flexibility. The form will allow you to do up to three combinations at a time, in addition to the watch list query. To illustrate how this works lets do a couple of examples.

4.5.1 Examples

Suppose you need to find all "Shuttle Jam" related events that happened between 1 am and 3 am. Type in "Shuttle Jam" in the first search string and 1:00-3:00 in the second search string and remember to select the appropriate search field as we did before with the single search trends. Figure 13 below illustrates how to do this.

Machine Number: 701 Period: DAY Plot Interval: DAILY

Field: 7:Event Text Search String: Shuttle Jam

Field: 2:Time Search String: 1:00-3:00

Field: 7:Event Text Search String: NONE

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 17 - Multi-field search [2 fields]

Continuing with the example above, let's say we also want to include events which had duration above 100 seconds, we simply complete the third field and again remember to select the appropriate search field. See Figure 14 below.

Machine Number: 701 Period: DAY Plot Interval: DAILY

Field: 7:Event Text Search String: Shuttle Jam

Field: 2:Time Search String: 1:00-3:00

Field: 12:Event Duration Search String: 100+

User Name: global Watch List: NONE

Click submit to generate

SUBMIT

Figure 18 - Multi-field search [3 fields]

Finally, let's also include a watch list in our search. If you have a valid username and watch list (see the section on Watch lists if you haven't registered) just type in your username and required watch list. See Figure 15 below.

Machine Number: Period: Plot Interval:

Field: Search String:

Field: Search String:

Field: Search String:

User Name: Watch List:

Click submit to generate

Figure 19 - Multi-field search [4 fields]

4.6 Sample trend graphs

The following screenshot shows the result after searching for events whose event type is "Timing". All the three screenshots came from one web page.

FMOCR Trend for 708

[Back](#)

Updating Log Files...

Found 1307 matches in 31 data file(s)

Total of Event Durations = 26:47:29

Generated by fmocrtrend - Reliability Improvement Section- NEU

[Generated \[24 Nov 2004 13:31\]](#)

Figure 20 - Sample trend output (Page top)

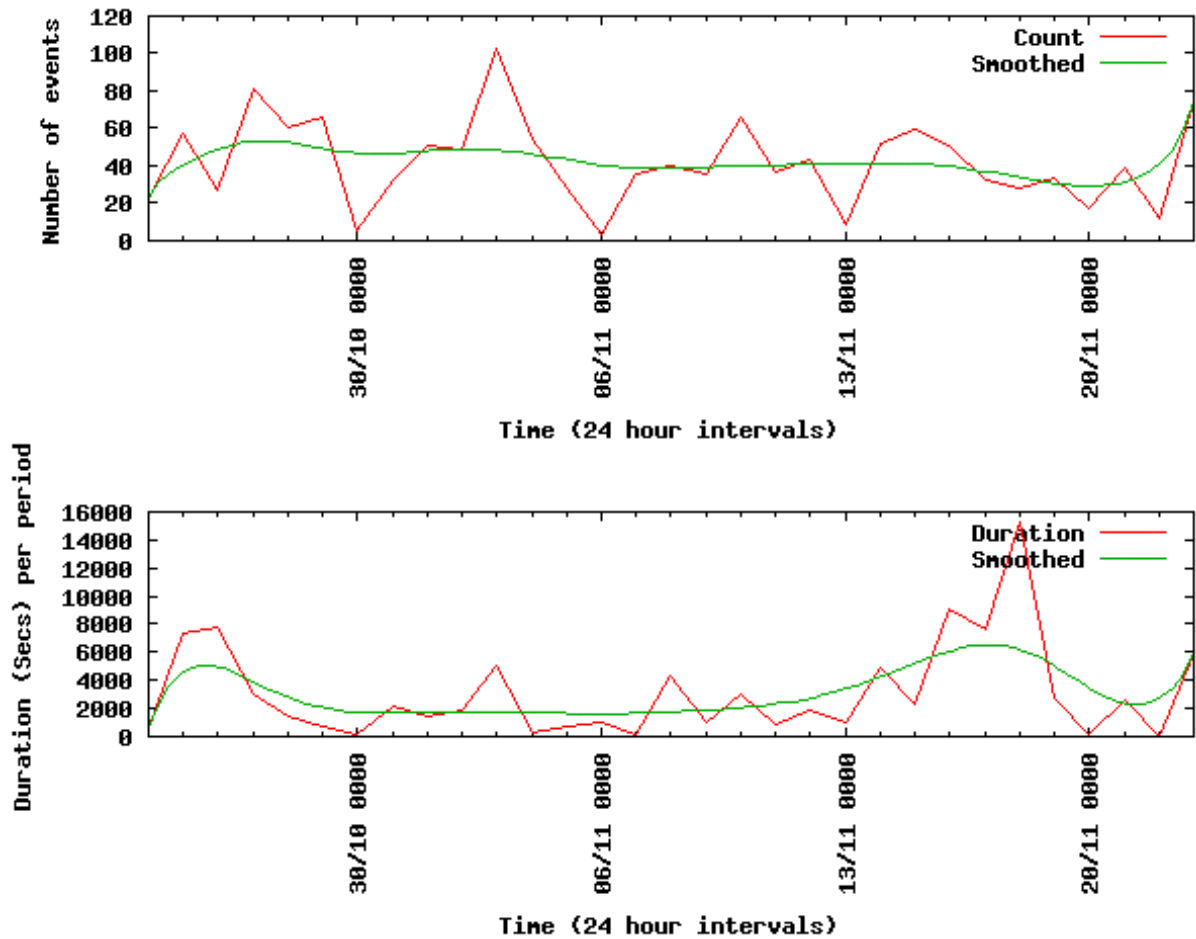


Figure 21 - Sample trend output (Page middle)

The trend report will tell you how many files when used to generate the report (31 for thirty one days in this case), the number of matches and the event durations total. There are two trend graphs. The first graph shows the number of per given time period and other shows the durations per period. Within each graph there is the count/duration graph (in red) and the other one which shows a moving average (in green). Finally the report lists all the unique events which where used to generate the graph as shown below.

- Including Event: CTUM: Timeout gripper cycle**
- Including Event: CTUM: Timeout at tray roller conveyor 1**
- Including Event: Shuttle 2: SCT fault; Full cartridge not in time inside exit gate at position 320**
- Including Event: Shuttle 2: SCT fault; Full cartridge not in time inside exit gate at position 151**
- Including Event: CTUM: Timeout tray input**
- Including Event: IDS: Encoder 2 fault**
- Including Event: Shuttle 1: SCT fault; Full cartridge not in time inside exit gate at position 420**

Figure 22 - Sample trend output (Page bottom)

5.0 FMOCR Log Report

There are two ways to get FMOCR log reports. The first is by using the FMOCR web page and the other by creating watch lists so you have the reports emailed to you daily in the morning. We will look at both in detail in the following sections.

5.1 Web Reports

From the FMOCR web page select "Do an FMOCR Log Report" as shown below in the figure below.

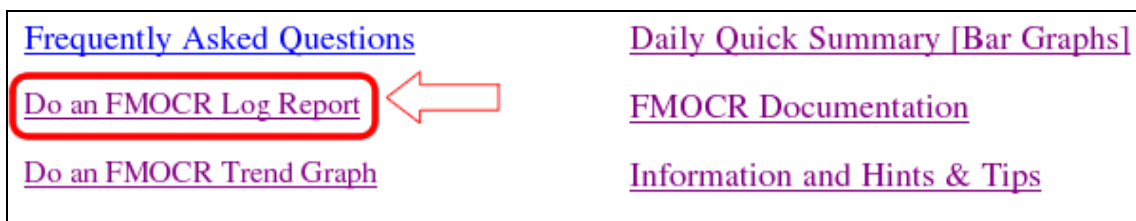


Figure 23 - Do an FMOCR Log Report

After selecting the "Do an FMOCR Log Report" option you will be presented with further options which are "Top 10 Reports (yesterday)", "Top 20 Reports (yesterday)", "Quick Watch List Report Yesterday" and "Watch List Report For Any Day". If you don't have watch lists already defined the top of the page has a link to the watch list editing page. More information on creating watch lists is defined later in this document. The figure below shows a snapshot of the reports page.

<p>Top 10 Report (Yesterday) Stop Impair Warning</p> <p>Top 20 Report (Yesterday) Stop Impair Warning</p> <p>Convert Upload Data Now</p> <p>Machine No: <input type="text" value="701"/> <input type="button" value="Start"/></p>	<p>Quick Watch List Report for Yesterday</p> <p>Machine No <input type="text" value="701"/></p> <p>Sort by <input type="text" value="Count"/></p> <p>Username <input type="text" value="global"/></p> <p>List Name <input type="text" value="ALL"/> <input type="button" value="SUBMIT"/></p>	<p>Watch List Report for Any Date</p> <p>Machine <input type="text" value="707"/></p> <p>Sort by: <input type="text" value="Count"/></p> <p>Day: <input type="text" value="10"/></p> <p>Month: <input type="text" value="11"/></p> <p>Year: <input type="text" value="2004"/></p> <p>Username: <input type="text" value="global"/></p> <p>List Name <input type="text" value="ALL"/> <input type="button" value="SUBMIT"/></p>
--	--	---

Figure 24 - FMOCR Log Reporting

5.1.1 Top 10/20 Report

These options allow you to view yesterday's top 10 or 20 events. You can select stop, impair or warning events. See the glossary section of this guide for a better description of what we mean by "Stop", "Impair" or "Warning" events. The figure below shows typical results from such reports.

Log report for 14/11/2004, machine ALL sorted by "Count"

Shortcuts to Sections									
Stop	Impair	Safety	Interlock	Warnings	Information	Unknown	Unclassified		
Impair		Count / Total duration (min) / Ave duration (secs)							
Event	Count	701	702	703	704	705	706	707	708
IU2: Jam Feeder	799	6 1.77' 17.71"	176 54.98' 18.74"	112 45.46' 24.35"	84 32.30' 23.07"	91 53.69' 35.40"	229 105.52' 27.65"	49 35.36' 43.29"	52 30.75' 35.48"
IU3: Jam Feeder	622	8 1.38' 10.36"	42 24.94' 35.63"	111 43.25' 23.38"	126 48.05' 22.88"	136 75.25' 33.20"	106 43.13' 24.41"	27 27.41' 60.91"	66 34.04' 30.95"
IU4: Jam Feeder	421			82 35.26' 25.80"	73 42.79' 35.17"	72 36.34' 30.28"	123 47.60' 23.22"	10 6.84' 41.05"	61 21.18' 20.83"
IU3: Jam at Pusher	342	7 2.32' 19.88"	18 5.11' 17.04"	70 19.66' 16.85"	20 26.14' 78.42"	150 133.73' 53.49"	37 7.46' 12.10"	13 8.98' 41.45"	27 19.30' 42.88"

Figure 25 - Top 10/20 Report for yesterday

Here the results are sorted by count and within each table cell there are other important statistics as well. Numbers which are green coloured represent the count for that event and for that machine whilst those which are brown coloured represent total durations, in minutes, for that event and for that particular machine. Finally, the purple coloured numbers indicate the average durations, in seconds, for those events.

5.1.2 Watchlist Report for Yesterday or any day

If you have already defined watch lists you can have a quick report for yesterday or any day for that matter. If you do not have a watch list just leave the username as "global" and watchlist as "ALL". The figure below shows how to complete the form for yesterday's report whilst figure 22 shows how to complete the form for yesterday.


<p>Top 10 Report (Yesterday)</p> <p>Stop Impair Warning</p> <p>Top 20 Report (Yesterday)</p> <p>Stop Impair Warning</p> <p>Convert Upload Data Now</p> <p>Machine No: <input type="text" value="701"/> <input type="button" value="Start"/></p>	<p>Quick Watch List Report for Yesterday</p> <p></p> <p>Machine No <input type="text" value="701"/></p> <p>Sort by <input type="text" value="Count"/></p> <p>Username <input type="text" value="global"/></p> <p>List Name <input type="text" value="ALL"/> <input type="button" value="SUBMIT"/></p>	<p>Watch List Report for Any Date</p> <p>Machine <input type="text" value="707"/></p> <p>Sort by: <input type="text" value="Count"/></p> <p>Day: <input type="text" value="10"/></p> <p>Month: <input type="text" value="11"/></p> <p>Year: <input type="text" value="2004"/></p> <p>Username: <input type="text" value="global"/></p> <p>List Name <input type="text" value="ALL"/> <input type="button" value="SUBMIT"/></p>
--	---	---

Figure 26 - Quick watch list report for yesterday

For a watch list report for yesterday select the machine number or "ALL" to include all machines in your report. The next step is to choose the sort type. You can have your

results sorted by duration or by frequency or if you prefer you can retrieve the raw data so you can use a spreadsheet program to sort the results yourself.

Figure 27 - Quick watch list report for any day

For a watch list report for any day the procedure is similar to that for yesterday's report except that you also have to choose a day, month and year.

The figure below shows sample results.

Log report for 10/11/2004, machine 707 sorted by "Count"

Shortcuts to Sections							
Stop	Impair	Safety	Interlock	Warnings	Information	Unknown	Unclassified
Impair							
More Info	Event Description		Count	Duration			
	IU3: Jam Feeder		250	01:52:21.776			
	IU2: Jam Feeder		235	01:47:50.222			
	IU4: Jam Feeder		134	00:59:08.785			
	Shuttle 2: Gripper vertical drive motor fault: Error0		80	01:40:33.644			
	Safety relay of shuttle 1 has tripped		79	02:14:47.439			
	Shuttle 1: Gripper command ERROR ACKNOWLEDGE failed		78	13:46:47.390			
	Shuttle 1: Gripper horizontal drive motor fault: Error 40		73	12:31:34.542			

Figure 28 - Quick watch list report for any day sample results

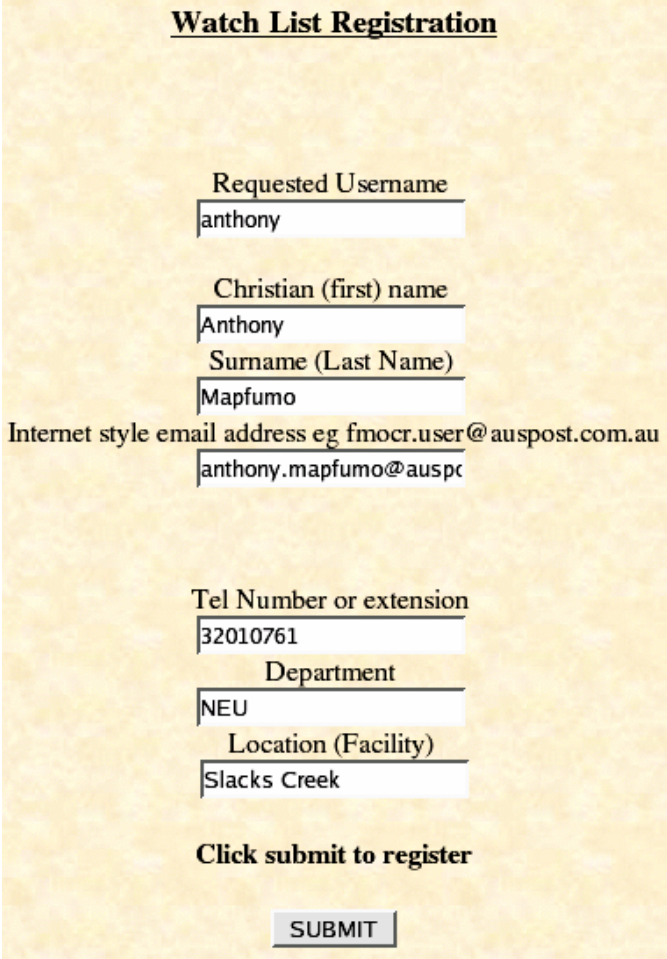
You can view sections of the report according to the links on top of the report ("stop", "Impair", "Safety", etc) in the yellow cells. From this report we can conclude that the most impairing event was "IU3: Jam Feeder" which had a count of 250 and a duration of one hour and 52 minutes.

6.0 Watch lists

Watch lists are groups of events which are of particular interest to you. There are many FMOCR events so it is logical for a user to select only those they require. This gives the user more useful information instead of getting a report with everything resulting in you having to spend a considerable amount of time getting the information you want. For example, you might be interested in "Stop events" only. All you have to do is create a watch list consisting of stop events. Once such a watchlist list is created you could have a report based on that watchlist or trend graphs based on your watchlist as per previous sections emailed to you daily. The first thing you need to do is to register your details.

6.1 Registering for watch lists

Registering for watch lists is simple and straight forward. You will need a username, first name, last name, email address (for emailing daily reports), telephone number or extension, department and location. The figure below shows a simple form to be completed for watch list registration.



The image shows a registration form titled "Watch List Registration" on a light yellow background. The form contains several input fields with labels above them. The labels are: "Requested Username", "Christian (first) name", "Surname (Last Name)", "Internet style email address eg fmocr.user@auspost.com.au", "Tel Number or extension", "Department", and "Location (Facility)". The input fields contain the following text: "anthony", "Anthony", "Mapfumo", "anthony.mapfumo@auspc", "32010761", "NEU", and "Slacks Creek". Below the form, there is a bold instruction "Click submit to register" and a grey button labeled "SUBMIT".

Figure 29 - Watch list registration

After clicking the submit button the registration process will be complete. The next step will be to define the actual watch lists. These will be stored in your profile. Later on we will discuss how you can have some of these watch lists form the basis of reports which can be emailed to you daily. Figure 26 below shows the screenshot

immediately after clicking the submit button. It also shows links the watch list editing page as well as the FMOCR home page.



Figure 30 - Registration confirmation page

6.2 Creating or editing watch lists

To create or edit watch lists already created you have to log in first by entering your username. If you haven't registered the page gives you the option to do so. Figure 27 shows a snapshot of the login page.



Figure 31 - Watch list editing login page snapshot

After entering your username and clicking on the submit button you will be presented with the following screen shown in the figure below.

Watches for User anthony

Watch List *anthony*

List Name: Event:

User has no watchlists defined yet

List of All FMOCR Messages, Cut and Paste to add to your list

Note: Use Internet Explorers Edit->"Find on this page" (or press control F) to search for messages

```
$(#compId): Busy
$(#compId): Disabled
$(#compId): Disconnected
$(#compId): Display was updated
$(#compId): Enabled
$(#compId): General HW failure
$(#compId): I/O operation failed
$(#compId): Ignoring errors
$(#compId): Internal RAM defective
$(#compId): Not ready
```

Figure 32 - Watch list editing introduction page snapshot

From this page there are a couple of things you should note. The first is that the system creates a default watch list based on your user name. In this case, the username name is *anthony* and the initial watchlist is also called *anthony*. The second thing to note is that all possible events are listed in the scrollable text area on the page as shown in Figure H23 and they are sorted in alphabetical order. In the page above the first possible event you can include in your watchlist is “\$(#compId): Busy” followed by “\$(#compId): Disabled” and so on. To add events to a watchlist find the event you want to add, press the “Control and F” keys on your keyboard or click “Edit” on Internet Explorer menu and click on “Find”. The next step is to type in what you want to find. For example, typing in “ctum” in the search field will give you all ctum related events (press F3 to move to the next one). When you find an event you like copy it and paste it into the “Event:” field on the form and then click submit.

In the following example I am interested in “ctum” related events. I have created a watch list called “ctum” by filling in “ctum” in the “List Name:” field and then clicking the submit button. After that I copied and pasted the ctum events I am interested in.

The figure below shows the end result.

Watches for User anthony

Watch List *ctum*

List Name: Event:

Current list (Click Reload to refresh)

```
CTUM 230V circuit breaker tripped  
CTUM Display left side - DCM Communication error
```

List of All FMOCR Messages, Cut and Paste to add to your list

Note: Use Internet Explorers Edit->"Find on this page" (or press control F) to search for messages

```
$(#compId): Busy  
$(#compId): Disabled  
$(#compId): Disconnected  
$(#compId): Display was updated  
$(#compId): Enabled  
$(#compId): General HW failure  
$(#compId): I/O operation failed
```

Figure 33 - Creating a watchlist

Add as many events to the watchlist list as you want. Once you are happy with the contents exit the page or type in a new watchlist name in the "List Name:" field and repeat the process. The next figure shows the screen shot when I have two watch lists defined.

Watches for User anthony

Watch List *ctum*

List Name: Event:

Current list (Click Reload to refresh)

CTUM 230V circuit breaker tripped
CTUM Display left side - DCM Communication error

Watch List *stop_events*

List Name: Event:

Current list (Click Reload to refresh)

: Stopped
CTUM: Automatic stop

Figure 34 - Adding watch lists to your profile

In the above example I have two watch lists. One is called *ctum* and the other is called *stop_events*. The watch list *ctum* consists of two events I am interested in, "CTUM 230V circuit breaker tripped" and "CTUM Display left side – DCM Communication error". On the other hand the watch list called "stop_events" consists of events " : Stopped" and "CTUM: Automatic stop". You can define as many watch lists as you like with as many events as you like. In the next section we are going to discuss how you can have reports emailed to you daily based on your watch list.

7.0 Email Reports

We understand that you don't always have the time to visit the RIS website to perform queries and access trend graphs. We have designed the automatic email reporting system to help you in getting timely information.

To get daily email reports you first have to register for watch lists as described in the preceding sections.

The first step involves selecting the type of report you are after. The options are "Top Events", "Watchlist Comparison" and "Watchlist Graph". After selecting the type of report you are after enter your username. If you do not have a username you are given the option to register yourself for watch lists as described in the section "Registering for watch lists".

7.1 Email reports – Top Events

Please select the type of report to register

Top Events Watchlist Comparison Watchlist Graph

Top Events Report

Username (You must register [here](#) before you can use this feature)

How many of the top ranking items should be included

10

In what order

Count = Events occurring most frequently,
Duration = Events present for the longest total time

COUNT

At what level of criticality

Stop

Click Save to register your report

Save

Clear Reports

Warning:
This will clear all your saved reports

Username (Register [here](#) before using this feature)

Click Submit to clear all your registered reports

Do it

Send Report

Just send me an e-mail report right now

For User:

Go

Figure 35 - Email reports (Top events)

Select the number of top ranking items that should be included in the report and finally select the level of criticality. Remember to click on the "Send me a sample report by e-mail" check box and submit to finalise your registration.

7.2 Email reports – Watchlist Comparison

If you have already defined watch lists and you want reports emailed to you based on those lists then you should click on the “Watchlist Comparison” radio button. Enter your username and a valid watch list name. If you only want to get an email report when the total number of events exceed a certain threshold then enter that threshold just after the label which says “Only send me this report when the total number of exceeds”. In figure H27 we used 1000 as an example. We will only get a report if the number of events defined in the watch list called “stop_events” exceeds 1000.

You can also include an events total duration threshold as well. In our example, in figure H27 in addition to a counts threshold of 1000 we also need a report only if the events duration total exceeds 35 minutes. If you are not interested in thresholds just leave them blank or with zeros in them.

Please select the type of report to register

Top Events Watchlist Comparison Watchlist Graph

Watchlist Comparison Report

Username (You must register [here](#) before you can use this feature)

Use this Watchlist (You must define a watchlist to use this feature)

Only send me this report when the total number of events exceeds:
 Events
or when the total duration of the events exceeds:
 Minutes

Click Save to register your report

Clear Reports

Warning:
This will clear all your saved reports

Username (Register [here](#) before using this feature)

Click Submit to clear all your registered reports

Send Report

Just send me an e-mail report right now
For User:

Figure 36 - Email reports (watchlist comparison)

7.3 Email reports – Watchlist Graph

The history graph allows you to have a graphical view of the performance of a machine or machines over a specified time based on the criteria in watch list. The type of graphs is same as the ones you get when are doing trend queries as described before. Similar to other email reports, you will need a valid username and watch list in addition to selecting the machine you are after, graphing period (day, week, month etc) and the thresholds which will trigger the system to send you reports.

Top Events
 Watchlist Comparison
 Watchlist Graph

History Graph

Username (You must register [here](#) before you can use this feature)

Use this Watchlist (You must define a watchlist to use this feature)

What machine do you wish the report for:

Over what period would you like the graph:

Over what interval should I accumulate totals (Plot one point per ...)
 (Note By Minute will actually plot one point for every 2 Minutes)

Only send me this graph when the total number of events exceeds:
 Events

or when the total duration of the events exceeds:
 Minutes

Click Save to register your report

Clear Reports

Warning:

This will clear all your saved reports

Username (Register [here](#) before using this feature)

Click Submit to clear all your registered reports

Send Report

Just send me an e-mail report right now
 For User:

Figure 37 - Email reports (history graph)

8.0 Daily FMOCR Reports - Histograms

Each day we reports that automatically generated by the system and put on the FMOCR web page. Select the link "Daily Quick Summary [Bar Graphs]" as shown the snapshot below.

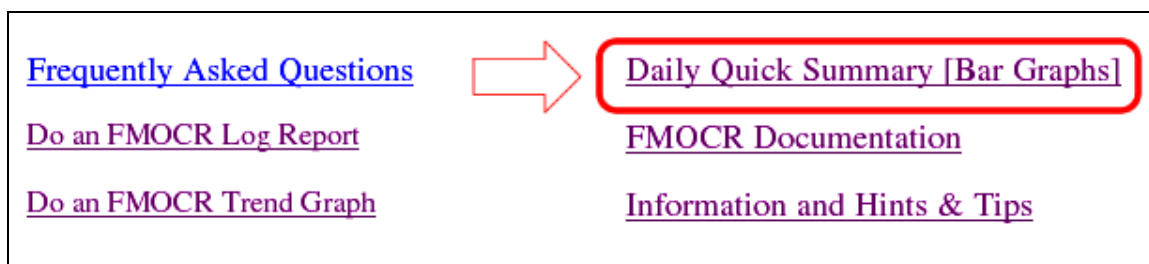


Figure 38 - Daily quick summary

Currently three reports are generated base on stop events, impair events (feeder) and impair events (non-feeder). Tables 2 to 4 lists the type of events listed in these watch lists.

Stop Events

Height monitoring 2 (left) triggered
Error at light barrier Thickness Measurement right
Error at light barrier Flat on Injector tube
Pocket not empty
Height Monitoring carrier triggered
Error at light barrier Thickness Measurement left
Finger Belt Feeding Section error: Light barrier
Height monitoring 1 (right) triggered
Compressed air pressure too low
Encoder 2 fault
Clock generator 1 fault
Image-Printer timed out in phase 3
Image-Printer: No inc
Groove box 865 missing
General optical link communication error
Optical link communication with IDS failed
Error at trigger light barrier OCR/Pre-Barcode-Read
Optical link communication with IU3 failed
Optical link communication with IP2 failed
Optical link communication with IU2 failed
Optical link communication with IU1 failed
Direct injector end switch (stop switch) triggered
JAM at timeout belt
Optical link communication with IP3 failed

Table 2 - Stop events included in the daily report**Impair events - feeder**

Jam Feeder
Jam at Pusher
Jam Diverter
MIU: Light Curtain Blocked
FLU Jam at Length Measurement Unit
Invalid Pos. Difference Laydown - Conveyer
Diverter Box almost full
MIU: Jam Security Flap / Security Flap open
Feeder in technical fault
Diverter Box full
Faulty Converter Lay Down Belt
Faulty Converter Side Belt
Position
Diverter Box Overflow / Jam Flap opened
Encoder Jump Transverse Pusher
Encoder Jump at MAIS (via Fiber Link)
Drag Difference Transverse Pusher
Jam in Conveyer Channel
Power Switch Transverse Pusher
Faulty Converter Conveyer Belt
Connecting Rod fracture
Grand Total

Table 3 - feeder impair events included in the daily report

Impair events – non feeder

Jam in transport conveyor 2
Gripper vertical drive motor fault: Error 0
Jam in transport conveyor 1
SCT fault; Full cartridge not in time inside exit
Transport stop by light curtain
Jam in tray label module no tray pickup
Internal tray queue error
Gripper command ERROR ACKNOWLEDGE failed
No tray on roller-conveyor in full field
CTUM to TLM tray transfer error
Cartridge inside diagonal light barrier - cartridge
Gripper vertical drive motor fault: Error 55
400V/AC circuit breaker tripped
Diverter (conveyor segments 13 - 18) full or block
Conveyor jam: segment 1
Label printer timeout
Gripper horizontal drive motor fault: Error 55
Entry stop gate left not closed at position
Entry stop gate left not open at position
Entry stop gate right not closed at position
Entry stop gate right not open at position
Tray stopper2: default pos. (up) not reached
Doors not locked
Main Status: Shuttle not ready
Tray stopper2: stop limit sensor down pos. Release
Gripper vertical drive motor fault: Error 40
Conveyor jam: segment 3
Gripper vertical drive motor fault: Error 57

Table 4 - non feeder impair events included in the daily report

The report lists the duration and frequency (count) of each event. It also lists the totals (frequency and durations) for each type of event for each machine, individually as well as for all the machines.

Above all the report generates a graphical comparison of machines in the form of a histogram thereby giving you a quick snapshot of the relative performances of all the Australia Post FMOCR machines on that day.

We will look at a couple of snapshots to illustrate how this report works.

Events by duration. 14-11-2004

Event description	701	702	703	704	705	706	707	708	Totals (mins)
Height monitoring 2 (left) triggered	0.4	10.4	4.2	12.9	3.5	18.2	3.9		53.6
Error at light barrier Thickness Measurement righ	0.7	1.2	11.1	6.7	12.9	21.9	4.5		59.1
Error at light barrier Flat on Injector tube	0.7	2.7	12.1	4.5	6.1	16.3	4.6	0.7	47.7
Pocket not empty									0.0
Height Monitoring carrier triggered		2.8		2.2	64.5	13.8		9.2	92.4
Error at light barrier Thickness Measurement left	0.7	2.0	4.1	3.4	14.7	20.4	0.2	0.7	46.3
Finger Belt Feeding Section error: Light barrier	1.9		4.0			0.7			6.6
Height monitoring 1 (right) triggered			1.1		2.0		1.7		4.8

Figure 39 - Stop events by duration snapshot

Optical link communication with IDS failed						0.5			0.5
Error at trigger light barrier OCR/Pre-Barcode-Rea									0.0
Optical link communication with IU3 failed						0.5			0.5
Optical link communication with IP2 failed						0.5			0.5
Optical link communication with IU2 failed						0.5			0.5
Optical link communication with IU1 failed						0.5			0.5
Direct injector end switch (stop switch) triggered							2.5		2.5
JAM at timeout belt	91.4	29.2	52.8	20.7	2.6	0.5	1.4		198.6
Optical link communication with IP3 failed						0.5			0.5
	171.3	74.5	81.1	308.0	64.7	74.7	50.0	100.3	924.5

Comparison Graphs

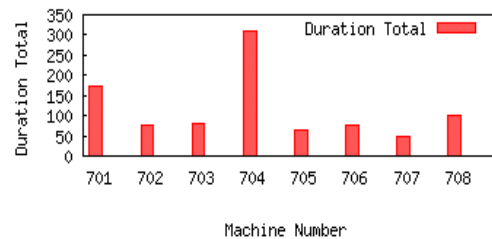
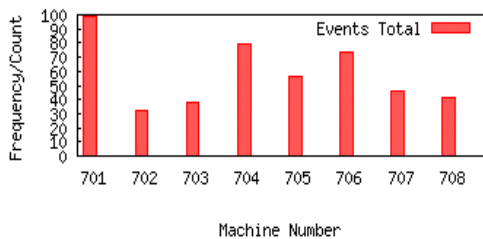


Figure 40 - Comparison Graphs

From the snapshots we can see that machine 702 performed better on the 14th of November 2004 as it had the smallest number of stop events hence the smallest durations total. This, of course, could be due to a number of factors including the amount of mail that machine processes per day.

We can also see that machine 707 had the least downtime whilst machine 704 was involved in a lot of interruptions.

9.0 Glossary

RIS – Reliability Improvement Section

NEU – Network Engineering Unit

Safety - Transport Stops, Emergency stops, open covers

Interlock - Other safety devices intended to protect the machinery rather than people

Stop - Events that will stop the machine immediately

Impair - Events that either impair the machines performance or will stop the machine after a delay

Warning - Events that indicate a failure is about to happen or that maintenance may be required in the near future.

Information – In the context of FMOCR event type it means “anything else”.

Count or Frequency means the same thing. This represents the number of times the event occurred in a particular period. For log reports the period is a day, on trend graphs the period is indicated at the bottom of the graph. When looking at a trend plot it is important to know the period since this will change the upper magnitude of the graph.

Duration means the duration of the event (The time the event is present). It should be noted that the event duration is not related necessarily to the downtime as downtime is strictly the time between the event start and the next machine start, where the event duration is the time from the start of the event to when it ends. There is always some time lapse from when a problem is cleared to when the machine starts producing again, so for events which stop the machine (most Stop, interlock or safety category events) the downtime will be greater than the duration.

Trend This just means the behaviour of a variable plotted over time. We plot two trends the count, and duration trend. By looking at these graphs you can quickly see when a particular issue starts and ends, how big the problem is and whether you are making an impact on.

15 Appendix B – Design documentation

Contents

1.0 FMOCR System Overview	60
1.0 FMOCR System Overview	60
2.0 FMOCR Log file format	60
3.0 Implementation language	61
4.0 Data Structures.....	61
5.0 FMOCR log tools Summary.....	61
6.0 Fmtrend.....	62
6.2 Fmtrend helper functions	62
7.0 Fmproc	63
7.1 Usage summary:.....	63
8.0 Fmreports.....	63
8.1 Usage summary:.....	63
8.2 Fmreports helper functions.....	63
9.0 Class Summary.....	64
10.0 Class Descriptions	64
10.1 Fmevent class	64
10.1.1 Fmevent class variables.....	64
10.1.2 Fmevent public class methods.....	65
10.1.3 Fmevent private class methods	66
10.2 Query class	68
10.2.1 Query class variables.....	68
10.2.2 Query class public methods.....	69
10.2.3 Query class private methods	71
10.3 Utility class.....	74
10.3.1 Utility class variables	74
10.3.2 Utility class public methods	74
10.4 Plot class.....	78
10.3.1 Plot class variables	78
10.3.2 Plot class public methods	78
10.3.3 Plot class private methods	79

1.0 FMOCR System Overview

FMOCR machines generate event data which is store in a log file on a machine at the mail processing centre in question. Each day these log files are then sent over the Australia Post network to a network centre logging centre server for processing. It is on this server that our logging tools will be operating on.

2.0 FMOCR Log file format

To understand how FMOCR tools operate it is important to understand the FMOCR log file format, universal event file format (UEFF).

Field #	Field Name	Description
1	Date	Event starting date
2	Time	Event starting time, floating point seconds from Midnight on this date
3	UID	A unique identifier assigned to the event
4	Machine Type	Type of machine. FMOCR machines start with 7
5	Machine ID	Machine Number
6	Event Type	A numeral indicating a common event type.
7	Event Text	Machine specific text relating to the event
8	Text Delineator	The character to be used as a field separator
9	Event Data	representation of additional event information not otherwise incorporated into the file
10	Module	Module name EG stacker Feeder
11	Module ID	Module Instance id EG stacker 56, Feeder 3
12	Event Duration	Duration of the event in floating point seconds
13	Event Criticality	A textual indicator of the criticality of the event such as Safety, Interlock, Stop, Impair, Warning etc
14	Impairment	An estimate of how this event impaired machine performance in floating point percentage

Table 5 - FMOCR Log file format

2.1 Example FMOCR log entry

The example below, taken from an actual log file, helps illustrate the format of FMOCR log files as describe in Table 1 above

```
8/09/2004,1400.041992,701400044,7,701,Uncategorised,Stop,@,machine_feeder_3@
@Stopped,Module,0,!0000000000,Unknown,-1
```

3.0 Implementation language

The task of analysing FMOCR log files could have been implemented using the many programming languages available which include Java, Perl and PHP among others. I choose to implement the required functionality in C++. To understand how I came to this decision lets at the following points.

- ✓ The original prototype was written in C
- ✓ The tools had to be up and running in very short period of time
- ✓ I am more comfortable with C/C++ than the alternatives
- ✓ The tools needed to run on Linux and be able to access system calls and programs like Gnuplot (for graph plotting)
- ✓ The tools needed be relatively fast. Users needed access to the information they require pretty fast.

Since the original prototype was written in C using C++ enabled me to use some of the existing functions like graph plotting algorithms whilst at the same time utilising the powerful built in C++ data structures like Vectors. All C programs can be compiled using the C++ compiler.

Because the FMOCR required tools had to be up and running in a relatively short period of time C++ cuts development time. For example C++ has automatic memory management and has an extensive library including lists, maps and vectors. The ability to overload functions is another productive feature of C++ I intended to exploit. The utilities needed to make use of external Linux utilities like Gnuplot (for plotting graphs). C++ enables you to easily call these programs from your tools/programs. Java can call these external programs as well but the procedure is not as straight forward as I would like it to be.

4.0 Data Structures

All the three tools in this report make extensive use of C++ **Vectors**. Arrays, Lists, Queues or Stacks are not used in any part of these tools. Because FMOCR log files come in different sizes (hence they contain different number of events) a static data structure like arrays is out of the question. In fact my own research tells me that arrays are only slightly faster than vectors.

Another alternative would have been to use a linked list to store the events but I choose vectors as they are already part of C++, allow random access and I do not have to worry about memory management.

5.0 FMOCR log tools Summary

Tool Name	Description	Classes Used
Fmtrend	Trend Tool. Reads log files and generate trends based on user queries.	Query, Fmevent & Plot
fmreports	Daily reports tool. Generates html reports to be emailed based on user watch lists.	Query, Fmevent & Utility
Fmproc	Form processing tool. Fmproc will split an html form query string into name value pairs.	

Table 6 - FMOCR log tools summary

6.0 Fmtrend

6.1 Fmtrend Usage summary:

fmtrend -f<search field> **-m**<multiple> **-w**<watch list> **-k**<search string> <log files>

fmtrend reads log files (which are in comma separated values form), searches for events which match the user search string. The whole operation is as follows:-

- ✓ Verify that arguments are entered correctly using the function *ValidateFormArguments*. Check that files are CSV files using the function *IsCsvFile*.
- ✓ Check if the watch list is valid and update the watch list flag called *watchlist* accordingly
- ✓ Check to see if *watchlist* flag is set. If it is set create a **query** object and pass it the watchlist filename. The query object and the whole query process will be discussed further later on in this document.
- ✓ If there is no watchlist to process just process the user search string
- ✓ If the query returned any results use the Plot class to create graphs to be displayed to the user.

6.2 Fmtrend helper functions

bool IsCsvFile(string str)

This function tries to establish that a command line parameter, *str*, given to *fmtrend* is a CSV file. It does this by checking if the last four characters are “.csv”. Further checks will be done by the *Query* class to see if the files can be opened. This function's main job is to separate FMOCR log files from other command line arguments (other command line arguments do not end with “.csv”).

ValidateFormArguments(vector<int>& field_keys, vector<string>& search_strings)

We expect the user to enter a search string in the form search fields. If they do not enter anything in those fields then they should be left with the word “NONE” or blank. This functions makes sure that empty fields or the ones with the string “NONE” are not included in the query.

- *field_keys* is a reference to a vector which will contain form field keys after validation. Contains integers between 1 and 14 inclusive. For example the field key for “Event Text” will be 7.
- *search_strings* is a reference to a vector which will contain form search strings after validation.

bool CheckFile(string str)

Given a filename we need to be able to verify that it is valid. This function simply tries to open that file. IF the subsequent file pointer is not NULL then we know its valid and we return “true”. If it is NULL then we warn the user that this filename does not point to valid file (return “false”) on the system and will be omitted from the queries.

- *str* is the filename

7.0 Fmproc

7.1 Usage summary:

fmproc <form query string>

When you enter search strings on a form they are posted (sent to the corresponding CGI) as name value pairs with special characters (including spaces) substituted with their hexadecimal equivalents. The challenge is to get the actual field values and convert those hexadecimal values back to the special characters. We especially need space characters to be preserved when doing phrases queries. This is where *fmproc* is useful.

8.0 Fmreports

8.1 Usage summary:

fmreports **-W**<watchlist> **-U**<username> **-C**<count threshold> **-D**<duration threshold>
<log files> **-e**<email report> **-i**<image filename>

fmreports generates reports that can either be displayed on the web server or automatically emailed to users. It accepts options arguments "username", "count threshold", "duration threshold", "email reports", "image filename" and the compulsory arguments "log files" and "watch list"

8.2 Fmreports helper functions

bool IsCsvFile(string str)

This function tries to establish that a command line parameter, *str*, given to *fmtrend* is a CSV file. It does this by checking if the last four characters are ".csv". Further checks will be done by the *Query* class to see if the files can be opened. This function's main job is to separate FMOOCR log files from other command line arguments (other command line arguments do not end with ".csv").

Int getMachineNumberFromFileName(string str)

Gets the machine number from the log file

void printUsage()

Displays Fmreports usage to the user, shows the user how to use the programs

bool adequateFiles(vector<vector<string> > filenames)

Determines if the correct number of log files are included on the command line

void SortCsvFiles(vector<string> files_from_command_line, vector<vector<string> >& filenames)

Puts each machine's log files into the appropriate vector for further processing by the Query and Utility classes

9.0 Class Summary

Class Name	Description
Fmevent	Defines and describes an event in the FMOCR log file. Defines such attributes as machine number, event duration, event impairment and the time the event happened.
Query	Handles all user searches. Can handle queries based on any property of an Fmevent (event text, event type, duration etc) or based on user watch lists.
Utility	This is a general class which contains many useful helper functions including html generation, file name generation, calculating events count and durations.
Plot	Makes use of the open source Gnuplot program to plot events trends. It prepares the data and control files to be given to Gnuplot program so that it can run in non-interactive mode.

10.0 Class Descriptions

10.1 Fmevent class

The Fmevent class is based on the description of the universal event file format as previously described in this document.

10.1.1 Fmevent class variables

Class variable	Description
vector<string> event_tokens	Temporary vector to store event fields
int day	C++ integer describing the day the event happened
int month	Integer describing the month the day happened
int year	The year the event happened
int uid	A unique identifier assigned to the event
int mach_type	Integer defining the machine type
int machID	Integer defining the machine id, 701, 702, ..., 708
int moduleID	Module instance identification
double seconds	Returns the number of seconds that has elapsed since midnight.
double event_duration	Duration of an event in floating point seconds
double impairment	An estimate of how this event impaired machine performance in floating point percentage
string event_type	C++ string describing the event type
string event_text	String describing the event text
string event_data	C++ string describing the event text
string module	Module description

string criticality	C++ string describing event criticality
string year_temp	Temporary full date storage before we split it into day, month & year
string text delimiter	Event text delineator

Table 7 - Fmevent class variables

10.1.2 Fmevent public class methods

Fmevent (string line)

This is the class constructor. After being given a comma separated string which contains an event information (date, duration, etc) it checks to see if the line contains valid event information and then initializes all the event fields (duration, module, criticality, etc) appropriately.

bool Valid()

Does a simple check to see if an event is valid by checking to see if all 14 fields are defined. Returns false if they are not all completed and true otherwise.

int GetDay()

Returns the day of the month this event happened.

int GetMonth()

Returns the month this event happened.

int GetYear()

Returns the year this event happened, 2004 for example

int GetMachineID()

Returns the machine identification number associated with this event

int GetModuleID

Returns the module identification number associated with this event

int GetUID

Returns this event's unique identification number

double GetDuration()

Returns this event's total duration in seconds

double GetSeconds()

Returns the number of seconds from midnight till this event started happening

double GetImpairment()

Returns the value of this event's impairment

string GetType()

Returns the type of this event

string GetText()

Returns the description of this event

string GetTextDelimiter()

Returns the text delimiter used to split this event

string GetEventData()

Returns this event's data

string GetModule()

Get the module name associated with this event

string GetCriticality()

Get this event's criticality

int GetTimeHours()

Returns Hours elapsed since 12 midnight when event happened

int GetTimeMinutes()

Returns minutes elapsed since the last hour when event happened

int GetTimeSeconds()

Returns seconds elapsed since the last minute when event happened

int GetMachineType()

Returns this event's machine type

void PrintEvent()

Print the contents of this event on the screen. This could be useful as a debugging function.

void PrintTime()

Print the time this event happened on the screen. This could also be useful as a debugging function.

10.1.3 Fmevent private class methods

void Tokenize(const string& str, vector<string>& event_tokens, string separator)

Given a character (e.g. comma) separated strings like an FMOOCR log file entry it separates the values and put them in a vector. All you have to do is give it the string to separate (str), a reference to a vector that will contain the end tokens (event_tokens) and a character(s) that separate the values.

For example if we give this method the following string (from an FMOOCR log file):-

```
8/09/2004,1400.041992,701400044,7,701,Uncategorised,Stop,@,machine_feeder_3@  
@Stopped,Module,0,!00000000000,Unknown,-1
```

It will be split into 14 different strings and stored in the specified vector. The use of allow us to have an arbitrary number of tokens in a string giving us the flexibility (in the future) to have log events with say 200 entities without having to rewrite this method.

string RemoveQuotes(string str)

Sometimes after splitting, using the tokenize method, a string may have trailing quotation marks. This method simply removes those quotation marks and returns a string without them.

10.2 Query class

The query handles all user queries ranging from the ones needed to plot trend graphs to the ones needed to generate user daily email reports. It can handle an arbitrary number of queries enabling it to handle multi-filed queries. For example the `fmtrend` trend tool currently allows you to search using four different criteria. This is not a limitation of the query class but of the search form itself. Finally the query class also understands watch lists. Watch lists are described in the user guide section accompanying this documentation.

10.2.1 Query class variables

Class variable	Description
<code>int records_count</code>	The number of events in this query
<code>vector<string> filenames</code>	The vector containing FMOOCR log filenames to be used to generate the report
<code>time_t start_time</code>	Start time of the earliest event. Useful for the plot class
<code>time_t end_time</code>	End time of the last event
<code>char s[LINE_LENGTH]</code>	The string to hold the line containing the event entry in the log file. <code>LINE_LENGTH</code> is defined in the query class header file as a constant equal to 2049
<code>FILE* filePtr</code>	An arbitrary file pointer
<code>Fmevent* event</code>	A pointer to an <code>Fmevent</code> object
<code>double events_duration_total</code>	The durations total of events in this report/query
<code>vector<string> search_strings</code>	The vector containing the search strings we need to include in the report/query
<code>vector<string> temp_search_strings</code>	Temporary storage for search strings. Used in the constructor to populate <code>search_strings</code>
<code>vector<int> field_keys</code>	Form field keys storage. Keys enable us to determine what criteria we are searching on. For example a key of 7 means we are searching on the event text field
<code>vector<int> temp_field_keys</code>	Temporary form field storage
<code>vector<vector<Fmevent*> > master_results</code>	When we have multiple queries to perform we store each result in a vector of <code>Fmevents</code> which in turn we also store in vector thereby ending up with a vector of vectors
<code>vector<Fmevent*> unique_events</code>	FMOOCR unique events. Unique events are used with the trend tool to indicate to the user the unique events included in the trend.
<code>vector<Fmevent*> fmreports_vector</code>	FMOOCR events to included for the <code>fmreports</code> tool
<code>int num_queries;</code>	Number of queries to perform. This is used for compound/multiple queries
<code>int current_query</code>	An indication of which query we are at when doing multiple queries
<code>string search_str</code>	The search string

Table 8 - Query class variables

10.2.2 Query class public methods

Query()

This is the default constructor used by the fmreports tool. It takes no arguments and does not initialise any variables.

Query(vector<string> files, vector<string>search_strings, vector<int>field_keys)

This is the second of three constructors used by the query class. This is used when we have multiple search strings to include in our queries.

- `vector<string> files` is the container with the FMOCR log files to use for the query
- `vector<string> search_strings` is the list of search strings to use for our query
- `vector<int>field_keys` together with the search strings allows us to know which field of the log file to base our search on. For example if search string is "Shuttle Jam" and field is "7" then we know that we are searching for an event text that matches "Shuttle Jam".

Query(vector<string> files, vector<string>search_str, vector<int>fields, string watch_filename)

This is the third and final constructor for the query class. It is similar to the second one above except for the fact that it contains a file name containing the watch lists (watch lists are discussed in the help file accompanying this documentation).

void GetResults(vector<Fmevent*>& v, vector<Fmevent*>& v2)

After performing multiple queries this methods update the final vector which contains the final result and it also updates the unique events found whilst doing the queries.

- `v` is a reference to a vector which will contain the final results
- `v2` will hold unique events included in `v`.

void Search(int key, string search_str)

Given a key and a search string this method will direct you to the appropriate method (as described later on in this documents) to handle your query and it will pass on the search string. For example if you have a key of "7" and a search string of "Shuttle Jam" this method will match this with a method in this class called `QueryByEventText(search_str)` by using C++ case statements.

- `key` is the form search field
- `search_str` is the search you are after

void QueryByImpairment(string var)

Perform your query based on event impairment.

- `var` is your search string

void QueryByEventCriticality(string var)

The query will be performed based on event criticality.

- `var` is your search string

void QueryByEventDuration(string var)

The query will be performed based on event duration.

- *var* is your search string. The method will convert it to a double using the C/C++ `atof` function

void QueryByModuleID(int var)

The query will be performed based on module identification number.

- *var* is the identification number we are trying to match.

void QueryByModule(string var)

The query will be performed based on the FMOCR module name

- *var* is the module name we are trying to match

void QueryByEventText(string var)

The query will be performed based on the event text

- *var* is the event text we are trying to match

void QueryByEventType(string var)

The query will be performed based on the event type

- *var* is the event type we are trying to match

void QueryByMachineID(int var)

The query will be performed based on machine identification number

- *var* is the machine identification number we are trying to match

void QueryByUID(int var)

The query will be performed based on the event's unique identification number

- *var* is the event's identification number we are trying to match

void QueryByTime(string search_str)

Query the time the event happened. For example to get all the events after 10pm the method will accept a *search_str* of 22:00+. Similarly for all events before 10pm the method will accept 22:00- (notice the use of + and – signs). More documentation on how to perform time based queries is in the user guide. This method delegates some of the processing duties to low level functions which are described later in this document, particularly the function the `ProcessTimeQuery`.

- *Search_str* is the time the event we are after happened

void QueryByDate(string search_str);

Performs the query based on the date in question. Use of + and – signs as in `QueryByTime` also apply with this method

- *Search_str* is the date the event we are after happened

void QueryByEventData(string search_str);

Performs the query based on event data.

- *Search_str* is the event data we are searching for.

void QueryByMachineType(int search_str)

Performs the query based on machine type.

- *Search_str* is the machine type we are searching for.

void ReportsQuery(vector<double>& totals_duration, vector<int>& totals_count, vector<string> filenames, string search_strings_source_file, vector<string>& query_strings)

This is the function responsible for processing daily html reports which are either emailed to clients or displayed on the FMOCR web site.

- *Totals_duration* is the vector that will be containing the events durations total after the query is finished
- *Totals_count* is the vector cor that will be containing the total number of events meeting the search criteria
- *filenames* is vector containing the FMOCR log files to use for generating the report
- *search_strings_source_file* is the file containing the watch list
- *query_strings* is a reference to vector where the search strings (as described in the watch list) used to generate this report. This is handy when generating the report html file.

bool Match(char* source, char* str, int fmreports)

This function simply determines if the given two strings are equal or if one is a substring string of the other. It does this by utilising the C library function *strstr*.

- *source* is the first string
- *str* is the other string we are trying to match with

string GetEventsDurations()

Calculates the durations total of all the events in the query results. It formats the result into a string which will be displayed to the user on the web page.

10.2.3 Query class private methods

time_t GetIntTime (int day, int month, int year, double secs)

returns the number of seconds that has elapsed since 1900.

- *day* is day of the month
- *month* is month of the given year
- *year* is the given year
- *seconds* is the given seconds

void UpdateTimeRange(vector<Fmevent*>& results)

This function updates the events time range to be used during the graph plotting routines.

string ToLowerCase(string source)

This function converts a given string, *source*, to lower case. This is useful especially when doing string comparisons.

bool MatchRecord(Fmevent* event, int key, string var)

Given an event, form field key and search string var, the method will use the form field key to determine with method to delegate the query process using a series of case

statements. For example `MatchRecord(event, 7, "shuttle Jam")` will result in this method calling the function `QueryByEventText(event, "shuttle Jam")`.

- `event` is a pointer to an `Fmevent` object
- `key` is the form field key
- `var` is the search string

The function returns true if there was a match.

bool ProcessEventDuration(Fmevent* event, string var)

This is a helper function for the `EventDurationQuery` method.

- `var` is the string containing duration ranges
- `event` a pointer to the `Fmevent` whose duration we are comparing with

Returns true event duration is within the range of `var`.

bool ProcessTimeQuery(Fmevent* event, string var)

This is a helper function for the `QueryByTime` query method.

- `var` is the string containing time ranges
- `event` a pointer to the `Fmevent` whose time we are comparing with

Returns true event time is within the range of `var`.

string QueryRemoveSpace(string source)

This method removes spaces and non-alphanumeric characters on a given string. This is also useful when doing string comparisons.

void Tokenize(const string& str, vector<string>& event_tokens, string delimiters)

Given an FMOCR log entry or a delimited string, `Tokenize` will split it into its constituent tokens and store the result in a vector.

- `str` is the delimited string
- `event_tokens` is the vector for storing resultant tokens
- `delimiters` is the character to used to determine token separation

void UpdateUniqueRecords(vector<Fmevent*>& events)

We need to be able to present the user with unique events that make up the result of their query. This function checks to see if an event has not been included already and if not so it will include it. It avoids duplicates being presented to the user.

`events` is the containing `Fmevents`

int SecondsSinceMidnight(int hour, int minutes)

Given the number of hours and minutes this function returns the number of seconds that has elapsed since midnight, 0000HRS.

- `hour` is the number of hours since midnight
- `minutes` is the number minutes since midnight

bool ProcessImpairment(Fmevent* event, string var)

This is a helper function for the `ImpairmentQuery` method.

- `var` is the string containing impairment ranges
- `event` a pointer to the `Fmevent` whose impairment we are comparing with

Returns true event impairment is within the range of var.

char* GetTextAndDuration(char* str, int event_field, int duration_field, double& event_duration, char *Buffer, int flag)

Given an FMOCR log file entry/line this function will split into tokens and extract the “event text” and “event duration” fields. It makes use of the C library function *strsep*. We could have used *strtok* for tokenising the entries but it suffers from its inability to handle null fields as compared to *strsep*.

- Str is the log file entry/line
- *event_field* is the field number for event text
- *duration_field* is the field number for the event duration field
- *event_duration* is a reference to the actual event duration extracted from the log file entry

The function returns the event text from the log file entry.

void WatchListQuery(string watchlist_filename, vector<Fmevent*>& events)

When the user wants to include a watch list in their query this function will handle that. It reads the contents of the watch list filename and stores the results in *events* vector.

- *watchlist_filename* is the filename containing the watch lists
- *events* is the storage container for *Fmevents*

10.3 Utility class

The utility class helps in implementing the **fmreports** tool by providing simple but powerful methods like generating html table captions, generating filenames. Instead of putting everything in the fmreports tool the utility class handles most of the general functionalities. In fact some of the methods in the utility class should be able to be used by future unified logging system tools.

10.3.1 Utility class variables

Class variable	Description
vector<int> machnos	FMOOCR machine numbers container
time_t current	The current time as C/C++ time_t
tm local	Time as C/C++ tm structure
char* imgtagtemplate	The image template to be used by the GNU/Gnuplot graph plotting program
string csvfilename	The FMOOCR log file
string picfilename	The filename of the image generated by GNU/Gnuplot
string working_dir	fmreports working directory
string watchlist	Watch list filename as string
char watchname[120]	Watch list filename as char
char ctlfilename[2048]	The control filename to be used by the GNU/Gnuplot graph plotting program
string username	The username of the person requesting the reports
bool multiple_graphs	Multiple graphs control flag
bool durations_graph_only	Events durations only graphs control flag
string graph_xrange	Graph plotting range
bool count_graph_only	Events count only graphs control flag

10.3.2 Utility class public methods

Utility(vector<int> machno, string user, string working_directory, string watch="none");

This is the class constructor. It takes the following arguments:-

- o *machno* is the vector containing the FMOOCR machine numbers
- o *user* is the username of the person requesting the report
- o *working_directory* is the fmreports working directory
- o *watch* is the watch list name

void PrintHtmlHeader(string html_filename, string html_title)

This method creates the report html file and writes the html head including setting the html page title. The rest of the file will be finished off by other functions including *WriteHtmlTable* and *PrintHtmlTail*.

- o *html_filename* is the required html filename
- o *html_title* is the html page title

void PrintHtmlTail(string html_filename, string image)

This function finishes off writing the html report by including the generated graph and closing the html tags like “</body>” and “</html>”.

- o *html_filename* is the html filename
- o *image* is the generated graph name

void WriteCsvFile(string plotfile_csv, vector<int> horizontal_totals, string pngfile, string ctlfile)

WriteCsvFile will generate a comma separate value file (based on the events **count/frequency**) to be used by the GNU/Gnuplot utility to generate comparison histograms.

plotfile_csv is the required CSV filename

horizontal_totals is the vector containing the events count totals

pngfile is the filename of the resultant comparison graph (histogram) in *png* format

ctlfile is the Gnuplot control filename

void WriteCsvFile(string plotfile_csv, vector<double> horizontal_totals, string pngfile, string ctlfile)

WriteCsvFile will generate a comma separate value file (based on the events' **durations**) to be used by the GNU/Gnuplot utility to generate comparison histograms.

plotfile_csv is the required CSV filename

horizontal_totals is the vector containing the events count totals

pngfile is the filename of the resultant comparison graph (histogram) in *png* format

ctlfile is the Gnuplot control filename

void WriteHtmlTable(vector<string> events, string html_file, vector<vector<int> > master_totals, int count)

This function writes the query results (**counts totals**) onto an html table for presentation to the user requesting the report.

- o *events* is the vector containing the Fmevents in the result
- o *html_file* is the html filename
- o *master_totals* is the vector containing vectors of **events totals**
- o *count* is user count threshold

void WriteHtmlTable(vector<string> events, string html_file, vector<vector<double> > master_totals, double duration)

This function writes the query results (**duration's totals**) onto an html table for presentation to the user requesting the report.

- o *events* is the vector containing the Fmevents in the result
- o *html_file* is the html filename
- o *master_totals* is the vector containing vectors of **events durations totals**
- o *count* is user count threshold

void GetHorizontalCountTotals(vector<vector<int> > master_totals, vector<int>& results)

GetHorizontalCountTotals calculates the table horizontal totals (*events count/frequency*) to be included in the report html table.

- *master_totals* is the vector containing vectors of **events totals**
- *results* is the vector that will be containing the results

void GetHorizontalCountTotals(vector<vector<double> > master_totals, vector<double>& results)

GetHorizontalCountTotals calculates the table horizontal totals (*events durations totals*) to be included in the report html table.

- *master_totals* is the vector containing vectors of **events durations totals**
- *results* is the vector that will be containing the results

void GetVerticalCountTotals(vector<vector<int> > master_totals, vector<int>& results);
int GrandTotal(vector<int> v)

GetVerticalCountTotals calculates the table vertical totals (*events count/frequency*) to be included in the report html table.

- *master_totals* is the vector containing vectors of **events totals**
- *results* is the vector that will be containing the results

void GetVerticalCountTotals(vector<vector<double> > master_totals, vector<double>& results)

GetVerticalCountTotals calculates the table vertical totals (*events durations totals*) to be included in the report html table.

- *master_totals* is the vector containing vectors of **events durations totals**
- *results* is the vector that will be containing the results

int GrandTotal(vector<int> v)

This function sums the contents of the vector v, containing integers (*events count/frequency*), and returns the total.

string GetCaption()

Generates the html table caption

double GrandTotal(vector<double> v)

This function sums the contents of the vector v, containing doubles (*event durations*), and returns the total.

void Plot(string csvfile, string ctifile, string pngfile)

Given a CSV file, a control file for the Gnuplot utility and the required portable network graphics filename the function will plot a histogram comparing different machines performance.

- `csvfile` is the comma separated values filename containing machines performance data
- `ctlfile` is the control filename for the Gnuplot program
- `pngfile` is the filename of the resultant PNG file

void Plot(string csvfile, string csvfile_2, string ctlfile, string pngfile)

Given a CSV file, a control file for the Gnuplot utility and the required portable network graphics filename the function will plot a histogram comparing different machines performance.

void SetControlFileTemplate()

This method sets the control file template. The control file is used by the Gnuplot utility. It contains control information like image size, image format and image title.

string GenerateFileName(string username)

Generates an output filename associated with the report based on username, current date and time. This is useful during generation of multiple reports.

- `username` is the name of the user requesting the report.

void setMultipleGraphs(bool my_bool)

Sets the Gnuplot control file template based on whether one or two graphs are required.

`my_bool` is a flag indication whether one or graphs are required

10.4 Plot class

The plot class is responsible for creating trend graphs for the mail sorting machines. In order to create such graphs appropriate data has to be gathered and prepared for the Gnuplot utility to produce the graphs. Appropriate time ranges have to be calculated based on what the user selected from the form. For example a user might a trend over a week's period of time or for the whole month.

10.3.1 Plot class variables

Class variable	Description
const char** ctlfiletemplate	Gnuplot control file template
int total_records	Total number of records/events involved in plotting the trend graph
int filecount	Total number of FMOCR log files involved in creating the graph
int inspect	Checks to see if two minute intervals are needed
int multiple	Do we need multiple graphs
long hours_per_bucket	Hours per plot bucket
long secs_per_bucket	Seconds per bucket
FILE *errout	Program errors file pointer, will eventually point to standard error, <i>stderr</i>
int HOUR, MIN, SEC	Hours, minutes and seconds
double plot_data[POINTS + 10][2]	Plot data to be included in the CSV file for the Gnuplot program. POINTS is declared in the definitions section as equal to 744
time_t plot_start, start	Plot start time
time_t plot_end, end	Plot end time
time_t plot_increment	Plot increment time
char* imgtagtemplate	Gnuplot image template
char csvfilename[2048]	CSV file containing plot data
char picfilename[2048]	Resultant image/graph filename
struct timerange_s	Plot time range
char ctlfilename[2048]	Control file name

Table 9 - Plot class variables

10.3.2 Plot class public methods

Plot(time_t st, time_t et, int int_multiple, int interval=0)

The plot constructor relies on the calling class to have calculated the plot start and end times. If no plotting interval is provided

time_t AlignTimeToDay(time_t t)

This functions aligns time so that the reference point is 12 midnight. A common reference point is important when generating trends.

- o **t** is the time that needs to be aligned

void PlotPoint(time_t x, double Duration)

PlotPoint generates the plot data to be used by the Gnuplot program

- o *x* is the aligned time
- o *Duration* is the events duration

void DumpPlot (time_t end, const char *image_filename)

After setting up the control file template DumpPlot calls Gnuplot to plot the trend graph.

- o *end* is plot end time
- o *Image_filename* is the name of output graph/image

void SetStartTime(time_t start)

SetStartTime sets the plot start time.

void SetEndTime(time_t end)

Sets the plot end time

void CalculatePlot (vector<Fmevent*> results)

Calculates the plot points by calling the DumpPlot function

- o *results* is a vector containing the query results from which we will get such attributes as event duration

10.3.3 Plot class private methods

void SetControlFileTemplate()

The Gnuplot (see glossary) program we make heavy use of in this project relies on control files for non-interactive operation. Gnuplot control files set the required output file format (png, jpeg, gif, etc) and the size of the image among other things.

This function sets the Gnuplot control file template.

time_t GetIntTime (int day, int month, int year, double secs);

This function combines the given day, month, year and seconds into a long integer such that it is the number of seconds that has elapsed since 1900.

Makefiles

Fmtrend tool makefile

```
fmtrend: fmtrend.o fmevent.o query.o plot.o
    g++ -pg -O3 -static -o fmtrend fmtrend.o fmevent.o query.o plot.o
fmtrend.o: fmtrend.cpp fmevent.h query.h
    g++ -c -pg -O3 fmtrend.cpp
fmevent.o: fmevent.cpp fmevent.h
    g++ -c -pg -O3 fmevent.cpp
plot.o: plot.cpp plot.h fmevent.h
    g++ -c -pg -O3 plot.cpp
query.o: query.cpp query.h fmevent.h
    g++ -c -pg -O3 query.cpp
```

fmreports tool makefile

```
fmreports: fmreports.o fmevent.o query.o utility.o
    g++ -static -g -o3 -o fmreports fmreports.o fmevent.o query.o utility.o
fmreports.o: fmreports.cpp fmevent.h query.h
    g++ -c -g fmreports.cpp
fmevent.o: fmevent.cpp fmevent.h
    g++ -c -g fmevent.cpp
utility.o: utility.cpp utility.h fmevent.h links.h
    g++ -c -g utility.cpp
query.o: query.cpp query.h fmevent.h
    g++ -c -g query.cpp
```


Testing

Because of the large data sets involved it have been difficult to do thorough testing within the given test plan. The following essential tests were still conducted.

User input tests

1. Non-existent log file tests
2. Non-existent watch lists
3. Empty search strings
4. Illegal search strings (i.e. entering alphabetic characters where integers are required or entering dates in the wrong format)

Program execution tests

1. Plotting of graphs
2. Automatic emailing of reports
3. NULL files handling
4. handing of incomplete records (less than 14 fields per record)
5. large log file processing (more than 15,000 records)
6. small log file processing (less than 500 records)
7. Caching of log files

Glossary

FMOCR - Flats Multiline Optical Character Recognizer

Gnuplot - Gnuplot is a command-line driven program for producing 2D and 3D plots.

UEFF – Universal Event File Format

PNG – Portable network graphics. It was developed as a replacement for the GIF standard due to legal entanglements resulting from GIF's use of the patented LZW compression scheme, and also because of GIF's many limitations. PNG is superior to GIF.

CSV – Comma Separated Values file format such as the ones used by FMOCR log files

Vector – In the context of this documentation, a vector is a C++ storage container capable of storing any valid C/C++ data structure.

16 Appendix C – Source code