<System Name> Technical User Manual

- Stephen McGregor wrote this document from scratch and in its entirety.
- This document is only for private evaluation.
- Do not retain this document after use.
- Do NOT distribute this document.

This document is only for use in association with hiring Stephen McGregor as a technical writer.

Authors : Stephen McGregor
Status: Distributed Review **Tuesday, February 11, 20XX** **Version: 0.9.3**

Table of Contents

TABLE OF CONTENTS : TABLES

1 Introduction

1.1 WHAT IS <SYSTEM NAME>

<System Name> is a messaging system, delivering messages *from* telephone, via a call centre or directly, from the internet and 'special-delivery' sources *to* pagers, fax machines, mobile phones and email accounts. Running on an HP 9000 server, with terminal servers, fax, internet, and history servers, call-centre hardware - plus a range of supporting devices - <System Name> is large suite of specialist telecommunication applications, written in the C programming language and supported by an HP Unix (version 11.00) operating system. An Informix C-ISAM database is used for high-speed data storage and retrieval.

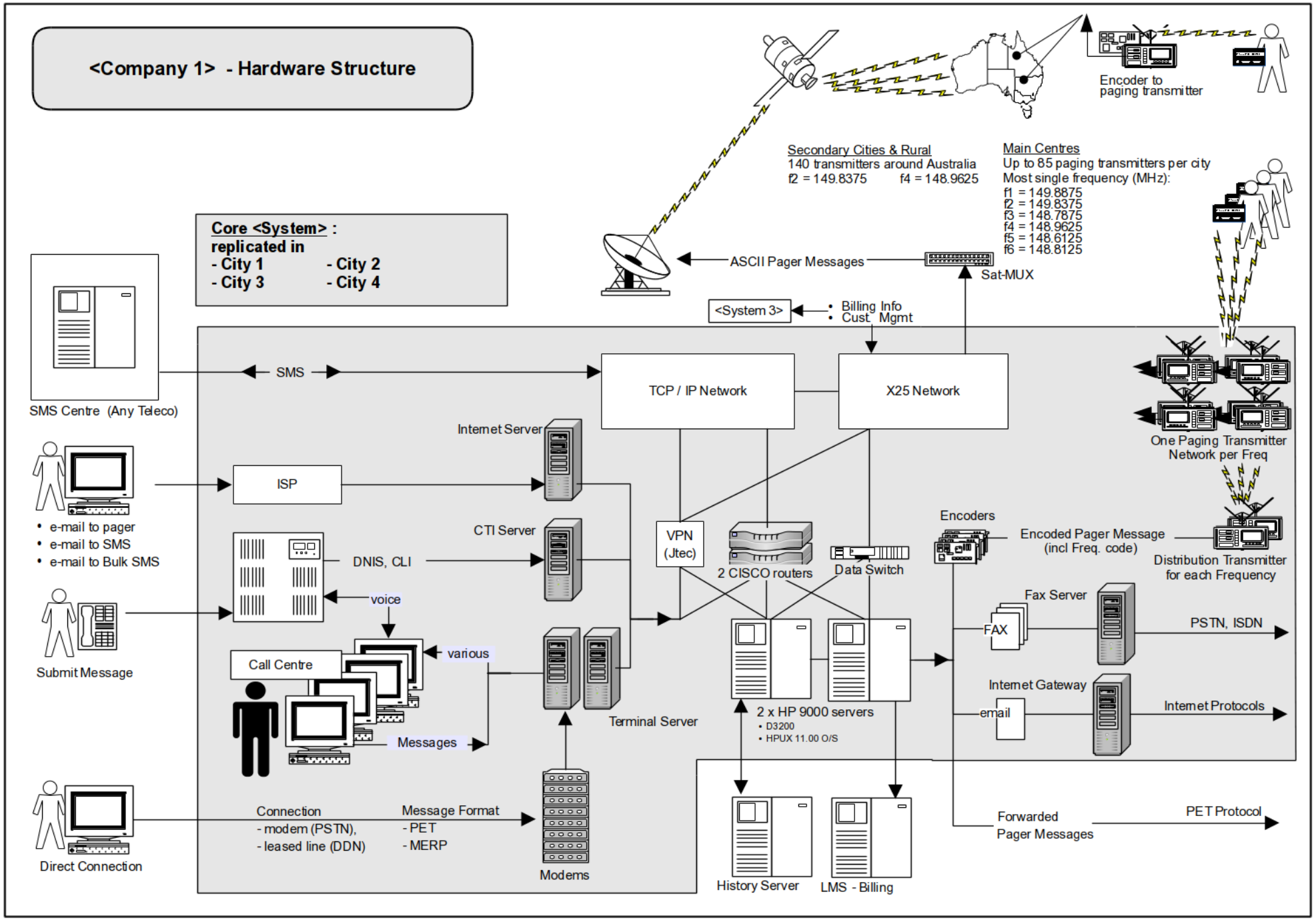
The five <System Name> sites in Australia (Adelaide, Brisbane, Perth, Melbourne and Sydney) are uniformly configured:

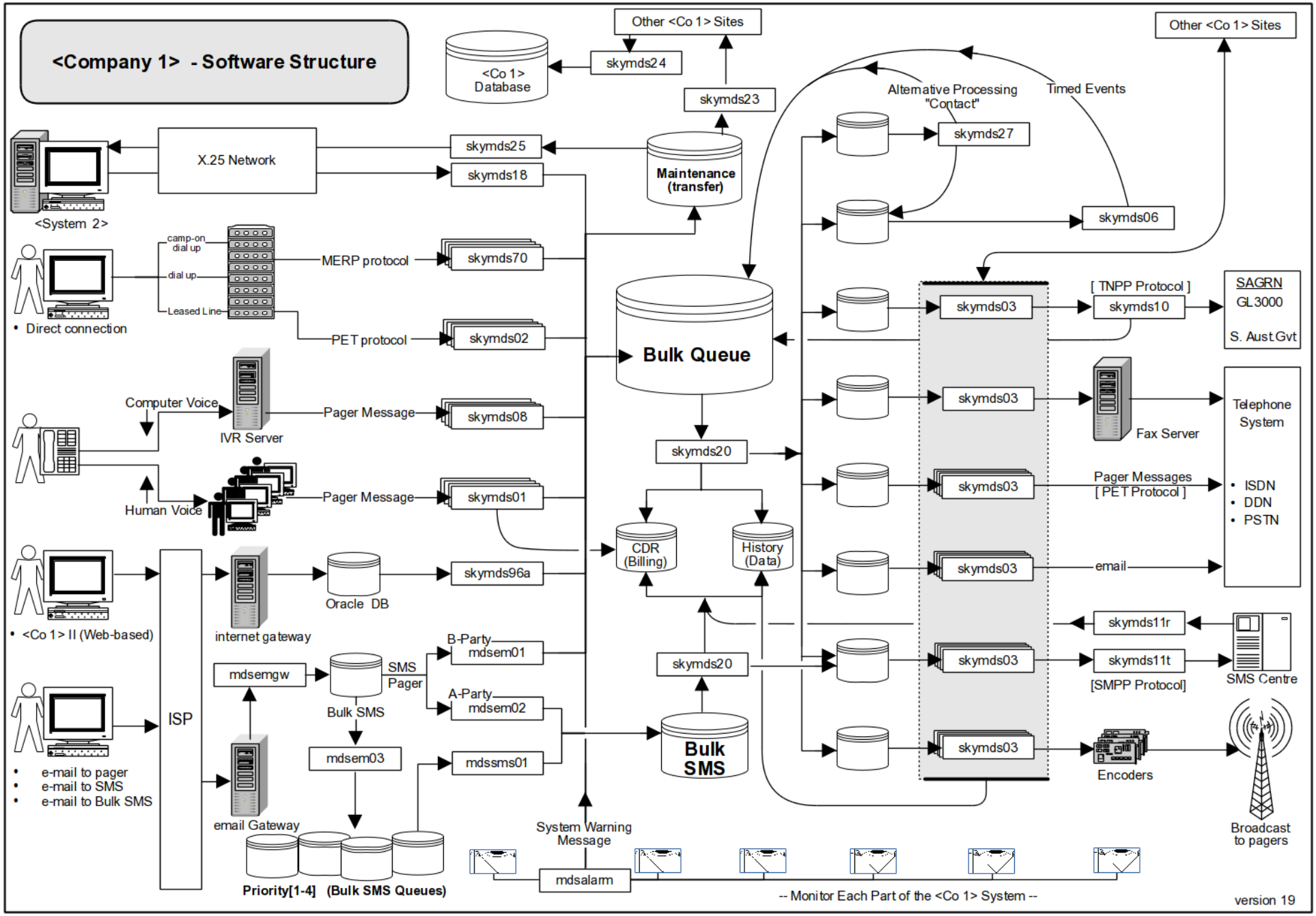
- to collect messages:
  - a call centre, PABX, and XACOM phone-call processor,
  - an internet gateway
  - a modem 'bank'
  - applications to run each of these devices and to manage collection of <System Name> messages

- to process messages:
  - One or two HP 9000 servers running HPUX Unix version 11.00
  - The Informix C-ISAM database. The message queues are implemented as flat files.
  - TCP/IP and an X.25 networks.
  - applications to run each of these devices and to manage the processing of <System Name> messages
  - a history collection process. All history older than seven days is moved to Melbourne.

- to send messages
  - encoders, amplifiers and transmitters for pager messages
  - fax and internet servers
  - connections to an SMS Gateway
  - PABX for forwarding pager messages for delivery by other telecommunications companies.
  - applications to run each of these devices and to manage transmission of <System Name> messages

<System Name> is a fast, integrated, and complete non-voice telecommunications solution. Born as a simple and robust paging system, twenty man-years of development has seen <System Name> grow into the fully-featured, highly-optimised system it is today.





2 The Background to <System Name>

2.1 A Brief History of <System Name>

Forty years ago saw the founding of Austas, an Australian telephone answering company that over the next twenty years grew to be the largest company of its kind in Australia. In 1985, Austas was brought by the young paging company, DataPage. Over the next couple of years this new conglomerate changes its name to, of course, <Company Name> ("<Company Name> Telecommunications") and, around the same time is bought by Bell South. Now, Bell South was in a buying mood and the next year they bought another set of Australian companies: <A>, <B> and <C>, which are merged into a single, all new, <Former Company Name>.

Bell South, one of the "baby bells" created in the demise of AT&T's monopoly, had come, at the end of the 1980's, to own two competing Australian pager companies - and the five RF networks that <Company Name> still operates today. <Former Company Name>, had its <Former Company Name> system: written in C and running on four PCs, each of which serviced one or more dumb terminals. The other, <Company Name>, had "<System Name>", a client/server application, externally built in both C and Cobol and running on - wait for it - "Tandem O/S" (of course) and XENIX/Unix [the client PCs]. Come 1991, BellSouth finally makes the decision to merge these two similar companies. <Former Company Name>, migrated to HP9000 servers, is chosen to form the basis of the new paging system, some of the extended <System Name> functionality is also included. Terminals connect via terminal servers, though over time small PCs (still running terminal emulation programs) replace dumb terminals. The system is called <System Name>.

Other additions are made. In 1991 Bell South took the almost unique step of purchasing its own satellite transmission company, "Equatorial Satellite Systems", one of the few telcos in the world to do so. The satellite is added to the <Company Name> corporation's assets while <Company Name> also gains another local paging company <D>. Finally, in 1994, Bell South sells <Company Name> to <Purchaser>.

The intervening years have witnessed constant improvements for <System Name>. After the 1994 sale, <Company Name> spent four years supplying (GSM) mobile phone services, while it moved permanently into the businesses of telephone bill payments and IVR (Interactive Voice Response). Towards the end of the decade the focus became concentrated on Call Centre services and the ability for <System Name> to process external clients' call centre work. Processing <major telco 1> and <major telco 2>'s SMS messages is just one example of this. Taking over all of <major telco 3>'s paging accounts in the months preceding the <date removed> is another.

Over the last 40 years <Company Name> has developed a number of almost unique advantages in its Call Centre business. <Company Name> has both extensive economies of scale and decades of experience. Further, no other company has developed its own software *and*, piece by piece, built up its own hardware network, a technical advantage which leads to being able to handle most client's communications business. The difference is : communications and call centre operations are <Company Name>'s core business. For everyone else they are at best a side-line – more usually some sort of distraction.

<Company Name> has become <removed>, taking over 30,000,000 calls a year, and often 2-300,000 transaction a *day*, while working with more than 17,000 separate Australian organisations. <Company Name> runs four Call centres, with 290 operators working 24 hours a day in Melbourne, Sydney and Adelaide, and 14 hours per day in Brisbane.

In mid <date removed>, after a long association, <Company Name> was purchased by the <Country> company '< new owner >'.

2.2 Current Directions

As just mentioned, the extension of <System Name> to third-party call-centre management has been one underlying theme to the progression of <System Name>'s functionality. Another more recent focus has been the extension of <System Name> to the domain of the ever-more-present internet. Currently one can email a pager message or an SMS via <System Name>, while <System Name> will send emails itself when appropriate. Beyond this a full web interface has been developed and further extensions of web and internet functionality will follow.

2.2.1 CURRENT PROPOSALS

Following are a couple of "current proposals" active at <System Name>. These are areas where some work has been done, plans made, and possibly some applications developed, yet not to any standard suitable for release as a <System Name> product. In fact, for one reason or another they might not become part of the system either the short or medium term. They do however give an indication of the direction(s) in which <System Name> may develop.

2.2.1.1 IVR - Interactive Voice Response

Some work has already been accomplished in this area: the computer generated voice that answers some calls and then prompts the user during automatic-pager calls. Further development is planned such that <System Name> can receive and interpret simple human spoken words and phrases to which it will generate spoken replies or act in other appropriate ways

2.2.1.2 2-way SMS

The proposal is that when a customer receives an SMS or a pager message they can, using a small keyboard, reply on the spot. The reply message will travel via a metropolitan wireless network to <Company Name> Communications, where <System Name> will forward it to its appropriate destination. Note that this destination may be any of a range of media, including email, fax, or another SMS or pager message. This, of course, would open up a whole new realm of mobile communications in addition to bypassing the necessity of making an SMS telephone call as the only text-based reply to a pager or SMS message.

In collaboration with <major telco 3>, such a system : "<Company Name> Airmail", is now operating.

2.2.1.3 Bulk SMS

While Bulk SMS has been available for a year or so, it is only now that we are seeing it used. This is probably a major change for <Company Name>, with, in the near future, a noticeable proportion of messages being Bulk SMS.

2.2.1.4 <System Name 2>

A bionic, web-based wrapper for <System Name> is in development.

3 Hardware of <System Name>

This section discusses, briefly, the machinery needed to run a <System Name> system. It is based on a typical <System Name> system found in the major Australian cities around the middle of <date removed>. As such, this list can be considered a sufficient collection of hardware upon which a <System Name> system may be established.

3.1 <System Name> Servers

The Servers in a <System Name> system are :

- The main <System Name> server(s). The <System Name> applications run on these machines
- The History Server : where historical data ends up
- LMS – the final destination of billing data. An Oracle database
- Internet Servers - for sending and receiving email messages
- Fax Servers

3.1.1 Central <System Name> Servers

The following details an arbitrary <System Name> machine. There may be slight, irrelevant, variances from site to site.

Machine	Task(s)	Spec	
lnkm, lnko	<System Name> Servers.	Machine :	2 HP9000/803 E-Class
lnks, lnkx		HD :	2 x 9 GB
lnka, lmkp		O/S :	HP-UX B.11.00
lnkb, lnkg		Software :	<System Name> system

Table 1 : <System Name> Server Hardware

- History Server

The History Server, the final resting place of <System Name>'s collective historical information, is a similar machine to the servers just detailed. As such it is (as at mid-<date removed>) a HP9000 machine running HP-UX B.11.00, though it neither is nor needs to be of the same specification as the actual <System Name> servers listed above.

- LMS

LMS is the Oracle billing system that receives the CDR billing data from the <System Name> system. The specification of this machine – practically external to the <System Name> system – will change independently of any <System Name> considerations.

- Call Centre Network - Terminal Servers

<System Name> installations generally use two or more Lantronix ETS32PR, 32 port terminal servers to connect (up to) 95 Call Centre operators' desktop computers to the <System Name> system.

- Fax Server

The <System Name> fax server is a simple desk-top Pentium running SCO-UNIX version 5.0.5.

- Internet Servers

Any <System Name> site will have at least two machines functioning as email servers. In Melbourne there is the additional <System Name 2> server and the machine supporting <Company Name>'s web site. The details of these machines (mid-<date removed>) are as follows

3.1.2 <SYSTEM NAME> INTERNET SERVERS

- Email Servers

Twin CPU Pentium IIIs with 256 MB of RAM, 18 GB hard drives, a Windows 2003 Server supporting an NT Sendmail program.

- <System Name 2> (Web-Based) Internet Server

Similar to the email servers, Pentium III, 600 MHz, but with a GigaByte of RAM and a tape-backup system A <System Name> server also has to have an Oracle database (8.1.5 or better) and IIS 4.0

- <System Name> Web-Site server

Another similar machine, a fast (733 MHz) Pentium III with 256 MB of RAM and 18 GB Hard Drive. Like the <System Name 2> server it runs NT4.0, IIS 4.0 and Oracle 8.1.5.

3.2 CENTRAL NETWORK

The <System Name> systems operate over two networks, a TCP/IP for the main traffic and an X.25 network for back-up, corporate use and a few other tasks. CISCO routers connect the one or two <System Name> servers to <Company Name>'s TCP/IP network, while a data switch and JTEC VPN modules connect the machines to the X.25 system.

3.2.1 TCP/IP

The <System Name> TCP/IP network connects each site via a 48K DDN (Dedicated Digital Network), connecting to each local 10 Mbit internal LAN. Most <System Name> messaging traffic travels over this network, the main back-bone of interconnection of <System Name> systems. The other major use is the connection by the SMS centres of external telecommunications companies for the transmission and receipt of SMS messages.

3.2.2 X.25

<Company Name>'s private X.25 network operates over a combination of 64K ISDN, and 48K, 19.2K, and 9.6K DDN lines. This network supports a 2Mbit Virtual Private Network (VPN) connecting the five <System Name> installations, used for :

- delivering messages to local and remote encoders
- a backup to the TCP / IP network
- As a connection point for <Company Name>'s <SYSTEM NAME 3> billing and customer management system.

3.3 CALL CENTRE

3.3.1 PABX AND ASSOCIATED

- Lucent PABX.

The Lucent PABX used by <System Name> is a twin CPU, Avaya Definity G3r10 model. It features twin hard-drives, 4000 ports and 2 CLAN cards per switch. The PABX is IP enabled, and runs "R010r.01.0.032.3" software. The PABX is accompanied by a CTI (Computer-Telephony Integration) server and "Automatic Call Distributor" software.

- IVR machines

<Company Name> often runs IVR (Interactive Voice Response) machines on behalf of its external clients. Currently a "Tracker" machine is installed which runs OS/2. However the details of the IVR machines will change. In the near future, as <Company Name> moves into this field, further IVR servers will be added to the collection of <System Name> hardware.

3.3.2 CALL CENTRE OPERATORS

As of mid-<date removed>, the four <System Name> installations have seats for a combined 290 Call centre operators, with 290 terminals. Each of these terminals is, in actuality, a desktop PC running a terminal emulation program, so little or no other software of note is stored on an operator's PC.

Machine	Task(s)	Spec	
Call Centre Operator's PC / Terminal		Machine :	various models
		CPU :	Pentium II or III
		RAM :	32-64 MB
		HD :	various
		O/S :	Windows 98
		Software :	Ericsson Terminal Emulation
			Internet Explorer 5

Table 2 : Call Centre Operator's Computer Hardware

3.3.2.1 Brief details of <System Name> Call centres

City	Call Centre Details	Computers
Sydney	93 Seats, 24 hours x 7 days / week	93 plus a few for supervisors
Adelaide	77 Seats, 24 hours x 7 days / week	77 plus a few for supervisors
Melbourne	75 Seats, 24 hours x 7 days / week	75 plus a few for supervisors
Brisbane	25 Seats, 14 hours x 5 days	25 plus a few for supervisors
		270 (plus 10 or 20)
Perth	No Call Centre. (<System Name>)	
Canberra	No Call Centre. (No <System Name>. Encoders & transmitters only.)	

Table 3 : Call Centre Details (brief)

3.4 EXTERNAL CONNECTIONS TO <SYSTEM NAME>

There are three hardware configurations via which an external customer may connect to a <System Name> system

- DMS – Digital Metropolitan Service

A Telstra managed 'DMS modem ↔ line ↔ DMS modem' service used by <Company Name>'s larger clients for 24 x 7 connection to a <System Name> system. At the time of writing there are about 25 DMS modems running in the Melbourne <System Name> installation.

- Dial-up Modem connections

A bank of about 45 modems manage the interim dial-up connections of casual <System Name> external connections. These are used by smaller clients who need only occasional connection to the <System Name> system.

- Dial-up "camp-on" Modem Connections

Dial-up "camp-on" modem connections are a combination of the above two connection channels. Connected to the <System Name> system permanently, or until the user disconnects, the connections use normal dial up modems and telephone lines as a cheaper 24 x 7 version of the Telstra DMS service described above. At the time of writing there are about 10 'camp-on' dial-up modems in the Melbourne <System Name> installation.

3.5 BROADCAST

<System Name> transmitter broadcast over 6 frequencies in Australia, one (f6) being used only for the South Australian Government. The 280 main-centre transmitters and 140 non-main-centre transmitters broadcast as follows :

Freq. Name	Frequency (MHz)	Broadcast locations
Frequency 1 (f1)	149.8875 MHz	Brisbane, Canberra, Melbourne, Sydney.
Frequency 2 (f2)	149.8375 MHz	Adelaide, Brisbane, Canberra, Melbourne, Perth, Sydney.
Frequency 3 (f3)	148.7875 MHz	Melbourne, Sydney.
Frequency 4 (f4)	148.9625 MHz	Adelaide, Brisbane, Canberra, Melbourne, Perth, Sydney.
Frequency 5 (f5)	148.6125 MHz	Adelaide, Brisbane, Canberra, Melbourne, Perth, Sydney.
Frequency 6 (f6)	148.8125 MHz	South Australian Government Only (Adelaide & South Australia).
Transmission outside main areas		
Frequency 2 (f2)	149.8375 MHz	One or both of f2 and / or f4 are used in various areas.
Frequency 4 (f4)	148.9625 MHz	

Table 4 : <System Name> Pager Frequencies

3.5.1 TRANSMITTERS

3.5.1.1 Urban - Distribution Transmitters

Distribution Transmitters send the encoded pager signals from the <System Name> system to the "paging" transmitters around each major city. <Company Name> has inherited - and still runs - five radio frequency networks, generally one distribution transmitter for connection to each network, though in areas with lower network usage, a single transmitters are configured to broadcast on two distribution frequencies. Configuring transmitters in this way obviously saves outlay on additional sets of paging transmitters. However, the catch is that it is very expensive in terms of capacity – the switching between frequencies for alternative sets of messages takes enough time that total network capacity is significantly reduced. As noted, this is only under-taken on networks and frequencies that posses a large amount of excess capacity.

City	Distribution Transmitters (Number of Networks)
Sydney	5
Melbourne	5
Brisbane	3
Perth	2
Adelaide	2

Table 5 : Transmitters by City

Comparing the number of transmitters listed here with the table of frequencies broadcast in each city, one will find for Brisbane, Perth, Adelaide and Canberra, paging is available on more frequencies than there are networks in those cities. This is due to the multiple-frequency configuration of one or more paging networks in those cities.

3.5.1.2 Rural - Radio Dish Up<Company Name> and the Equatorial Satellite System.

For transmission of paging messages to sites around Australia a Radio Dish / Satellite system is used : the Equatorial Satellite System. Encoded messages travel from <Company Name>'s X.25 network via Telstra's DMS (Digital Metropolitan Service) to the large Radio Dish "up<Company Name>".

3.5.1.3 Paging Transmitters

Both urban and rural (Distribution and Satellite) transmissions are re-broadcast by local paging transmitters as final delivery to customers' pagers. There are around 380 paging transmitters around Australia.

Within each major Australian city between 2 to 5 networks of paging transmitters rebroadcast messages from the matching 2 to 5 distribution transmitters in that city. The approximate number of urban paging transmitters are as follows :

City	Distribution Transmitters (Number of Networks)	Paging Transmitters
Sydney	5	80-85
Melbourne	5	60
Brisbane	3	66
Perth	2	18
Adelaide	2	15
Canberra	(Satellite)	4

Table 6 : Paging Transmitters by City

Over the remainder of Australia, 140 paging transmitters receive messages from the Equatorial Satellite System for delivery to paging transmitters in secondary cities and rural areas.

3.6 <SYSTEM NAME> SYSTEM REDUNDANCY

To ensure optimal levels of service, a large amount of effort has gone into establishing large amounts of redundancy in the <System Name> system. Some of the major redundancies are as follows :

- Mirrored disks on all servers
- Mirrored database on all servers
- Full system backup of each server every night
- Dual HP 9000 servers in the two major call centres : Sydney and Melbourne
- All <System Name> interfaces are connected via terminal servers on the TCP/IP network allowing any <System Name> server nationally to connect and process any interface
- The <System Name> software can automatically switch between the X.25 and TCP/IP network for delivering messages in the event of a network failure.
- Most sites have their own dedicated connection into each encoder
- Automatic re-routing between the <System Name> and corporate LAN/WAN in the event of a router failure

X.25 nodes <Company Name>ed by a primary and backup ISDN/DDN service

- Dual encoders running parallel for each frequency in each site
- Faxing servers which can accept and process faxes from any site
- Call Interflow. The ability to answer and process any call on any <System Name> server
- All HP 9000 servers have a 24 hour, 7 days a week, 365 days a year, 4 hour on site support agreement with Hewlett Packard.

4 Structure of a <System Name> System

4.1 MESSAGE COLLECTION BY <SYSTEM NAME>

4.1.1 TELEPHONE CONNECTION TO A <SYSTEM NAME> CALL CENTRE

Calls to a <System Name> call centre can initiate:

- standard pager messages
- multiple communications, of varying types, for customised accounts
- a retained transcription of the message for later collection
- information being relayed back to the caller and no message being sent

While the Call Centre operator is talking to the caller, the XACOM is collecting technical data from the telephone line. Information from both is used to construct a call record for billing and messaging purposes.

The following is a brief description of each of the <System Name> Call Centre services.

4.1.1.1 Normal Paging

1. A customer phones (or is forwarded to) a Call Centre.
2. The Call Centre operator requests the pager number and a message
3. The message is sent to the pager

4.1.1.2 Personalised Paging

Similar to Normal Paging, with the following enhancements

- Each pager has a unique phone number attached. The customer calls a unique phone number ("in dial") to supply the pager message.
- The details of the pager holder are displayed to the Call Centre operator, who can answer the call appropriately
- The caller doesn't have to know the pager number, only the phone number to call

4.1.1.3 Group Paging

Personalised paging is used in situations where a client wants a number of pagers attached to one unique in dial. When someone sending a pager message calls, the PABX, the CTI server, and skymds01 work together to display, on the Call Centre operator's screen, a list of people and pagers associated to that unique in dial. The caller then identifies the appropriate pager via its holder's name or role.

- The caller doesn't have to know the pager number
- This is useful where a client, say a company, wishes to give out a one single phone number over which all customers can contact all (or a group of) company representatives.

4.1.1.4 Optional Paging/SMS

The Call Centre operator sends the message to a pager and/or an SMS (GSM or CDMA) mobile phone.

4.1.1.5 Fax [FAX]

The message is sent to a Fax number.

4.1.1.6 Group messages

Once the operator takes the message it is sent to a number of separate destinations simultaneously. These destinations can be any combination of pagers, fax machines, special Call Centre activities (via Call Centre reminders) with or without delayed delivery,

4.1.1.7 Mobile Answering to SMS

<System Name> operators answer (forwarded) cellphone calls. A message is taken by the Call Centre operator who sends an SMS message to the cellphone. There are two versions of this scheme :

<Company Name> Messenger – <Company Name>
SurePage – other telecommunications companies.

4.1.1.8 Contact (advanced handling)

The Call Centre operator is displayed with series of screens, each of which prompts them to ask the caller further questions in order to obtain the correct person to send the correct type of message to. This enables complicated determinations to be undertaken :

- Where there are many people, many departments, whom the caller may not know
- Pages, SMS, Faxes etc can be sent based on attributes of the call, e.g.
    - sales enquiry, based on questions about the type of product
    - emergency, fault et.c, based on questions about the situation

Many of the simpler corporate customer handling and help-desk functions can be replaced by using this functionality. More extensive customisation is available via "Customised Forms" (*q.v.*) - Customised Call Centre operator screens.

4.1.1.9 Customised Forms

A fully developed, customisable implementation of the Contact – Advanced Handling above. The screens, their connections, data, and relationships are developed individually for each client. Full Customisation (*q.v.*) is also available.

4.1.1.10 Full Customisation

For complex and unique needs. Specially developed screens, reporting, processes and other responses are constructed by <System Name> staff to meet a client's specific requirements.

4.1.1.11 Guaranteed Delivery

Escalations. To ensure delivery of important messages, various delivery mechanisms can be stipulated. The mechanisms "escalate" from one to the next until delivery is confirmed.

4.1.1.12 Rosters

On the basis of a customised time and date calendar, handle pager and other messages in various ways. Examples may be to simply hold onto the message until 9:00 am tomorrow or, alternatively, to send a fax to the Head Office in Darwin (or both).

4.1.1.13 Advisory

The Call Centre operator is supplied with information to read to the caller, thus acting as a reference service for callers

4.1.1.14 Message Storage

Telephone Message Service. The message(s) taken by the caller are stored for later retrieval. Here <System Name> is acting as a message collection service.

4.1.1.15 Message Retrieval

A service that supplies the messages collected for a customer to that customer.

4.1.1.16 Follow Me

If a pager owner is moving out of the geographical area they usually receive messages in, the messages can be re-routed through the <System Name> system for re-transmittal in the customer's new location.. For example, a businesswoman from Perth may be spending a week in Brisbane. If she uses the Follow Me service she will still receive her pager messages – broadcast in Brisbane – as opposed to not receiving them when broadcast in Perth.

4.1.1.17 Temporary Message

Not an advisory (*q.v.*), but a short term message relayed by the Call Centre operator back to the caller. An example my be "I'm on Holiday, please call …."

4.1.1.18 Booked Page

Pager messages are sent automatically at pre-set times.

4.1.1.19 Reminders

Messages are established to be sent to the Call Centre operators themselves, perhaps to telephone someone or to fax a document. The <System Name> message storage service ("Telephone Message Service") and escalations can also generate reminders as stimulus for call centre action.

4.1.2 AUTOMATICALLY PROCESSED TELEPHONE CONNECTIONS

Call may be processed automatically by computer. In response to the recorded message the caller enters a number on their touch tone phone, the number being processed by <System Name>'s XACOM system and delivered to the target pager. This process, and the devices used to implement it, are generally referred to as IVR : Interactive Voice Response.

4.1.3 DIRECT CONNECTION TO <SYSTEM NAME>

Connecting on a leased line (DDN protocol) or modem Dialup (PSTN), customers send messages in either PET or MERP format. PET is the industry standard "Pager Entry Terminal", which allows an appropriate client machine to emulate a terminal for the entry of pager messages. Most access uses the PET protocol. MERP is <Company Name>'s proprietary access protocol, with extended functionality as compared to PET, such as the ability to retrieve historical messages.

4.1.4 EMAIL CONNECTIONS TO <SYSTEM NAME> : CONCEPTS

With the introduction of advanced email-to-message services came some new concepts. The most important of these are A-party and B-Party

"A-Party" The sender (the A-party) must be registered with both <System Name> and <SYSTEM NAME 3>, and the sender (the A-Party) is charged for the communication. Further, A–Party messages are both validated and security checked before being sent to the client, and a (sender's) PIN number must be embedded in the address of all A-Party emails sent.

"B-Party" The destination of the message (the B-Party) must be registered with <System Name> and the cost is covered as part of the B-Party's existing payment schedule. This type of customer B-Party customers are the

"Blocking File" A file that contains permission information for incoming A-Party Email-to-SMS messages

"priority" queue One of four SMS queues used to deliver Bulk SMS messages. All the SMS are sent from the priority 1 queue before any from the priority 2 queue, all queue 2 before queue 3, all 3 before 4.

The relationship between sending a pager message and sending an SMS should also be clear. One can send a pager message to a mobile phone - appearing as an SMS message but handled by <System Name> as a pager message, or one can send an SMS message – processed as an SMS message (not a page), both these techniques having the same effect.

4.1.5 EMAIL CONNECTIONS TO <SYSTEM NAME> : SERVICES

4.1.5.1 email to Pager / SMS

email to **<pager-ID>@pager.<Company Name>.com.au** will send the message-line and email text to the appropriate "B-Party" pager.

4.1.5.2 email to SMS

Email to **< mobile_phone_number-PIN_NUMBER >@sms.<Company Name>.com.au** will send an SMS message to the GSM mobile-phone with that phone number. Only registered A-Party clients are able to successfully use this address.

4.1.5.3 email to BulkSMS

After being set-up in the <System Name> system, an A-Party client can send an appropriately formatted file containing a list of SMS messages to

< batch_number-PIN_NUMBER >@bulksms.<Company Name>.com.au

for delivery simultaneously to a large number of GMS or CDMA mobile phones. smssms03 constructs the SMS messages from the attached file and inserts them as <System Name> messages in

four ‘priority’ queues, from which mdsms01 extracts the individual messages, passing them to the bulk queue.

4.1.5.4 Web to Pager

Any person with internet access can send free emails to any <Company Name>-issued pager. Go to

http://www.<Company Name>.com.au/solutions/solutions.html

and the <System Name> paging webpage will be in front of you. Enter the pager number and the message. Done? A pager message is on its way, the email-to-pager system (4.1.5.1) delivering the message transparently.

4.1.6 OTHER <COMPANY NAME> SYSTEMS

4.1.6.1 <System Name 2>

<System Name 2> is the bionic, web-base wrapper for <System Name>, running on its own servers, using an Oracle database and Microsoft Technologies. However, <System Name 2> still uses the <System Name> delivery systems to process its messages, and thus is really built ‘on top of’ <System Name>. The <System Name> program skymds96a extracts messages from the <System Name 2> message queue, reformats them into <System Name> messages and posts them to the <System Name> bulk queue.

4.1.6.2 <SYSTEM NAME 3>

<SYSTEM NAME 3>, <Company Name>’s Customer Management and Billing system, may send messages into the <System Name> system, e.g. to ensure that a customer is correctly set up, or to terminate a customer.

4.2 MESSAGE PROCESSING BY <SYSTEM NAME>

Messages from each source are gathered , passed to the Bulk Queue, and segregated into sub-queues from where each message is transmitted to the appropriate recipient. Message delivery is discussed in the next section.

Within <System Name>, messages are held in a standard internal format, all messages being handled in the same way. Only upon delivery (by skymds03) to their external destination are message re-processed into their external format : Fax Message, email, pager message …

4.2.1 STANDARD PROCESSING

[See Diagram 1.2]

Messages arriving in a <System Name> systems are

1. collected and pre-processed by one or more Unix daemon processes,
2. transferred to the “Bulk Queue”, the clearing house for all <System Name> messages. From here details are taken for billing and record keeping purposes,
3. the process sykmds20 distributes each message to the appropriate sub-queue
4. each sub-queue has either a skymds03 process for delivery to the appropriate distribution system (fax server, pager RF encoder, SMS centre …) or one of a couple of other programs for special processing of the message.

4.2.1.1 Message Collection Daemons

For each message source one or more Unix daemon processes collect, check, and parse the messages.

- Computer Automated Message Receipt : skymds08

  The answer-phone message and the collection of the caller’s touch-tone phone keystrokes are handled by skymds08 and the IVR (Interactive Voice Response) server.

- Call Centre Messages : sykmds01

  Multiple copies of the skymds01 daemon are running at any one time, each collecting message data from the various programs used by the call centre operators, from the CTI (Computer Telephony Integration) server, from the PABX, and from the PABX’s in dial processor.

- Direct Connection To <System Name> : skymds70 & skymds02

Messages in the PET protocol are processed by skymds02 while those in the MERP format are handled by skymds70

- email to B-Party Pagers and Mobiles : mdsem01
  When an email is sent to??? @???, mdsen01 converts the subject-line and body to either a Pager Message or an SMS, and adds the message to the bulk queue. mdsem01 processes B-Party (normal) customers' messages only.

- email from A-Party clients, to SMS messages : mdsem02
  Just like mdsem01, except the emails must originate from an A-Party Client. The main difference between A and B party is that A party pays for the calls it sends, B for those it receives.

- email to Bulk SMS : mdssen03 & mdssms01 – A-Party only
  Emails destined to become multiple SMS messages are first collected by mdssen03 for distribution one of four Bulk SMS queues, the choice of which queue being based on the priority requested, or the set-up of the (A-Party) client's account. mdssms01 transfers the messages from the SMS queues to the bulk queue, selecting messages in strict order of the queues' priorities.

4.2.1.2 Bulk Queue

The "Bulk Queue" is the distribution point for each and every <System Name> message, from where messages are re-distributed into their appropriate sub-queues, and their details collected. As the Bulk Queue is a central point in the <System Name> system, and its efficient operation essential for high-speed processing, the queue is implemented as a (large) text file; completely eliminating any database overhead. The Bulk Queue exists as /var/mds/data/mdsqqb.txt on each <System Name> machine.

From the Bulk Queue records are of two types are gathered, CDR (Call Detail Record) for billing and, for retention and recall, a History Record. The CDR billing records are written to /var/mds/data/cdr/CDR*Myyyymmdd* (e.g. CDRM<date removed>0720 - M and O for Melbourne, S for Sydney …), while the history is stored at /var/mds/data/hist/hi*YYYYmmdd.*dat. hi*YYYYmmdd.*idx – the primary history (information *about* the call) and hs*YYYYmmdd.*txt, the secondary histpory, the call message itself.

4.2.1.3 Distribution into Sub-queues

From the Bulk Queue the skymds20 process re-distributes the messages to one of 7 sub-queues :
- for the Fax Server
- for forwarding PET protocol messages to another telecommunications provider
- for sending email
- for deliver to an SMS Gateway
- for transcription by one of 4 encoders en route to someone's pager.
- for later delivery.
- for "Contact", to initiate alternative message processing

… and also into two History files :
- CDR - for billing purposes
- The History Server

Like the Bulk Queue, each sub-queues is implemented as a simple flat file. The history records - where speed of access is less critical are mostly normal database tables.

4.2.2 DELIVERY - EXTERNAL AND INTERNAL

From each sub-queue messages are collected, processed, and delivered : to either an external distribution system, or simply back into the bulk queue.

Delivery to each external process is managed by an instantiation of the skymds03 daemon, one for each of the five external message delivery types :

- fax messages delivered to the fax server
- PET protocol messages for forwarding to another telecommunications provider
- email
- SMS messages delivered to the Optus SMS Centre
- pager messages for encoding and transmittal

Internal re-delivery comes in two forms : The "Contact" process, skymds27, implements modified message handling while the Timed Events process, skymds06, tags or generates messages (of any type) for later delivery.

- skymds27 : The "Contact" server, for advanced handling of messages. Examples its use are :
  - escalations; where messages are re-directed to an alternate delivery method due to delays or other problems
  - rosters; where message handling is changed due to the date and/or time
  - message storage; where, in some cases, messages are to be only recorded and not delivered
- skymds06 : The Timed events process. As described, this manages messages for later delivery.

4.2.3 SUPPORT SYSTEMS

4.2.3.1 alarms

The mdsalarm monitors critical parts of the <System Name> system for a range of problem conditions. Upon appearance of such a conditions an alarm message is posted to be delivered to the appropriate support personal. But "Ah Ha", you say, "what if the Bulk Queue isn't working?". No problem, the message is forwarded to the <System Name> system in another Australian city for processing and delivery. The message (alarm) gets through.

mdsalarm will send alarm messages :

- when disk space is below a set minimum
- when any of the message queues have either grown too big, thus a backlog of unsent messages, or are not sending messages at all.
- when the database update queues are, likewise, too big or not moving.
- when the encoders don't seem to be working normally
- when failures are detected during checking that the ports in use are actually active and data is being received
- when there doesn't seem to be enough network traffic
- when the history files don't look OK
- when any program has crashed, or any program that should be running isn't.
- when there too many reminders – Call Centre Operator 'To Do' tasks.

4.2.3.2 Message History

Two types of historical message data are gathered in the <System Name> system

- billing data : CDRs (Call Detail Records), containing the message length, the sender plus other details
- history data : primary history, containing all the information *about* the call.
- secondary history, containing the actual message contained in the call.

CDR records are written by the Call Centre operators' skymds01 processes.

History records are written by :

- skymds20, as it distributes messages from the bulk queue to the destination queues
- skymds03, during its delivery of messages to various external distribution systems
- skymds67a which inserts new history records to add the data regarding the final broadcast network that an SMS message transported over (Optus, Telstra, Vodaphone)

For more detail of the History and CDR collection processes see section 6.2. and Volume II

4.3 OUTPUT FROM <SYSTEM NAME>

4.3.1 MESSAGING TO EXTERNAL SYSTEMS

In <System Name>, six instantiations of the skymds03 program deliver messages to one or more of the six delivery system types : SMS, fax, Paging, email, SAGRN (the South Australian Government), and PET messages forwarding, which includes messages sent to special corporate clients. A little more detail follows.

4.3.1.1 SMSC skymds03 delivers SMS messages to an SMS Centre, currently Vodaphone or Optus, en route to any GSM mobile phones.

4.3.1.2 Fax Another skymds03 program delivers Facsimiles from <System Name>'s SCO Unix FAX server.

4.3.1.3 Paging A central focus of <System Name>, pager messages in the systems are encoded, optionally sent through the X.25 Network, before being transmitted to the recipient's beeping pager.

4.3.1.4 email Pager messages sent as emails.

4.3.1.5 SAGRN <Company Name> Communication Corporation delivers messages to the South Australian Government's GL3000 system in both a unique format : TNPP, and on a unique frequency: a6 – 148.8125 MHz

4.3.1.6 Leased & Public lines Pager messages are re-transmitted for delivery by other telecommunications companies. These go via the public telephone system (PSTN), leased DDN (digital) or ISDN (multi-use) lines. The forwarding of messages to other networks is also accomplished in this way.

4.3.2 OTHER OUTPUT FROM <SYSTEM NAME>

4.3.2.1 <SYSTEM NAME 3> <Company Name>'s billing and customer management system, <SYSTEM NAME 3>, receives billing and customer management data from <System Name>

4.3.2.2 Call Centre Messages appearing on Call Centre operators' screens, especially those to be read back to the callers are an important output of <System Name>. Advisories and Temporary Messages are examples of these

4.3.2.3 Computer

1. The feed-back that a customer gets during a PET, MERP or rlogin/telnet session
2. The automated answering phone voice upon placing an automated pager call

5 Running <System Name> : Software start-up

5.1 OVERVIEW

The UNIX script 'start.mds' starts up the <System Name> software and it gets the start-up done in three stages:

- Initialization. Set up the environmental variables, specify the terminal for log messages, create the shared memories, and reset message queues.
- Starting up main processes. These processes include inter machine processes, PABX processes, customer remote access processes, internal access processes, encoder process, and other processes.
- Starting up checkup/system monitoring processes. These processes include network monitoring process and message queue monitoring process.

More details will be provided for each individual stage in initialization section, start-up main processes section, and start-up checkup/system monitoring process section, respectively.

5.2 INITIALIZATION

Due to the dependencies among the programs to be started up, the initialization must be done in some order. The first thing to do for the initialization stage is to set up common environmental variables as these variables will be looked up and used by other initialization processes, main processes, and checkup/monitoring processes. Followed are specifying a terminal for log message display, creating shared memories, and resetting message queues.

5.2.1 SETUP ENVIRONMENTAL VARIABLES

The environmental variables to be set are all included in the UNIX script 'start.env'. The following is a list of environmental variables set for host lnkm.

Env_variable	Description	Example
DATABASE_DIR	Some old (Informix) database files	/opt/mds/mds.dbs
INFORMIXDIR	Where the Informix executables are	/usr
LOG_SCR	What terminal to display log messages on	/dev/tty1p0
LPDEST	Where to route line-printer jobs	ml_net_prn1
MANPATH	A long list of paths to man files	/usr/share/man/ …
MDS_BIN_DIR	All the <System Name> executables	/opt/mds/bin
MDS_C_DIR	<System Name> source code files	/opt/mds/c
MDS_DATA2_DIR	SAGRN data goes here	/var/mds/data2
MDS_DATA_DIR	The root of the data directories, queues, database and history.	/var/mds/data
MDS_DOC_DIR	Various <System Name> documentation	/opt/mds/doc
MDS_EMAIL_DIR	the directory for email scripts	/var/mds/data/email
MDS_FAX_DIR	faxes are written here before being transferred to the Fax server	/var/mds/data/bfax
MDS_LOG_DIR	the directory for logs to be written to	/var/mds/data/logs
MDS_MACHINE_ID	M,O Melbourne S,X Sydney A Adelaide B Brisbane …	O
MDS_PRINTER	What to use as a printer	cat >/home/cont1/print.log
MDS_SHMIDS_DIR	Location of the agreed IDs for Shared Memory blocks	/opt/mds/shmids
MDS_TEXT_DIR	Message storage directory	/var/mds/data/text
MDS_TMP_DIR	The temp directory	/opt/mds/tmp
OUR_CITY	What machine we are on.	m1
PATH	The usual huge long list of directories to search	/usr/bin:…
TRUFAX_DIR	Fax binary directory	/opt/mds/trufax

Table 7 : Important <System Name> Environment Variables

The environmental variables listed above can be grouped into two categories. One is host and site independent. The other is host or site dependent. Environmental variables that are site or host dependent need to be modified based on the host and site while site and host independent ones are the same for all hosts. In the list above, only four of them, that is, LOG_SCR, OUR_CITY, MDS_MACHINE_ID, and LPDEST, are site or host dependent. The others are site and host independent.

5.2.2 SETUP TERMINAL FOR LOG MESSAGE DISPLAY

The <System Name> software generates log messages to indicate major events happening. These events include but not limited to:

- Program start up and finish up
- Major program errors such as unable to get shared memory block, TCP/IP server failure, etc.
- Pager message delivery, which include remoter message delivery and local message delivery
- System disk space full
- Database queue updated

Log messages are written into a log file, and at the same time they are also displayed on the logger of the host that the <System Name> software runs on.

Log messages that are displayed on the logger can be redirected to a specific terminal, which makes it easier for support and management staff to view the log messages for system behaviour or potential problems. The program *keepopen* specifies and opens one or more such terminals for all log messages to display.

The program can be set up to display the log messages on multiple terminals. The environmental variable 'LOG-SCR' provides the program with log message terminal or terminals, depending on single value or multiple values contained in the variable. For the example of environmental variables mentioned previously, the display terminal is /dev/tty0y5 only.

5.2.3 SETUP SHARED MEMORY

Initialization stage then sets up shared memories and semaphores for its inter process communication. Several shared memories and semaphores are set up and initialized for different purposes, which include the shared memory for port usage, reminder usage, public holiday usage, global variable usage, and for security group usage. Individual program is invoked to set up the shared memory for individual usage.

- Program mdssetpo. This program sets up the shared memory and initializes a semaphore for port usage. In detail, the program creates a piece of memory if the memory does not exit, and attaches the memory to port shared memory if not attached. Then it initializes the semaphore to indicate that the shared memory has been created and been in use. With the port shared memory and the semaphore setup, an agent's screen output can be duplicated on the supervisor's screen such that the supervisor is able to monitor what the agent is doing.

- Program mdssetrm. This program sets up the shared memory and initializes a semaphore for reminder usage. In particular, the program creates a memory segment if the memory does not exit, and attaches the memory to reminder shared memory if not attached. With the reminder shared memory and the semaphore setup, operators and supervisors will be reminded of some events that are currently happening in the system. Please refer user manual for more details.

- Program mdssetpu. This program sets up the shared memory for public holiday usage. The program creates and initializes the shared memory in the similar manner described above. Then it reads the public holiday master file and fills the shared memory with the message read from the file. Please refer user manual for more details.

- Program mdssetgl. This program creates a shared memory to hold global variables such as master machine, history online, history server, and other history related variables. The program creates and initializes the shared memory in the similar manner described above. Then it reads the configuration file that contains all global variables and then fills the shared memory with the set of global variables.

- Program mdssetse. This program creates a shared memory and initializes a semaphore for security group usage. The program first creates and initializes the shared memory in the similar manner described above. Then it reads the security group configuration file for security group details and fills the shared memory with the security group details. With the security group shared memory and the semaphore setup, the <System Name> system is able to find out which security group the user should be when he or she logs in the system. Please refer user manual for more details.

5.2.4 SETUP COMMUNICATION PORTS AND MESSAGE QUEUES

In addition to the shared memories and the terminal to display the log messages, the communication ports and message queues are also created and initialized in the initialization stage. These tasks are accomplished by invoking three programs.

- Program mdsclrpo. This program clears up end time in all port records to avoid continuous unwanted alarms generation. In particular, the program reads the ports master file and puts the details in a record, and then it gets the ports that belong to the designated city and resets the end date time field to all 9s, which stops the alarm generation.

- Program mdssetty. This program sets up tstty's to manage the dtc terminal sessions. The program achieves this in three steps. Firstly, it opens the port file and terminal server name file; secondly, it reads these files; and finally it sets up the tsttys based on what it reads from the configuration files.

- Program mdschkqu. This program checks the validity of the message queues. The <System Name> software uses several queues for its short message delivery and paging functionality. Some of the queues are bulk queue, paging queue, fax queue, and timed event queue. This program checks the validity of all these queues before the main processes of the software start up.

5.3 START-UP MAIN PROCESSES

As the environmental variables, shared memories, message queues, and the terminal(s) to display log messages, are created and/or initialized properly, the UNIX start up script starts up the main processes for system operation.

There are a few categories of main processes to start up in this stage, the tasks of which are inter machine message delivery, PABX information retrieval, system logging, remote customer computer access to the system, internal access to the system, message encoding, and message processing, respectively. This section addresses each category of the processes in details.

5.3.1 INTER MACHINE PROCESSES

There are two main purposes of the inter machine processes. One is to synchronize database among hosts. When setup in one host changes, the setup in others need to change accordingly. The other is to duplicate information. When one machine is corrupt for some reasons, the information can be obtained from others.

There are two inter machine processes that run on each <System Name> host. One process sends messages to other <System Name> hosts via network. The other receives the messages that are sent from other hosts and uses the received messages to update local database. The messages that are passed among the hosts are mainly the maintenance messages.

- Program skymds23l. This program performs as a message sender. It extracts messages from the management queue and sends them via the network to other machines where the process skymds24l is ready to receive these messages.

- Program skymds24l. This program behaves as a message receiver. It receives messages from the network (sent from process skymds23l running on other machines) and uses these messages to update its local database.

5.3.2 PABX PROCESS

There is only one PABX process running on each host. This process captures the inbound call messages and passes them among <System Name> software components and operators in call centre. With these messages, the screen pop is implemented for call centre agents.

There are two types of messages. One type is call present message and the other is polling messages to make sure that the PABX is still operational.

- Program skymds05g. This program reads through the port file for valid operator positions. Then it connects to PABX/CTI and waits for any call coming, and once a call coming, it searches the database for in dial number, if the in dial number is valid, it saves the in dial record into the port usage shared memory. Finally it triggers the process skymds01 that pops a screen for call centre agent to use.

5.3.3 OPERATOR PROCESS

The operator process is an interface to <System Name> system, through which the operator is able to log into the system, input messages for delivery, and maintenance the system. Refer user manual for functionality the operator process can provide.

- Program skymds00. This program displays the logging screen for operator to input user id and password. After successfully going through the logging process, the program displays operator bulletin board, checks the security levels, and starts the program skymds01 to go to a submenu based on the operator's choice.

5.3.4 CUSTOMER REMOTE ACCESS PROCESSES

Customers who are located remotely from the <System Name> system can access to the system through pager entry terminal (PET) protocol and message entry and retrieval (MERP) protocol.

- Program skymds02. This program is PET protocol receiver. It waits for a connection request from remote customer computer and once the connection has been established the program runs in background as a daemon to serve the customer computer. A customer computer can be connected to the system with a limited timeout, or connected forever without any timeout.
- Program skymds70. This program is MERP protocol receiver. Like the program skymds02, this program waits for a connection from remote customer computer and once the connection has been established the program runs in background to server the customer computer. A customer computer can be connected to the system with a limited timeout, or forever without any timeout. The only different than skymds02 is that the program uses MERP protocol.
- Program skymds08. This program is an interface to XACOM pabx/indial processor, which is another facility for remote customer to connect to the system. The program retrieves messages from in dial cards and passes the messages to pager. A message can be tone only or numeric only, depending on the system setting up.

5.3.5 INTERNAL ACCESS PROCESSES

Internal users, that include call centre operators, system support and management staff, can gain access to the system via internal access processes. These processes include mdsmtps and remterm.

- Program mdsmtps. This program is message transaction protocol (MTP) server. It receives messages from message transaction protocol client and puts the messages received into bulk queue for processing. MTP client may obtain messages from database or other sources, depending on the infrastructure of the system.
- Program remterm. This program provides access to <System Name> system via standard telnet request. This program enables internal personnel, such as service managers, developers, customer service personnel, and salesmen, to gain access to some of the <System Name> applications, to send, retrieve and print messages that are stored in the <System Name> database.

5.3.6 ENCODER PROCESS

Encoder process skymds03 receives messages from queues, encodes the messages it receives, and then sends them to different equipment/systems for delivery. These equipment/systems include

- Encoder that transmits messages to pagers
- Other short message service centre that process or relay messages
- Email server that sends messages as emails
- Fax server that sends messages as fax
- TNPP protocol receiver skymds10 that transmits messages to gl300

This process uses multiple protocols for message delivery. These protocols include

- Telocator network paging protocol (TNPP)
- XACOM protocol
- Pager entry terminal (PET) protocol
- Short message peer to peer (SMPP) protocol
- Transmission control protocol/internet protocol (TCP/IP)

Please refer diagram <System Name>-Software Structure for more details.

5.3.7 PROCESSES FOR MESSAGE PROCESSING

This group of processes are core of the <System Name> software. They work together cooperatively to process all messages that originate from all different sources.

- Program skymds28. This program is network CISAM file server for history data. The program retrieves messages from history file and transmits the messages to the pager's hometown when the in dial and pager are not in the same town.
- Program skymds81. This program is network server for statistics data. The program is used in the situation as the program skymds28, but it retrieves and transmits statistics of either pager or in dial, depending on the configuration.
- Program skymds06. This program processes the timed event messages. The program receives messages from timed event queue and puts them back to bulk queue for reprocessing. The timed event messages include booked call messages, rostered call messages, and escalation messages
- Program skymd27. The program is contact nucleus clearing system. The program extracts contacts (Rosters Escalations TMS etc) from the contacts queue file, and calls the appropriate function to process the contact. If the contact is successfully processed it is deleted from the queue. Otherwise it remains until successfully processed.
- Program skymds10. This program is message delivery module. The program receives messages from IP user datagram and delivers the messages to a TNPP device.
- Program skymds20. This program is a message dispatcher. The program reads messages from the bulk queue and dispatches them into destination queues, such as pager queue, email queue, and short message queue.
- Program mdsem01. This program is the MIME file processor. The program extracts pager data from the MIME file and sends the extracted data to bulk queue for processing.
- Program mdsem02. This program is another bulk MIME file processor. The program extracts SMS data from the MIME file and sends the extracted data to bulk queue for processing.
- Program mdsem03. Like program mdsem01 and mdsem02, this program is a MIME file processor. The program extracts bulk SMS data and sends them to bulk SMS queues.
- Program mdsms01. This program processes bulk SMS queues. The program reads messages from bulk SMS queues and sends them to bulk queue for processing.

5.3.8 OPTIONAL PROCESSES

The processes described above must run on each <System Name> host for proper operation. The following is a list of optional processes that do not need to run on each <System Name> host.

- Program skymds18. This program is an interface through X.25 network to <SYSTEM NAME 3> (Wang) stock and billing computer to receive test messages and file maintenance.
- Program skymds11a. This program is the SMPP server for SMPP version 3.4, which is a clone from program skymds11 for SMPP version 2.0. This program takes messages from skymds03 and sends the messages to other SMS centre.
- Program skymds67a. This program processes packets from other SMS centre and sends the messages to program skymds03. The protocol used for this program is SMPP version 3.4.
- Program skymds96a. This program processes pager messages that are stored in oracle database. The program is a clone from program skymds96.c.

5.4 CHECKUP/SYSTEM MONITORING PROCESSES

The <System Name> system is a real time system that requires operation 24 hours a day, 7 days a week. It is vital that the system is continuously monitored and relevant personnel are informed in the event of a problem or potential problem. The following is a list of the <System Name> components that are monitored for problems or potential problems:

- Disk space. An alarm is generated when the disk space is less than a minimum threshold.
- Message queue. An alarm is generated when the number of unsent messages in a queue is above a certain threshold. An alarm is also generated when no messages in the queue have been sent since the last time it was checked.
- Database update queues. An alarm is generated when the number of unsent message in a queue is above a certain threshold, or when no messages in the queue have been sent since the last time it was checked.
- History files. An alarm is generated when any of the history files is missing.
- Processes. An alarm is generated if any <System Name> processes have aborted abnormally.
- Reminders queue. An alarm is generated when the number of reminders in the queue is above a certain threshold.
- Port inactivity. Ports can be configured to generate an alarm if they are inactive. An alarm is generated when a port has been inactive since the last time it was checked.

There are three processes, mdsalarm, skymds83, and skymds89, for alarm generation when a problem or potential problem is detected.

- Program mdsalarm. This program is system alarm generator. The program checks all processes that are supposed to be running for <System Name> system. If any of the processes has aborted, the program generates an alarm.
- Program skymds83. This program is network delay monitor. The program monitors any delay of the <System Name> network, and generates an alarm if such a delay is detected.

Program skymds89. Gets the correct time and sets the HP system clock every hours.

6 Software / applications comprising a <System Name> system

6.1 CALL-CENTRE OPERATOR / SKYMDS01 BASED PROGRAMS

6.1.1 SKYMDS00

The <System Name> user / Call Centre Operator login screen and base menu. After prompting for a user-name and password, displaying bulletins to the user, skymds00 displays and manages the base <System Name> menu. Upon a choice from this menu or its sub-menus skymds00 spawns the appropriate application while skymds00 remains running, in the background, for the full duration of the user's interaction with the <System Name> system.

6.1.2 SKYMDS00B

Used to edit (add, delete, change) a user's password. Of course you, (i.e. the user group you belong to) must have sufficient rights to use this program.

6.1.3 SKYMDS00D

A utility to edit the bulletin that appears each time you log into <System Name>. Again, one must have sufficient rights to use this program.

6.1.4 SKYMDS00J

This important program displays the client maintenance screen, straight off the main menu of the skymds01 root screen. This program enables the user to perform most tasks associated with a particular client, especially the list of contacts (special processing) services available to the client.

6.1.5 SKYMDS00P

The public holiday screen set-up / maintenance screen. Again, one must have sufficient rights to use this program.

6.1.6 SKYMDS01 – CALL CENTRE OPERATOR BASE INTERFACE

skymds01 manages a suite of programs that comprise the <System Name> Call Centre operators' message entry system. Spawned by the skymds00 logon program, skymds01 presents Call Centre operators with message entry screens, from which it collects and manages the various message entry functionality.

skymds01 uses the Unix 'curses' system to display text-boxes, frames, input/output, overlapping windows and the like on the Call Centre operators' computer screens. Curses, the stone-age version of Windows dialogs, windows and other pop-ups, uses text characters to display text on a screen. For example, a curses dialog may a string of '----------' to draw a line. A discussion of the <System Name> curses system appears in volume 2 of this manual or, alternatively, see the Unix man pages 'curses_intro' and 'curses(5)' for a simple overview.

This program, skymds01, is one of the oldest - started in 1989 - and most central <System Name> programs; the vast majority of <System Name> messages entering through this interface (via <System Name> Call Centres). Further, skymds01's age and importance have lead to a huge amount of work being applied to this/these program(s). The largest (sets of) functions have been split off into sub-modules, many different programmers have made changes and improvements, and skymds01 has, over time, developed into a complicated labyrinth of code.

6.1.7 SKYMDS01 – SUB-MODULES

6.1.7.1 skymds01a

skymds01a manages message retrieval.

6.1.7.2 skymds01b

skymds01b manages booked pager messages. Having received the valid line record of a pager, it allows the user (Call Centre Operator) to add, delete, change and view booked pager messages. The booked message(s) is (are) not written to the bulk queue, instead being written to the timed events table from which they are written to the bulk queue at appropriate times.

6.1.7.3 skymds01d

skymds01d pops-up a simple window in which the user can adjust the name of the pager holder.

6.1.7.4 skymds01e

skymds01e manages temporary forwarding of pager messages to alternative geographical regions[1], functionality known in <System Name> as "Follow Me". If a pager holder wants to receive messages in a different part of Australia they must organise a Follow Me service with <Company Name> Communication Corporation.

6.1.7.5 skymds01f

Sometimes, for various reasons, a customer will want to forward messages, destined originally for their pager, onwards to another pager. The Temporary Alternative Pager Maintenance functionality manages this forwarding.

6.1.7.6 skymds01g

One type of 'contact' advanced processing service is the ability to send single pager messages to many pagers at once: Group Paging. The pager number becomes a reference to a group of pagers. The Group Paging Maintenance routines in skymds01g set-up and control these services.

6.1.7.7 skymds01l

Temporary messages are for Out-of-Office type of requirements "John Smith is on Holiday till the 22nd. If this is an emergency …." and are to be read out to the caller. In this they differ from Infopacks, which are background information only for the Call Centre Operators, and from Advisories, which are permanent messages with data collection facilities : "Please report the number of units sold and the area code". Note that while temporary messages are time stamped with a start and end date, only one temporary message per pager can be in <System Name> at any one time.

6.1.7.8 skymds01p

skymds01p displays a selection of pager details, collected from various database tables, but does not provide functionality to make any changes.

[1] Pager message transmission is local and geographically dependent. A message transmitted in Melbourne will be received in neither Geelong nor Brisbane.

6.2 CALL CENTRE OPERATOR / SUPERVISOR : MAINTENANCE SCREENS

6.2.1 MDSDESCR

<System Name>'s 'destination' (message queue) maintenance program, run from the skymds01 curses-based Call Centre Operator <System Name> interface via the 'file' sub-menu.

6.2.2 MDSEMBLK

The <System Name> email-blocking maintenance screen, used to block individual email senders and/or domains from accessing <System Name> email services. Like mdsdescr, mdsemblk is called from the skymds01 curses-based Call Centre Operator <System Name> interface via the 'file' sub-menu.

6.2.3 SKYMDS34

The reminder parameter maintenance screen, used to control how reminders are handled, e.g. reminders for one call centre will be re-directed to another call centre when the original one is closed. Once can also edit and delete previously established reminder management procedures, all of which are stored in the mdsremp database table.

Called from the <System Name> 'file' sub-menu.

6.2.4 SKYMDS39

Security ('function') code maintenance screen. Using this screen (and program) one can assign access to various parts of the <System Name> system to various groups of users. There are a number of flags, representing the parts of a <System Name> system, all of which are attached to each group in either an enabled or disabled state. As noted, this program toggles the state of chosen flags for chosen groups, allowing or forbidding access.

6.2.5 SKYMDS40

skymds40 enables a user to print out reports detailing the number of calls made to a chosen range of pager numbers, or placed by a particular account number. skymds40 implements a curses based display and is accessible from the 'report' skymds01 sub-menu.

6.2.6 SKYMDS46

Print out reports of <System Name> history data. All-important information about a call is contained in the history, and the data / records can be chosen based on :

- pager number
- in dial number
- event number
- system reference number
- contact id

This is the human interface to the <System Name> history system, using skymds28 to interface to the history server behind the scenes. But what actually happens is that skymds46 passes control to one of the sub-programs skymds46a – skymds46d.

6.2.7 SKYMDS46 HISTORY REPORT SUB-PROGRAMS

skymds46a generate a <System Name> history report, what data appearing being determined by the event number (escalation acknowledgement) or system reference number, a number given out to a customer to identify their particular call or issue in the future.

skymds46b generate a <System Name> history report, the data being selected by the supplied contact id.

skymds46c generate a history report based on the supplied pager number

skymds46d generate a history report based on the supplied in dial.

6.2.8 SKYMDS46E

Print (or display) a report detailing all the batch faxes that are due to be sent from a particular <System Name> server today.

6.2.9 SKYMDS46F

Generate a text-file containing a report with the same appearance as the (customised) form currently displayed on the Call Centre Operator's screen. This program is run when the operator selects "SEND" after successfully filling out the form.

6.2.10 SKYMDS47 AND SKYMDS47A

These two programs implement the 'line' in dial reporting functionality available on the report sub-menu. The first, skymds47, displays and manages the screen, where the user selects what data related to an in dial (of which there is a lot) to display. skymds47 then calls skymds47a to query the database and generate the report.

6.2.11 SKYMDS48 AND SKYMDS48A

Like the skymds47 pair, these two program implement the screen (skymds48) and processing (skymds48a) for the generation of a report listing all (yes, all) the answer phrases used on the <System Name> systems, by state. The report can be optionally emailed to a selected email address.

6.2.12 SKYMDS49 (OVERNIGHT)

Run once a week (2:30am Monday morning), skymds49 tests the set-up of each pager appearing in the contacts (mdsconta), escalation (mdsescal), roster (mdsnros), form (mdsform), and in selected custom solutions. All invalid, expired, inactive, or suspended pagers or set-ups are printed on a report.

6.2.13 SKYMDS74

Batch fax cover pager generator, a 'batch' fax being a single facsimile composed of many individual messages, these messages being 'batched'; together on the fax. Running on the SCO Unix fax server, his application is used to format the cover page for batch faxes

6.2.14 SKYMDS74A

A clone of skymds74 customised to generate the cover pages of faxes for <Other Company 1>., a major client of <Company Name> Communications.

6.2.15 SKYMDS76

Another clone of skymds74, but used to generate faxes that are going to be emailed, also known as emails.

6.2.16 SKYMDS77

skymds77 displays all the recent additions, edits, and deletions made to a range of database tables involved with <System Name> 'contacts' (advanced message processing). Accessed via: skymds01 main screen > 'file' menu > 'audit trail', this application supplies audit trail functionality covering Batch Fax and email, Alternative pager, Booked calls, Follow Me, pager holder name changes, and temporary messages.

6.2.17 SKYMDS78

The fax monitor. Choose 'utility' > 'Fax Delivery Status' to view the faxes pending and sent for a particular in dial, fax number or as an alternative delivery mechanism for a chosen pager number.

6.2.18 SKYMDS79

The program manages the system of network connections with in <System Name>, of which there are many, and they are complicated. Used by other <System Name> programs, notably skymds03 (the message delivery daemon), skymds79 can also be accessed via the 'utilities' menu > 'Network Connection Maintenance'.

6.2.19 SKYMDS83C

skymds83c monitors the network delays between <System Name> sites, and displays the timing data on a screen. The Network Delay Monitor (skymds83c) is accessible via 'utilities' menu > 'Network Delay Monitoring'.

6.2.20 SKYMDS84

Port Maintenance Screen. Every application and daemon in the <System Name> suite is registered against a 'port' in the ports database table (mdsports). This record contains, or is used to hold, a range of other disparate system information about the process, e.g. :

- the semaphore(s) the process listens to
- port type (about 25 types)
- the destination for messages from this application
- the operator than ran the process
- Start and end time. Used by mdschkpr to raise alarms for non-existent programs without end-times.
- … and heaps of other stuff.

The Ports Maintenance screen (utilities > Port Maintenance) is the user interface to skymds84.

6.2.21 SKYMDS85 AND SKYMDS85A

These two programs, the daemon (skymds85) and the user-interface application (skymds85a) are used to alter the status of a fax in the <System Name> system as its delivery status changes. The fax audit file on the fax server is read by skymds85 to ascertain the status of the recent faxes being delivered. skymds85 then calls skymds85a, on a <System Name> server to update the <System Name> database with the delivery status of these faxes.

6.2.22 SKYMDS88

This is the program used to set-up customised solutions developed by <Company Name> staff for particular clients. Accessed via the utilities menu, this is not to be confused with skymds30, which sets up the codes that these systems use internally to access the various forms that they use.

6.2.23 SKYMDS92

skymds92 supplies the user interface to manipulation of the alarm system. Each alarm in <System Name> may page a selection of people, optionally take one or more additional actions, or just be suppressed. The Alarm Maintenance Screen ('utilities' > 'Alarm Maintenance') details whom to page, whether to suppress, how often to send an alarm and a couple of other technical details.

6.2.24 SKYSTAT

As calls are received by the <System Name> Call Centre, skymds05 gathers the answer phrase and a few other pieces of information and writes this information into a large block of shared memory, this shared memory being assigned piece by piece to each of the Call Centre Operators[2] in the call centre. skystat is an application that displays this shared memory on a terminal, in human readable form, so that a supervisor or any other interested bystander can observe the traffic through the call centre.

skystat will be replaced by skystat2 when the new T-Server (C.T.I) is installed.

6.2.25 SKYSTAT02

A currently-working replacement for skystat (*q.v.*) developed for the new T-Server (C.T.I) machine.

6.2.26 TSTTYPID

tstty is the utility that connects data-flows into the hardware ports on a <System Name> server to the appropriate terminal server device (which may be a modem, printer, other network or a terminal). tsttypid, called by tstty, writes the PID of the tstty program into the mdsports <System Name> database table.

6.2.27 TTYMON

Sometimes a supervisor or technician needs to monitor the traffic going across a particular hardware port on a <System Name> server, i.e. the traffic has to be echoed to some other device so it can be observed at one's leisure. tstty, the port data flow management program, has the ability to do this echoing, with ttymon switching the echoing on.

[2] Allocation is made for 2000 Call Centre Operators at each < System Name > site. Obviously most of these shared memory slots are empty at any one time

6.3 CALL CENTRE CUSTOM SOLUTIONS : "EXTERNAL DATABASE PROGRAMS"

Custom solutions can be handled in the <System Name> system by implementing an 'external database'. External database solutions read set-up information from a database table and then call the appropriate programs to implement the application, all within the Call Centre Operator's (skymds01) terminal screen.

There are four remaining <System Name> external database solutions.
1. <Other Company 1>Facilities Response Centre (FRC),
2. <Other Company 2>, a Personal Safety monitoring system,
3. <Other Company 3> (cars) service request management
4. <OTHER COMPANY 4> – A home-services franchise call handling

Now, and in the future, all new custom-solutions will (probably) be implemented within <System Name 2>, a web-based wrapper being built on top of the existing <System Name>. These four remain in <System Name>.

6.3.1 <OTHER COMPANY 1>FACILITIES RESPONSE CENTRE (FRC)

A work and resource management system developed for <Other Company 1>, divided into two areas : "Enquiry" (How are things?) and "Maintenance" (This is how they are). The system uses customised (non-<System Name>) logic to determine what work and resources are needed in a particular area, before using <System Name> (including the 'contact' advanced handling functionality) to send out work request and status messages to the appropriate people.

Program	**Functionality**
cusfrcec	Enquiry Call Processing Screen
cusfrcmc	Maintenance Call Processing Screen
cusfrcwr	Work Request Call Processing Screen.

Table 8: <Other Company 1>: 'Enquiry' Programs

Program	**Functionality**
cusfrcfm	Facilities Manager Maintenance Screen.
cusfrcet	Enquiry Type Maintenance Screen.
cusfrcen	Enquiry Code Maintenance Screen.
cusfrcsi	Site Maintenance Screen.
cusfrces	Enquiry Code / Site Relationship Screen.
cusfrctr	Trade Code Maintenance Screen.
cusfrcma	Maintenance Code Maintenance Screen.
cusfrcsp	Service Provider Maintenance Screen.
cusfrcts	Trade/Site/Service Provider Relationship Screen.
cusfrcsc	Site/Cost Centre Maintenance Screen.
cusfrcer	Build Escalation records for Facility Response Centre

Table 9: <Other Company 1>: Maintenance Programs

Other	
cusfrccg	Cost Centre / GL Maintenance Screen.
cusfrcch	Change the Service Provider code in the Maint Log.
cusfrcfx	Work Request Fax

Table 10: <Other Company 1>: 'Other' Programs

6.3.2 <OTHER COMPANY 2> – PERSONAL SAFETY MONITORING SYSTEMS

<Other Company 2> enables companies to keep track of employees, especially those working at night or in dangerous situations. The company or worker calls a <System Name> centre with the expected schedule of a job. Once the job is completed the worker again calls a <System Name> call centre to signal "I'm OK, job complete". If he or she fails to call in, an alarm (a "duress") is raised and sent via <System Name> as an escalation to one or more respondents – until an "I've got the message" reply is received. The idea is that these respondents will act appropriately given the non-response of the original 'worker'.

Program	Functionality
cussaf01	<Other Company 2> Login Call
cussaf02	<Other Company 2> Logout Call
cussaf03	<Other Company 2> Change/Extend Time
cussaf04	<Other Company 2> Change Detail
cussaf05	<Other Company 2> Menu
cussaf06	<Other Company 2> Reference Enquiry
cussafpr	<Other Company 2> Profile Screen
cussafsp	<Other Company 2> Service File Screen
cussaflg	<Other Company 2> log file Enquiry
cussafrp	Report Generator.

Table 11: '<Other Company 2>' programs

6.3.3 <OTHER COMPANY 3> (STARTEL)

<System Name> handles all feedback and service request calls for <Other Company 3> in Australia. While, previously this was handled by a STARTEL system, some rare, unsupported platform with proprietary applications, now the <Other Company 3> functionality has been ported to and integrated with the <System Name> system.

The core <System Name> functionality sends simple customer feedback messages to the <Other Company 3> system. Service requests however, needing to locate a number of <Other Company 3> dealerships near the caller's reported location, as well as supporting specialised call-reporting to <Other Company 3> head-quarters, need the additional functionality provided by the programs listed in the table below.

Program	Functionality
cussta01	Call Processing Screen. – called from skymds01.
cussta02	DETCO/<OTHER COMPANY 3> –Dealer Maintenance Screen – called from skymds00.
cussta04	DETCO/<OTHER COMPANY 3> – Location Maintenance Screen – called from skymds00.
cussta03	fax to DETCO or <OTHER COMPANY 3> the details of a call processed by a Call Centre Operator

Table 12: '<Other Company 3>' ('Startel') programs

6.3.4 <OTHER COMPANY 4> HOME SERVICES FRANCHISE

The fourth customised service supplied by <System Name> is support for the VIP home services franchise system. <System Name> processes all calls to VIP before the call details are sent on to the VIP marketing department. Separate maintenance functionality is accessible via the customer menu on the skymds00 start-up screen.

Program	Functionality
cusvip01	VIP Home Services Screen.
cusvip02	VIP Bulk Data Entry Screen.
cusvip03	VIP Database Maintenance Screen.
cusvip04	VIP Database Deletion.
cusvip05	VIP Suburb List.
cusvip06	VIP Database Report Initiator Screen
cusvip06a	VIP DATABASE Report

Table 13: 'VIP' programs

6.4 COMMUNICATION PROGRAMS : EXTERNAL

6.4.1 MDSMTPS

mdsmtps implements the Message Transaction Protocol Service (MTPS) protocol enabling external customers to send messages via the <System Name> system. Running as a daemon it awaits for a client connection, processes their message, sending the message to the Bulk Queue and an ACK (acknowledge character) back to the sender.

While similar to the other external connection support systems (skymds02, skymds70, and skymds10), mdsmtps is used specifically as part of the "Traka" service.

6.4.2 REMTERM

remterm provides remote access to the <System Name> system over a telnet connection, enabling a <Company Name> sales rep. to demonstrate <System Name> to a client, or a technician work on the system, while away from the building housing a <System Name> server.

6.4.3 SKYMDS02

skymds02, implementing the P.E.T. protocol, runs as the P.E.T. ("Pager Entry Terminal") receiver for the <System Name> system, a standard protocol for interfacing to paging systems. Most customers use this protocol when connecting via computer to the <System Name> systems, enabling them to send message from their computers, possibly without human interaction, and to activate limited message retrieval capabilities. This PET protocol has been extended by the <Company Name> proprietary "M.E.R.P." for additional retrieval capabilities.

Each instance is executed from the <System Name> start-up script "start.mds", a number of individually configured programs running concurrently, each having been initialised with different ports and (possibly) other command-line options.

At the time of writing between 8 and 27 instances of skymds02 run at any one time on the various Australian <System Name> systems.

6.4.4 SKYMDS10

skymds10 provides a TNPP (Telelocator Network Paging Protocol) <Company Name> between <System Name> and an external site.. Pager messages are delivered both from and to the <System Name> system via this daemon, with limited additional capabilities: specialised history and billing tracking, and some custom message handling.

At the time of writing this has been implemented for only the South Australian Government Radio Network (SAGRN) running out of Adelaide

6.4.5 SKYMDS11T AND SKYMDS11R

skymds11t transmits SMS messages from <System Name> to an external SMS Centre (SMSC). Its partner in SMS, skymds11r, receives the replies back from the SMSC for filing in the <System Name> database and billing systems. Both of these run as daemons, constantly monitoring the message queue or the SMSC, and transmitting the messages as appropriate when need be.

It is these two programs that enable the capabilities of <System Name> to extend to and act as an SMS system.

6.4.6 SKYMDS12

skymds12, a program that simulates SMSC traffic and behaviour, is used to test the skymds11t and skymds11r SMS transmission and reply receipt programs, both programs being able to bind to and interact with it as they do a genuine SMSC gateway. No messages are sent to mobile phones, but the behaviour of the tested programs and their generation of PDU records are noted and tested.

6.4.7 SKYMDS23G

A variant of skymds23, skymds23g extracts messages from the maintenance queue (mdsqqm.txt) and sends them (via skymds10) to the SAGRN GL3000 network in South Australia. skymds23g runs only on lnkg, the <System Name> server dedicated to servicing the South Australian Government Radio Network (SAGRN).

skymds23g ensures that, like the rest of the <System Name> system, the South Australian Government is kept synchronised with changes originating from other parts of Australia.

6.4.8 SKYMDS70

skymds70 implements the MERP interface for the connection of (remote) customer computers to the <System Name> system. MERP (Message Entry and Retrieval Protocol) receives messages from remote computers and adds them into Bulk Queue for transmission to their destination, which may be a mobile phone, a pager, an email box, a fax machine, or (possibly) SAGRN[3]. An alternate protocol is pager entry protocol (PET), implemented by skymds02 (*q.v.*)

M.E.R.P. is a <Company Name> custom improvement over the (standard) P.E.T. protocol, implementing mainly advanced message retrieval services.

6.5 COMMUNICATION PROGRAMS : TO / FROM OTHER <COMPANY NAME> SYSTEMS

6.5.1 SKYMDS18

skymds18 send messages into <System Name> from <Company Name>'s billing system, the Wang-based '<SYSTEM NAME 3>'. Basically it casts streams of data arriving on the communication port into a data structure, reads a 'what-am-I' field in this data structure, before passing the collected data to the appropriate function (from 30 or so) for processing *in* or manipulation *of* the <System Name> system. Its partner program, skymds25, sends message from <System Name> to the Wang.

<SYSTEM NAME 3> is in Sydney; thus skymds18 only runs on lnks. skymds25 is also only found in Sydney.

One should note that most of the functionality and facilities of <System Name> are thus available on the WANG via the use of skymds18 and skymds25.

6.5.2 SKYMDS24S

skymds24s retrieves maintenance messages sent over the network by instances of skymds23l at various <System Name> sites, using this data to update the Oracle database tables on the GDR server for use by the SAGRN system. Only data from the mdsline (in dial information), mdspager (pager information), and mdsclien (user information) database tables are transferred to the South Australian Government's system by skymds24s.

6.5.3 SKYMDS25

skymds25 sends messages and updates from the <System Name> system to <Company Name>'s <SYSTEM NAME 3> billing system. Compared to the extensive messaging capabilities built in skymds25's partner program skymds18, skymds25 itself only sends :

- temporary messages
- SMS phone addition, deletions, and changes
- pager holder name changes
- answer phrase changes
- follow-me updates (messages being broadcast in an alternative area)
- SAGRN (Sth Australian Government) error code changes

to the Wang. Each of these is handled by a different function, the remainder of skymds25 consisting of support utilities.

[3] SAGRN : the South Australian Government Radio Network

6.5.4 SKYMDS96A

skymds96a is the '<System Name 2>'–to–<System Name> interface, reading messages from <System Name 2>'s message queue – an Oracle database table – into <System Name>'s bulk queue, ensuring that all <System Name 2> messages are delivered normally via the <System Name> distribution systems.

<System Name 2> is the bionic web-based wrapper for <System Name>.

6.6 COMMUNICATION PROGRAMS : INTERNAL

6.6.1 SKYMDS03

skymds03 manages the delivery of <System Name>'s messages to a range of external destinations. 30 to 40 (differently configured) instances of skymds03 run at any one time on a <System Name> installation, sending messages of many different types (fax, email, pager messages, …) to an arbitrary number of destinations. skymds03 includes 10 sub-modules when built, and, after the huge call-centre message entry system, skymds03 is the largest program in the <System Name> suite.

Currently skymds03 sends messages to the following destinations :

- <System Name> Encoders. These messages are delivered as pager messages.
- The Fax server(s)
- Email servers
- SMSCs, (Short Message Service Centres), for the delivery of SMS text messages.
- SAGRN : The South Australian Government Radio Network
- External paging systems. Some messages are forwarded from <System Name> to external parties' paging systems
- forwarding messages to instances of skymds03 running on <System Name> systems in other cities

To actualise these deliveries, skymds03 uses multiple protocols. These include:

- TNPP : Telelocator network paging protocol
- XACOM protocol – for interface to XACOM brand devices
- PET : Pager entry terminal protocol
- SMPP : Short message peer to peer protocol
- TCP/IP : Transmission control protocol/internet protocol

6.6.2 SKYMDS08

skymds08 is an interface to XACOM pabx/indial processor, used for processing touch-tome phone automatic pager alphanumeric input. If, when a customer calls a <System Name> centre, a computerised voice answers and the customer enters a series of digits to be sent to the pager, then skymds08 is the program managing these interactions. One copy of skymds08 runs as a daemon at each <System Name> site.

6.6.3 SKYMDS23

skymds23 reads <System Name> messages from the local maintenance queue, passes them through the <System Name> network to where they are read by one or more non-local instances of skymds24. skymds24 writes these messages to the (non-local) <System Name> system. This system (skymds23, skymds24) is used to keep the databases at each <System Name> site synchronised with each other. All database updates are replicated in this way, so that *each* <System Name> site contains all the data necessary to run *all* the other <System Name> sites.

6.6.4 SKYMDS24

skymds24 receives messages and updates from non-local <System Name> servers and updates the local database system with this new information. skymds24 reads these messages through a socket over the <System Name> LAN, and is the sister program of skymds23, skymds23 sending the messages and updates that skymds24 receives and processes.

Together, skymds23 and skymds24 enable any <System Name> site to process, support, back up, or replace any or all of the other <System Name> sites.

6.7 COMMUNICATION PROGRAMS : EMAIL AND SMS

6.7.1 MDSEM01

mdsem01 extracts B-Party SMS and Pager messages from the email queue – an Informix database table – processes them, and adds them to the standard <System Name> Bulk Queue[4] for transmission to their destination – a mobile phone or a pager. For your information, B-Party messages are free to send, the cost being covered by the receiver: by a set charge, or on a per-message or per-character basis. mdsem01 is the B-Party message processing application.

6.7.2 MDSEM02

mdsem02, running as a background daemon, extracts A-Party (paid for by the sender) SMS messages from the email queue database table (mdsemqu), processes these messages, and inserts them in the bulk queue.

One should note that the message itself isn't extracted from the mdsemqu queue, rather the file-name of the text file containing the message is retrieved. Also mdsem02 only processes SMS messages, while mdsem01, the B-Party email processor, produces both pager messages and SMS messages – as pager messages sent to a mobile phone.

mdsem03, the Bulk-SMS daemon, is an extension of the mdsem02 program.

6.7.3 MDSEM03

mdsem03, the Bulk-SMS daemon, produces collections of Bulk-SMS messages, the details of which have been pre-established by (the attachments to) messages sent to the mdsemgw daemon. Running as a background daemon, mdsem03 extracts the file locations of A-Party[5] Bulk-SMS emails from the email queue (database table mdsemqu), processes the *attachments* in these emails, and inserts the generated SMS messages them into one of the four 'priority' SMS queues. Another program, mdsms01, extracts the messages from these priority queues and posts them to the Bulk-SMS Queue for eventual delivery. Bulk-SMS messages are SMS messages sent to a number[6] of cellphones at once.

Please note that :

- it is the file-name of the text file containing the email message – not the message itself – held in the mdsemqu queue record.
- the email messages must contain attached files. It is these attached files that contain the instruction for the generation of the Bulk-SMS messages.

mdsem03 is a bulk SMS extension of mdsem02, the email-to-SMS daemon.

6.7.4 MDSEMGW

mdsemgw is the <System Name> email gateway daemon. Its task is to collect, manage and supply emails for mdsem01, mdsem03 and mdsem03, the A-Party, B-Party, and Bulk SMS email processing daemons. mdsemgw does this by writing messages to the mdsemqu queue, each message containing, among other things, the file path and name of a text-file into which mdsemgw has written the email. Once an email record has been written to the email queue, the processing daemons, mdsem01, mdsem02, and mdsem03 can process the email at their leisure.

mdsemgw is run by sendmail. Collecting sendmail's email messages on its stdin, it checks each message before writing a the record to mdsemqu, the email message queue database table.

[4] Not to the SMS Bulk Queue. This is for A-Party messages only.
[5] paid for by the sender
[6] up to 50,000. Small numbers of SMS messages in a "Bulk" SMS send are possible, e.g. 5 SMS messages to one's five associates, but the minimum limit is currently set at 10

6.7.5 MDSSMS01

mdssms01 is the <System Name> application that :
1. extracts individual Bulk SMS messages from each of the four priority queues
2. posts them to the bulk queue.

Doing little checking and processing, mdssms01 is a rather simple program.

6.8 DAEMONS (NON-COMMUNICATION)

6.8.1 SKYMDS06

skymds06 processes "timed events", that is <System Name> messages that need to be delivered (or not delivered) at or between certain times only. The list of these is something very similar to :

Timed Event	Description
Booked Calls	send a pager message at a set time
extra messages	(as per Booked Call)
delayed messages	(as per Booked Call)
follow-me start messages	(as per Booked Call)
follow-me finish messages	(as per Booked Call)
Reminders	Messages to the Call Centre Operators
Manual Reminders	Messages to the Call Centre Operators
Escalations	Messages that get new delivery if not acknowledged in time
Faxing Daily (et al) reports	faxing info to someone daily
emailing Daily (et al) reports	emailing info to someone daily
<Other Company 2>	sending an alarm message if a customer hasn't called in (<Other Company 2>)
A-Party email report	send a report on A-Party (possibly bulk) emails or SMSs

Table 14: Timed Events in <System Name>

6.8.2 SKYMDS20

skymds20 extracts <System Name> messages from a bulk queue and assigns them to the correct copy of skymds03. A running <System Name> system will have 2 copies of skymds20: a SMS skymds20, and an 'everything-else' skymds20. The former directs messages only to the SMSC destination queue, while the later delivers each message to the correct instance of the 30 or so copies of skymds03.

6.8.3 SKYMDS27

"Contacts" are the <System Name> messages requiring advanced handling. These messages ("contacts") pass through the bulk queue, to skymds20, which passes them on to this program – skymds27 – to implement the advanced handling.

Contact	Description
Advisories	Messages to callers, with their (simple) collected responses
Call Patches	Connecting a caller to a destination. Like a telephone operator.
Escalations	Change the method of message delivery until receipt is acknowledged
Emails	Send an email from <System Name> to an external destination
Faxes	Send a fax
Forms	Process the data entered on a customised form
Group Paging	Send a pager message to multiple pagers
Paging – Standard paging	Send a normal pager message
TMS – answer phone	Record a message from the user for later recall for the client.
Text processing – answer phone	Same as TMS, but stores the messages in a text file, the file itself delivered.
Reminders	Add a reminder message to the Call Centre's reminder queue
Reminders : Supervisor Reminder	Add a reminder for a Call Centre supervisor to read and act on
Reminders : Manual Reminders	Nothing. No processing required for manual reminders.
Reminders : Phone Dummy Rem'rs	Send a reminder into the void – for some reason
Rosters	Outside certain times apply alternative processing to the message.

Table 15: 'Contact' (advanced processing) in <System Name>

Messages representing contacts of each of these types are delivered to skymds27, which, calling one of a number of sub-functions, processes the contacts appropriately. 'processing the contacts appropriately' can involves :

- sending one or more pager or contact messages back to the bulk queue : if it is a contact it ends up, in its new form, in skymds27 again.
- generating new simple pager or contact messages and iteratively processing these within skymds27.
- posting the processed message to its final destination, e.g. the fax queue
- posting the processed message to the maintenance queue.

6.8.4 SKYMDS81

skymds81 is the pager and client statistics server, supplying usage data to non-local <System Name> sites. Being spawned in response to each data request, typical usages include 'Online Pager Stats' and 'Online Client Stats' retrieving and reporting on the number of messages made by a user or pager over any period up to a year in the past.

6.8.5 SKYMDS83

In order to monitor network speeds and delay, skymds83 reads messages being sent through the encoders, finds the matching record in the <System Name> history and compares the time difference between the two. This indicates the time it took the message to travel from skymds20, through (usually) skymds03, and to the encoder. If the time period is greater than five minutes, and alarm is raised.

6.8.6 SKYMDS83D

Similar to skymds83, skymds83d monitors the speed at which messages are delivered to the SAGRN network, raising an alarm if the delay is greater than 2 minutes and 30 seconds. skymds83g only runs on the lnkg server, which is dedicated to the South Australian Government.

6.8.7 SKYMDS89

Running constantly as a daemon, every hour skymds89 reads the correct time from a 'Zach' device and sets the time on the <System Name> servers appropriately. If the time has drifted more than 15 seconds over the previous hour an alarm is raised.

6.9 OVERNIGHT PROCESSES

6.9.1 CHKFRCEL

Each day chkfrcel cleans out old FRC (<Other Company 1>) Enquiry log records, old being 200 days or more. The FRC (<Other Company 1>) is a custom <System Name> solution provided for that company. Like all the other overnight / daily process chkfrcel is run from its own script (start.chkfrc) run by cron.

6.9.2 CHKFRCML

chkfrcml cleans out old FRC (<Other Company 1>. custom database solution) records to free up disk-space. This utility is scheduled on an ad hoc basis or as a cron job depending on demand.

6.9.3 MDSCDREM

This application cleans up local CDR (billing) records after they have been transferred to the appropriate accounting system(s). Generally CDRs are kept for 7 days locally before being deleted.

6.9.4 MDSCDREP

mdscdrep is a utility to produce reports for a particular day from the merged collections of CDR (billing) records. Used by the <System Name> support and accounting personnel for the production of miscellaneous and daily reports.

6.9.5 MDSCHKGL

Another old-data clean-up utility, mdschkgl, deletes old SAGRN files on the SAGRN <System Name> server, files that have previous been delivered to or from their GL3000 network system. Run at 4:20 am every day mdschkgl goal is to recover disk space from these out-of-date files.

SAGRN is the South Australian Govt Radio Network, an organisation that maintains a special, in-depth relationship with <Company Name> and the <System Name> system.

6.9.6 MDSCHKAU

Run once a day (from a script triggered by cron), mdschkau deletes all the records from the audit-trail database table (mdsaudit) that are older than 30 days. The audit trail keeps simple records of each transaction and change actioned by the <System Name> system.

6.9.7 MDSCHKES

Similar to mdschkau, mdschkes deletes old escalation records from the escalation transaction table. Also run once a day (by cron via the start.chkes script), mdschkes deletes all escalation records older than seven days.

6.9.8 MDSCHKPR

An important program, mdschkpr, ensures that all the processes that should be running are, in fact, running. It does this by reading through the start.mds <System Name> start-up script and testing that each program listed there-in is currently running. mdschkpr is actioned once an hour (also by cron via the start.chkpr script) and raises and alarm upon finding any processes not running.

6.9.9 MDSCHKTM

This application removes old temporary messages, as well as temporary messages assigned to non-existent pagers and, like the other clean-up programs it is run by cron via its own script program (start.chkfils). Temporary messages are temporary replies phrases read out by the Call Centre Operator's in response to a caller. Typical uses would be "John Smith is on holiday until the 5th of March. If this is urgent …"

6.9.10 MDSDNCLR

As Call Centre Operators take calls, hourly and daily counts of messages initiated are tallied, for each operator, in a block of shared memory set aside for that purpose. These tallies are used to generate reports on the Call Centre Operators as a group and for monitoring the performance of individual operators. However once an hour the hourly totals have top be reset and daily the daily totals have to be removed. mdsdnclr resets the counts back to zero, being run (from cron) via the script start.dnclr-h when resetting the hourly totals and start.dnclr-d for the daily totals.

6.9.11 MDSHIMER

Each <System Name> server generates local history files, recording the (local) messages generated by that server. Once a day mdshimer, run by cron on the history server, lnkh, merges these history files into a daily global history file stored in the /var/mds/data/hist directory on the history server. The history files and systems are discussed in detail elsewhere.

6.9.12 SKYMDS50

At 12:30pm every night skymds50 is run to print out the daily message counts and statistics. After producing messages by category and by queue skymds50 then, after storing the information, re-zeros the counts in preparation for the new day. skymds50 is run from the <System Name> End-of-Day script 'start.eod'.

6.9.13 SKYMDS51, SKYMDS52

These two programs zero out call counts at the end of each month (skymds51) and year (skymds52) in the following five <System Name> database tables:

- mdspager: the pager database table,
- mdsports: the ports database table,
- mdspassw: the password database table,
- mdsline: the 'line' – indialed telephone number – database table,
- mdsdests: the message queues, 'destinations', database table.

These two are run by the end of month and end of year shell scripts; start.eom and start.eoy, respectively.

6.9.14 SKYMDS55

Every morning (3:00am) skymds55 appends the running total of calls to each pager and for each client to the 'daily' total, and this daily total to the total for the current calendar month. Basically skymds55 builds up (some of) the statistics that skymds50, skymds51, and skymds52 clear out at the end of the day, month and year.

6.9.15 SKYMDS57, SKYMDS59

This process updates the <SYSTEM NAME 3> billing systems customer ('client') data, based on changes in the <System Name> in dial ('line') database table. At 10:30pm (22:30) on the first day of each month skymds55 reads the <System Name> line table and deletes out all records in the (<System Name>) client table that have no in dial record. Obviously <System Name> can no longer service these customers. It then transfers these changes to the Wang computer system that runs <SYSTEM NAME 3> so that the <SYSTEM NAME 3> records can be updated.

skymds59 goes through the same process and skymds57, but send a monthly update of the <System Name> *pager* data for Wang to process.

6.9.16 SKYMDS58

skymds58 cleans out old phone records (phone numbers for <System Name> Call Centre Operator's to call). This involves two related tasks :

- deleting out phone records — if the phone number can not (no longer) be found in the mdsline table.
- Find any out standing messages to be sent to this phone number and, by using the <System Name> escalation process, ensure that the message will still be sent.

6.9.17 SKYMDS55

Like mdsdnclr, skymds55 re-zeros call counters in the mdsstats (pager statistics) and mdsclsts (client statistics) database tables. Examples are counts of calls taken by the operator (like mdsdnclr), external calls made, internal calls and faxes. Every day cron runs the script start.paclstat, which in turn runs skymds55.

6.9.18 SKYMDS60

Another clean-up program, skymds60 deletes out old records from a number of database tables. Run once a day (by cron via the start.60 script) skymds60 deletes

- inactive mdsline (in dial) records older than 60 days,
- inactive mdspager (pager) records older than 60 days
- delivery status records older than 10 days
- public holiday records that are more than 3 days in the past

6.9.19 SKYMDS60A

An accomplice to skymds60, skymds60a finds all the inactive in dial records in mdsline and removes the related records from associated tables. The tables from which records may (will) be deleted are :

Table	Description of Table contents
mdsadvis	advisories : special messages for the user with data collection (e.g. "How many units sold?")
mdscallp	call patches: where the Call Centre Operator connects an incoming call to an (outgoing) number they have dialled
mdsconta	contacts: advanced processing services set up for individual customers
mdsemail	email(s) that may be sent in response to calls by customers
mdsescal	escalations : instructions for resending the message via different means until
mdsfax	fax messages and transmission data.
mdsforms	forms : highly customised interfaces and message collection set up for certain clients
mdsrem	reminders : messages directed to Call Centre Operator instructing them to do certain things, quite often call a customer on the telephone
mdsnrost	rosters : changed call handling at various times and on various days
mdsphone	Telephone Message Service : The plain old answering service. "Mr Jones is out. Can I take a message?"
mdstext	Telephone Message Service : Messages stored in a text file for later delivery (as a text file).
mdsinpak	infopack : A message or pieces of information displayed to the Call Centre Operator to do with this number. For example, "Most callers are elderly, speak very clearly", or "Offer to transfer the caller to a hospital. Do NOT give ANY advice"

Table 16: Database tables cleaned by skymds60a

So, when an individual or company ceases to use <System Name> services, their set-ups have to be removed from all these tables.

skymds60a, like skymds60, is run once a day by (cron via) the start.60 script

6.9.20 SKYMDS83A

The mdsdelay database table is used to track network response times, especially in order to detect and act on network delays. skymds83a cleans out these network timings after a specified time, currently 31 days. skymds83a is run daily by the start.83a script (via cron).

6.10 UTILITIES : OPERATOR / SUPERVISOR UTILITIES

6.10.1 MDSFAXRP

A program that produces a 'fax delivery status' report for the given period, each report containing details of :

- faxes queued
- faxes delivered
- faxes cancelled
- faxes expired

mdsfaxrp is also run once a day by the start.faxrp script to report on any unsent faxes.

6.10.2 MDSFRCML

This program simply updates the status of a collection of FRCML records from 'Open' (O), to 'Reconciled' (R), and is part of the <Other Company 1> collection of custom solutions supplied to that client.

6.10.3 MDSTYMON

A Call Centre supervisor uses this utility to remotely monitor an operator's terminal, and thus his or her performance.

6.10.4 SKYMDS19

This program is used to establish a set of batch (bulk) faxing rules to be attached, possibly later, to a selected in dial or client. Aspects such as fax frequency (daily, which days?, monthly, or once off) and the type of content can all be set using skymds19. Outputs from other services (i.e. <System Name> contacts and customised forms) can also be included on the batch faxes. One must have sufficient rights to use this program.

6.10.5 SKYMDS19A

Derived from skymds19, this program establishes the sending of batch emails, possibly transferring the message originally sent to a pager into an email that is then sent to multiple destinations.

This program has been superseded by the <System Name> mdsem* programs that compose the <System Name> email processing subsystem.

6.10.6 SKYMDS30

skymds30 is used to prepare the codes for customised solutions developed by <Company Name>, these codes being required before the customised solution is added to the <System Name> system. skymds30 is used by the <Company Name> service delivery team as they are setting up a new service, though most (all?) new developments of this type, i.e. custom solutions, are now being developed in <System Name 2>, so this utility is probably seldom used.

6.10.7 SKYMDS31

A back-office utility for printing the details of a particular in dial, similar to the skymds48 and skymds47 utilities available to Call Centre Operators and general users.

6.10.8 SKYMDS32

The group-level security program, used to set up security groups and access rights available to members of this / these groups. Used by the service delivery team when they set up a new application, client, customised solution or service, skymds32 is the group-level partner of skymds39, the program that manages the codes used by skymds32.

6.10.9 SKYMDS33

Called by selecting 'Operator List Report' on the 'utilities' menu of the skymds01 top-level screen, skymds33 generates a report detailing all the operators in a particular city that are members of the selected security group. Used mostly by Service Delivery when planning security requirements for new client services.

6.11 UTILITIES : BACK-OFFICE UTILITIES

6.11.1 CI

<System Name>'s various pieces of peripheral computational machinery (modems, the Call Centre Operators' terminals, and occasionally programs) communicate with, and are generally controlled by, the <System Name> server via the use of 'device files'. These are files in the server's /dev/ directory used as proxies for external devices connected to the computer, or connected to a terminal server that is in turn connected to the server.

ci sends and receives streams of characters to and from the device files on a server, and is used to send particular control characters or text to a device, check ports on the terminal server, or test that a device connected to the machine or terminal server is OK.

6.11.2 COPYLINE

This utility enables a Call Centre Operator to copy a customer's <System Name> set-up into a duplicate, so the operator can then alter this duplicate to create a new, similar, service. This can greatly ease the work involved in establishing some new customers.

Records are duplicated in the following tables :

Table	Description of Table contents
mdsconta	contact – advanced processing
mdsforms	forms – customised solutions
mdsescal	escalations – multiple delivery mechanisms
mdsnrost	rosters – different delivery mechanisms at different times and on different days
mdsadvis	advisories – messages to be read out to customers with optional data collection
mdsinpak	infopack – information displayed to the Call Centre Operator about the caller
mdsfax	faxes
mdsschd	search data – for finding the correct pager from incomplete information supplied by the caller
mdsschp	similar to mdsschd
mdsemail	email
mdscallp	call patch, where a Call Centre Operator connects a caller to an out-dialled line that they ring, like a telephone exchange operator

Table 17: Tables written to by copy_line

6.11.3 CPER, MVER

On some Unix installations previous versions of edited files have an (incrementing) number appended to their filename, each changed version of the file thus being both saved and indexed.

For example :

	myFile.c	becomes	myFile.c.1	after it's first edit
and	myFile.c	becomes	myFile.c.4	once myFile.c.1, .2, and .3 already exist

cper implements a version of behaviour where copied files have numbers either incremented or appended to their name. mver does the same for moving files. These are both used in <System Name> to save a backup copy when editing a <System Name> source code file.

6.11.4 DEL_LINE

In a way the opposite of copyline, del_line cleans out all the records related to a customer based on their (no longer serviced) in dial. Records are deleted from the following tables (the same as for copyline) :

Table	Description of Table contents
mdsconta	(contact – advanced processing)
mdsforms	(forms – customised solutions)
mdsescal	(escalations – multiple delivery mechanisms)
mdsnrost	(rosters – different delivery mechanisms at different times and on different days)
mdsadvis	(advisories – messages to be read out to customers with optional data collection)
mdsinpak	(infopack – information displayed to the Call Centre Operator about the caller)
mdsfax	(faxes)
mdsschd	(search data – for finding the correct pager from incomplete information supplied by the caller)
mdsschp	similar to mdsschd
mdsemail	(email)
mdscallp	call patch, where a Call Centre Operator connects a caller to an out-dialled line that they ring, like a telephone exchange operator

Table 18: Tables deleted from by del_line

6.11.5 FIXJOB

fixjob is an often used utility that stops and restarts processes when required, usually one process every couple of days or so – not bad considering the hundreds of processes running at any one time on each of the 7 <System Name> servers, probably 1000 processes in all, many running for a year or more continuously with-out problems.

The user passes the process name, the port that the process is registered against in the mdsports database table, or the process's label. Using information in the <System Name> start-up script, 'start.mds', fixjob stops and re-starts the background <System Name> process by raising the appropriate semaphore.

6.11.6 FIXFILE

fixfile inserts specified characters into a 'file' (a <System Name> queue) at the specified position and is used to forcefully fix a corrupted message queue. Usage: fixfile <file> <posn> <chars>

6.11.7 GETKEY

Receives a user's keystroke and prints it to a screen. Used to print out control characters.

6.11.8 GL3000LI

This utility displays screens and manages the adding, updating and deleting of GL3000 line records. "GL3000" is the name of the South Australian Government Radio Network's (SAGRN) system, to which <System Name> connects, while "line records", stored in the mdsline database table, hold all the information relevant to a particular phone number, such as the answer phrase, the subscriber type ('G' for SAGRN), pager number connected, maximum number of messages, etc.

There are other ways of adding a line record. This utility adds a customised screen and ensures some aspects of the record are set up properly.

6.11.9 GL3000PA

Like gl3000li (*q.v.*), gl3000pa displays screens and manages the adding, updating and deleting of South Australian government pager records in the <System Name> system.

6.11.10 HELPCOPMP

When using <System Name> one will note that there is a simple help system that can be summoned by pressing F1. The helpcomp utility prepares the help files, compiling them from their raw, text form, into the indexed and properly formatted pattern required by the <System Name> / curses help system.

6.11.11 MDSCAT

A tiny, simple program that pipes a file (or standard input) to a file (or standard output), with checking for certain control characters. When mdscat is used, the input or output is usually redirected to a comm port so that files can be manipulated in this way remotely.

6.11.12 MDSCHKQU

This utility checks and corrects possibly corrupt message delivery queues, by testing and optionally trying to fix the 'next free' (where should the next message be inserted in the queue) and next send 'which message should be sent next' message indexes in the queue header.

Useful especially after a process, or an entire <System Name> system, is re-booted, mdschkqu is also called automatically by the <System Name> start-up script: start.mds.

6.11.13 MDSCLEAR

Used to clean out and reset a message ('destination') queue. All the messages in the queue are instantly annihilated and the queue reset (as new).

6.11.14 MDSCLRPO

Every 15 minutes or so the mdsalarm process reads the mdsports table and checks that each process in there without an end time is still, in fact, running, otherwise it raises an alarm (the process has finished without logging and end time, thus it has finished abnormally, thus some problem, thus raise an alarm). There are times however when this behaviour is not appropriate, such as when the system is still being started up or one wants to suspend all alarms. In these scenarios the mdsclrpo program is used to fill the end-time field of each record in the mdsports table with '9's. This temporarily fools the mdsalarm process into thinking that all the processes have finished normally, by themselves – as they have lodged a (admittedly insane) process end time – thus all must be OK so no alarms.

6.11.15 MDSDATE

A <System Name> customised Unix 'date', program, which prints out the date in various formats.

6.11.16 MDSHIDSP

mdshidsp is a utility to write history data to standard out (i.e. the user's terminal). The user specifies some combination of pager number, line number, port, operator, 'from' date and time, and 'to' date and time; the appropriate history records being written to the screen or captured in a text file.

6.11.17 MDSLOGOUT

Use mdslogout to remotely log-out a user from a terminal port. This could be used in emergency situations, where an operator's processes have crashed, or as a standard Call Centre supervisor utility.

6.11.18 MDSLOOK

Taking either a PID or a program name the custom-defined signal SIGUSR1 is sent to the process. Ideally this will be not caught, but will tell the application to close some of its files and re-open them. If looking at the source code be aware that the kill( ) system function only sends signals, it's just that many of these signals will result in some sort of program termination.

6.11.19 MDSQDISP

mdsqdisp dumps to the user's terminal copies of messages sent to the specified queue. Useful for testing and tracking particular messages being sent through particular queues. If you want to monitor all the <System Name> queues at once use mdswq instead.

6.11.20 MDSQDUMP

Writes out the stats for the most prolific inputs to the queue that you specify. For example, mdsqdump may list a call centre operator with her/his character and word input speed and the phone number of a dial-up connection.

6.11.21 MDSQINC

A utility to allow the skipping (ignoring) of one or more messages in one or more queues. The user enters the queue names and the number of messages to be skipped, and these messages are not sent. Useful for testing and skipping corrupted messages in the queue.

6.11.22 MDSQMINC

As for mdsqinc, mdsqminc skips (effectively deletes) one or more messages from a maintenance queue so that these changes are not duplicated in other cities. Again useful for testing or if one is doing some special fix-up on one site that should not be duplicated to other <System Name> sites.

6.11.23 MDSRESET

This is a softer version of mdsclear (*q.v.*). mdsreset waits until all the messages in a queue have been sent before resetting the queue. If there are corrupted or stuck messages in the queue, mdsreset will fail to reset the queue.

6.11.24 MDSSCHLD

mdsschld duplicate the 'search data' from one customer to another, search data being rules about how to locate a call destination given only partial information. This program is only to be used by back-office <System Name> developers and support.

6.11.25 MDSSETGL

Similar to mdsetfu, mdssetgl either writes the global variables shared memory to stdout or rewrites the memory block as a copy of the **`$MDS_DATA_DIR / globvar.txt file`**

6.11.26 MDSSETPO

Run as a command-line utility mdssetpo set-ups and / or attaches to the ports shared memory, writes the ports shared memory to stdout and a text file, and clears out entries in the ports shared memory belonging to processes that are no longer running.

6.11.27 MDSSETPU

Like the other shared memory maintenance utilities mdssetpu manages public holiday shared memory. It writes it to stdout and a text file (if the '-l' parameter is used), establishes the public holiday shared memory from the data in the mdspubh database table ('-c' parameter), or rereads the public holiday information ('-u' parameter).

6.11.28 MDSSETRM

The reminders shared memory management utility. This program can either create the reminders shared memory ('-c' parameter) or update it ('-u'), it can not delete or list it. The reminders share memory is initialised by the data in the /shmids/reminders text file.

6.11.29 MDSSETSE

A utility to either

- write the security shared memory to stdout (so it can be captured in a file), or
- to clear and re-establish the security shared memory data from the mdssecur database table.

6.11.30 MDSSETTY

As described, the <System Name> tstty program transfers data flows from the device (/ dev / tty_xyz) on the <System Name> server to the terminal server into which the various devices (modems, terminals) are connected. The mdsetty program runs a copy of tstty for each device on the server that is (or is to be) connected to the terminal server in this way, as indicated by a flag in the mdsports table.

The earlier version of this program, mdssetocd, operated similarly but used the HP-Unix 'ocd' program instead of the <System Name> 'tstty'.

6.11.31 MDSWQ

mdswq displays a list of all the queues present on the <System Name> server, with totals of the messages passed through them that day (often in the tens of thousands), plus messages outstanding and being processes (seldom greater than 10). It is a good way to get a feel of the operation of a particular <System Name> system.

If one then wishes to monitor a particular queue in more detail the program mdsqdisp can be used to display the text of the messages passing through a particular queue.

6.11.32 MYMAIL

This is a simple wrapper for sendmail that simplifies the sending of various types of documents as email messages. The function that does the work, MIME_sendmail( ), is found in the mdsemrtn.c source file, which contains a sizeable collection of MIME message sending utilities.

Probably used in the testing of the <System Name> email handling (email SMS, Bulk SMS, fax …) systems.

6.11.33 PAGE

A fully featured utility for sending messages to a particular queue 'by hand', by passing the <System Name> / skymds01 systems. The simplest example is writing, on the command-line :

```
> page 98989 Hi John, this is my test message. Bye
```

and the message is delivered to the bulk queue and onto the pager number 98989. One can also

- enter the pager cap-code instead of the pager number,
- force the message to be sent to a particular queue, ending up as, for example, a fax,
- adjust one of a few other technical specifications.

An easy and simple way to both test and send message, page could easily be converted into a general use interface for people to send faxes, emails or to store messages, from a Unix command-line, other system, or their cellphones.

6.11.34 ST

By default st produces one great big list of all the <System Name> processes running on a server, being much more helpful in this regards than the HP-Unix ps command. A couple of parameters can be used to change some of the data presented.

6.11.35 TSTTY

tstty manages the data-flow between the <System Name> servers and the terminal servers to which most <System Name> devices – Call Centre Operator terminals, external users' modems, and other devices – connect. As an example, when someone connects to <System Name> via a modem using the PET protocol, tstty connects this terminal server port, via a device in the /dev/ directory of the <System Name> server, to a running skymds02 (PET protocol) server.

As an intermediary tstty provides additional functionality, such as the ability to monitor the traffic to, say, a Call Centre Operator's display screen.

6.11.36 ZAP

A utility to display a file in both hexadecimal and ASCII characters at the same time.

6.12 UTILITIES : CUSTOM INFORMIX UTILITIES

ISADD
Adds a new index into an Informix database table.

Usage: isadd <filename> <index-details>

6.12.1 ISBUILD, ISBUILD_VARLEN, ISBUILDU

Builds a new Informix database table with the tablename, (fixed) record-length and index details as supplied. Both isbuild_varlen and isbuildu are the variable length variants of isbuild, contracting database tables with variable length records.

6.12.2 ISCHECK

Check a last character of the records in an Informix database table is not a newline, which is easy to do by mistake.

6.12.3 ISCLEAR

Delete all the records out of an Informix database table but leaves the table's structure intact.

6.12.4 ISCLUSTR

If left to their own devices, Informix database table records are ordered effectively randomly, not usually in the order of any of their indexes. As access is faster if the records in the database actually are (instead of just appearing to be) in the order of their index, the iscluster( ) utility has been produced to reorder the physical records into the order stipulated by the selected index.

6.12.5 ISDELETE

Deletes an index from an Informix table.

6.12.6 ISINFO

Writes information about an Informix table's indexes to the screen.

For example :

```
$ isinfo mdsemqu

Record size       = 276
Number of keys    = 1
Index record size = 1023
Number of records = 1
U
D C 0-1/1-14
```

ISINFOC

Same as isinfo, but isinfoc prints out is data in such a format that it can be copied straight into an isbuild command used to contract a new Informix database table.

ISINFOV

A more detailed version of isinfo.

```
$ isinfov mdsemqu
Record size       = 276
Number of keys    = 1
Index record size = 1023
Number of records = 1
Dat file size     = 2048
Empty     records = 6
Slack bytes       = 1662
U
D C 0-1/1-14
```

6.12.7 ISLIST, ISLISTF

islist writes all records in a specified table to the terminal (stdout) while islistf does the same but in the order specified by the supplied index.

ISLOAD
Append the contents of a file to an Informix database table.
Usage: $isload infile table
e.g. $isload pager.txt mdspager

6.12.8 ISMOD

A utility usually used to view and traverse the labelled and formatted records of an Informix database, though one can also add, delete, or edit database table records with this utility.

7 Running <System Name> : databases, history, and queues

As its data-containers <System Name> uses both a database system and a large series of text files, the latter for high-speed, real-time access (i.e. messages) while the database, as discussed below, holds various less time-sensitive support information. The database system is Informix C-ISAM, designed to have its tables directly updated by C calls, thus by-passing any SQL or RDBMS overhead, while the message queues are implemented as text ("flat") files.

7.1 DATABASE

As mentioned, <System Name> uses an Informix C-ISAM database system. Systems of this type hold data in *.idx *.dat pairs, e.g. *filename*.dat and *filename*.idx, the *.dat file holding the fixed length records and indexes in the matching *.idx table[7]. While being a very exposed collection of data structures, an Informix C-ISAM system still implements locking, indexing and transactions as per usual RDBMS.

7.1.1 ACCESSING DATA

<System Name> stores both its database tables and its queues in the directory /var/mds/data/ The C-ISAM datatables are /var/mds/data/*.dat, the index tables /var/mds/data/*.idx. The queue text files are kept in the same location as /var/mds/data/mdsq??.txt. As one would expect, this directory is mounted as a separate RAID disk.

Under C-ISAM, one accesses database records by seeking through the file the correct distance (i.e. record-length * record-number) then copying the data into an array (or better, a record) the same size as the datastructure. C-ISAM provides its own functions for reading and writing to the datatables :

```
stchar(my_text, p_record_location, strlen(mytext))
thenumber = ldlong (record_Start_ptr)
```

for storing character, and
for loading (retrieving) long (interger) data.

or one can use the (unbuffered) C standard library functions `read( )`, `write( )` and `seek( )`. Unique and Primary keys for fast record location via indexes can also be used. To produce an object file or program that accesses a C-ISAM database #include isam.h and add -lisam to the 'cc' command-line and the set the INFORMIXDIR environment variable - '/usr' for <System Name> - before running the program.

7.1.2 MANIPULATING DATA

7.1.2.1 Manipulating Database tables

For each database table <System Name> maintains an associated C header (*.h) file, as shown in the following table. Thes0e header files, with a few exceptions, contain four standard data structures:

1. The index to the matching C-ISAM table, e.g. `#define KEY_EC_1 "U0-10,10-10,20-3"`
2. A struct representing the structure of the table, e.g. `typedef struct { ….. } ECREC;`
3. the size, in bytes, of a single table record e.g. `#define ECRECSIZE sizeof(ECREC)`
4. The filename of the database table e.g. `#define MDSEXTCO "mdsextco"`

which, together, supply all the detail one should need to access data from the detailed C-ISAM table.

Once one has identified the appropriate database table and header-file, the process to interact with a <System Name> table is as follows :

1. include the appropriate database table header (e.g. `#include "mdsecrec.h"`) and `isam.h` in your program,
2. declare variable of the database-record-struct type e.g. `ECREC ec;`
   and an integer to act as the file descriptor of the database files, e.g. `int lun_extco;`
3. in your program, use C-ISAM functions to open the database table,

```
if (!isinit(&lun_extco, MDSEXTCO, ISINPUT | ISMANULOCK, KEY_EC_2))
    return(FALSE);
```

isinit(…) has been written by <Company Name> programmers to wrap the C-ISAM open calls – which can still be used.

[7] the *idx table of a *.dat / *.idx pair also holds variable part(s) of variable length records. However, no variable length records are used in any part of the < System Name > system

4. if reading from the table, use C-ISAM functions (e.g. `isread(lun_extco, (char *)&ec, ISEQUAL);`) to find the appropriate record,
5. load the record into a buffer, using C-ISAM functions to match the buffer to your struct, or simply cast the DB record if you must
6. do what you will with the record, utilising functions such as stchar( ) and ldlong( ) described above
7. move on to the next record, or the program's next task.

7.1.2.2 Manipulating (adding and removing) queue records

First, some background information the reader should understand.

- In <System Name>, all queues are implemented as text files.
- each queue is associated with a port
- the ports are implemented as devices in /dev/*, e.g. `/dev/ttyq2`
- each queue has its own semaphore, used to indicate locked / available
- information about each queue is held in the mdsdestm C-ISAM table, including which port it operates on
- information about the ports used is held in the mdsports C-ISAM table, including each port's semaphore

So, the procedure for writing to a queue is as follows. What we have to do is find the port that a queue is written to, then find and test the semaphore attached to that port to ensure that the port is actually free to be written to. Once we are sure that the port is available, we open the queue file, lock it for our use and start writing to it. We unlock it once we are done.

So :

1. Decide which queue you want to write to
2. Open the mdsdestm C-ISAM table, and find the record for the queue you have chosen.
3. form the queue record, find what port it writes to
4. open the mdsports C-ISAM table, the "ports master table", and locate the port from (3)
5. find the semaphore for this port
6. open the file
7. get the semaphore and indicate that the queue is being written to –increment or lock the semaphore
8. write to the queue

7.1.3 DATABASE AND HEADER FILE REFERENCE TABLE

DB Table	Description	Header File *.h
hiccyymmdd	the 'logical' history	mdshirec.h
hsyymmdd	History secondary storage	mdshsrec.h
mdsacrec	Alt call details master	mdscdrec.h
mdsadvis	Advisory	mdsadrec.h
mdsalarm	Mdsalarm	mdsalec.h
mdsaparty	Mdsaparty	mdsaprec.h
mdsaudit	Audit trail	mdsaurec.h
mdsbull	Operator bulletin	mdsbtrec.h
mdscallp	Call patching master	mdscprec.h
mdscdrec	Call details master	mdscdrec.h
mdschkpnt	Generic checkpoint	mdsckrec.h
mdsclien	Client master	mdsclrec.h
mdsclsts	Client stats master	mdscsrec.h
mdscmdet	Company details	mdscmrec.h
mdsconta	Contacts master	mdscorec.h
mdsctext	Escalation large canned text	mdsctrec.h
mdscustp	Customer program	mdscurec.h
mdsdelay	Delay monitoring	mdsdlrec.h
mdsdestm	Destinations master	mdsderec.h
mdsdsrec	Delivery status master	mdsdsrec.h
mdsemail	Email master	mdsemrec.h
mdserror	Mds generic error	mdserrec.h
mdsescal	Escalation master	mdsesrec.h
mdsestrn	Escalation transaction master	mdsetrec.h
mdsevent.txt	Event number master	mdsevrec.h
mdsextco	External contact master	mdsecrec.h

Table 19 : Database tables

DB Table	Description	*.h
mdsfacil	Facilities	mdsfarec.h
mdsfax	Fax master	mdsfxrec.h
mdsfmtxt	Forms text	mdsfmrec.h
mdsforms	Forms	mdsfmrec.h
mdsfunct	Function codes	mdsfurec.h
mdsgl02	gl3000 vbilling protocol	mdsg2rec.h
mdsglerr	Gl3000 error	mdsgerec.h
mdsinpak	Info pack .	mdsiprec.h
mdsline	Line	mdslirec.h
mdsln2	<System Name>2 switch position	mdsln2rec.h
mdsmccnm	Mobile call completion name	mdsmcrec.h
mdsnetcn	Network connection	mdsncrec.h
mdsnrost	New (<System Name> v) roster master	mdsnrrec.h
mdsopsts	Operator statistics	mdsosrec.h
mdspager	Pager master	mdsparec.h
mdspager	Sms delivery status	mdsshrec.h
mdspassw	Password username	mdspsrec.h
mdspasts	Pager stats master	mdsstrec.h
mdsphone	Phone master	mdsphrec.h
mdsports	Ports security	mdsporec.h
mdspubh	Public holiday	mdspurec.h
mdsrem	Reminders queue	mdsrmrec.h
mdsremp	Reminders parameter	mdsrprec.h
mdsschd	Search data	mdssdrec.h
mdsschp	Search parameter	mdssprec.h
mdsscode	Service code	mdsscrec.h
mdssecur	Security codes	mdsserec.h
mdssequ	Message sequence number	mdssqrec.h
mdssmsta	Sms stats	mdsssrec.h
mdssrf	Software request form	mdssfrec.h
mdssubrb	Suburb post-code master	mdssurec.h
mdstext	Text header	mdstxrec.h
mdstimev	Timed events	mdsterec.h
mdstmesg	Temporary message master	mdstmrec.h
mdsuniqn	Unique number	mdsunrec.h
snet.list	Smartnet description	mdssm.h

Table 20 : Database tables(continued)

7.1.4 HEADER FILES ASSOCIATED (ONLY) INDIRECTLY WITH DATABASE TABLES

*.h	Description	
mdstrrec.h	Inter processor transaction queue	mdsqa1.txt
mdsqarec.h	Bulk queue	mdsqqb.txt
mdsqurec.h	Destination queues	mdsqxx.txt
mdsxprec.h	Pabx interface communications layout	XACOM COMMS
mdsxzrec.h	Xacom encoder transmission buffer.	XACOM COMMS
mdshrrec.h	layout for thetransmission records received and sent to the new history server..	
mdsb1rec.h	layout for the transmission record in SKYMDS75.Instacomm wide area protocol format..	
mdslcrec.h	layout for the transmission record in SKYMDS07..Line Card Alarm Unit message transmission record.	
mdsmprec.h	Message Entry and Retrieval Protocol (MERP)	
mdsmrrec.h	MERP Message Entry and Retrieval Protocol (MERP). - definition of the message transmission buffer	
mdsmtrec.h	Message transaction protocol	
mdsperec.h`	layout for the transmission record in SKYMDS02.. : PET message transmission record.	
mdsqdrec.h	Detailed data for bulk and dest queue file	
mdswarec.h	layout for the transmission records received and sent to the WANG in SKYMDS18..	

Table 21 : Headers and data-structures related to databases

7.2 THE HISTORY SERVER AND CDR

While <System Name> is running, the history server and the CDR are recording all transaction details. The history server records the details of every message, to a level such that the message could be resent if necessary. CDR is more billing focused, thus recording message length, subscriber type, priority, phone number and the like. There is only one History Server and it lives in Melbourne. As a continuation of past behaviours the CDR data is gathered in Adelaide before being passed back to the Oracle LMS system in Melbourne for billing purposes. Note that this – a single billing system – is unlike most of <System Name> in that it is not dutifully replicated in every city across Australia.

Historical information is, to some extent, duplicated on the local and the history server. The history data, both primary and secondary (see below), are held as per the table :

	local <System Name> server	**History Server by City (e.g. …hist / M /)**	**History Server Combined (…/hist)**
next 50 days	-	-	empty
next 7 days	empty	-	empty
dated today[8]	full	-	empty
to 7 days ago	full	-	empty
7 to 14 days ago	full	full	full
14 to 43 days ago	full	compressed	full
43 to 365 days ago	-	compressed	full

Table 22 : History data availability

7.2.1 HISTORY

7.2.1.1 Machine – History Server

The History Server – lnkh ('h' for history), and its directories /var/mds/data/hist/? (A, B, M and O, P, and S and X for Adelaide, Brisbane, Melbourne, Perth and Sydney respectively) are the home of the <System Name> history files. Local history files live on each of the other <System Name> machines, in their own /var/mds/data/hist/ directory – but not in parallel A,B,M,O,P,S,X directories.

7.2.1.2 History Data

<System Name> history exists in two supplementary forms : Primary and Secondary.

- Primary history, a daily C-ISAM database table (*.dat and *.idx pair) , which holds all the information *about* the message, except the message id and the message text itself. That information is stored, separately, in the secondary history file. An example of a primary history file is hi<date removed>0730.dat and hi<date removed>0730.idx
- Secondary History : a textfile (*.txt) accompanying the primary history database table, and containing the core message : its text and its message ID. Secondary history files are named as hs<date removed>0730.txt

As the examples show, history files are named :
Primary: hiccyymmdd.dat and hiccyymmdd.idx, e.g. hi<date removed>0730.dat and hi<date removed>0730.idx. NB: "h**i**" for primary
Secondary: hsccyymmdd.txt, e.g. hs<date removed>0730.txt. NB: "h**s**" for secondary

On each <System Name> server, local history is stored in /var/mds/data/hist/. Here one will find the previous 40 days of primary and secondary history files, plus the next 7 days of empty history files waiting to be filled. Each <System Name> system must have a week of empty history files to ensure that no records are ever lost.

On the history server, lnkh, is both the combined daily history files from each Australian site (in /var/mds/data/hist/), and compressed copies of each site's history files one will find a large collection - about 1 year's worth.

[8] Yes, today's history. Each History file is made during the small hours of the morning – containing the previous 24 hour's data. So the date on the most recent history file is, by the time you come to work, today's date.

The hi*YYYYmmdd*.dat.Z and hi*YYYYmmdd*.idx.Z are the (compressed) primary history file, the secondary history file is the matching hs*YYYYmmdd*.dat.Z. All files older than 7 days are compressed, while those greater than a year are deleted - though still retrievable from backup if you are really interested.

7.2.1.3 Maintenance of the History Files

Three scripts maintain the history files : start.68, start.hisrcp, and start.hismer. All of these run as overnight cron jobs.

1. Creating History Files

First, the Cron Line :

```
5  2  *  *  *     /opt/mds/bin/start.68          #New History File Creator
```

instructs that at 2:05 am every morning, cron on each machine calls start.68, which, as the name suggests, starts the program skymds68. skymds68 creates new, empty history files, ensuring that there are always 7 days of empty history files available.

2. Storing History Files

Next cron (on lnkh) runs hiscrp to copy the previous day's history files from their respective locations on machines around Australia to itself :

```
0  3  *  *  *     /opt/mds/bin/start.hisrcp      #Copy history files
```

3. Merging History Files

Finally, at 8am, start.hismer calls mdshismer to merge various collected history files together

```
0  8  *  *  *     /opt/mds/bin/start.hismer      #Merge all history file
```

During the day the empty history files are filled by :

1. skymds20 - The Bulk Queue to Delivery Queue process .
   The code to write skymds20 history details is in mdsqall.c. These functions write the secondary history record file (the message text and pager, SMS id, whatever) but only create (not write to) the more detailed primary history record. Examples are :

setup_and_write_hist_primary( )	Set up the primary history file record and call hist_write_prim( ) to write it
write_contact_parent_hsec_rec( )	writes the file offset of the matching secondary history record (in the secondary history text file) onto the primary history record.
hist_write_sec( )	write the secondary history record

2. skymds03 - The distribution queue to delivery system daemon .
   For all messages except TMS, ADV (advisories), Batch Fax, Batch Email and Remote data a history record is written. "I got your message" type replies from a (remote) SMS centre also stimulate the writing of a history record. `update_hist_file( )` and `update_hist_by_remote_rec( )` are the functions in skymds03 used to write local and remote SMS history records respectively.

3. skymds67a – data back from SMSC .
   skymds67a processes the data received back from external SMS centres. However, the only information it writes to the history files is a note of which final carrier (e.g. Telstra, Vodaphone, Optus) transported the SMS message. This also operates as an "I got your message" signal from the remote SMS centre.

7.2.1.4 Primary History File (e.g. hi<date removed>0304.dat)

This is all the information about the call, and is described in in complete detail in mdshirec.h. As an example of a Primary History file record :

```
P000033599102145100827351 99A/dev/adb21       C0082567f  M MORRIS b  515987503O0060000308835000309224
```

The call information held in this Primary History file record is as follows :

Field Value	Meaning
P	this is a pager call
335991	the pager number
02:14.51	time message received
8273-5199	originating telephone number
A	code of the originating machine
/dev/adb21	the port that the message was received on.
C	The contact clearing system (skymds27) wrote this. The codes are in mdshirec.h.
82567 (f)	The alternative pager or SMS number
M MORRIS	The operator that took the call
b	a message sequence number
515987503	The event (escalation) related to this message
'O' ("oh", not zero)	Operator initiated this message
0060	the length of the transaction
000308835	Index to first related secondary history record
000309224	Index to the last related secondary history record

Table 23 : Meaning of Primary History file fields

7.2.1.5 The Secondary History File (e.g. hi<date removed>0304.txt)

The secondary history file contains the core data of the messages themselves, detailed in mdshsrec.h.

Examples of secondary history records are :

```
m 50010735827 WARWICK PLUMSTED, , , , PLS PH YOUR WIFE AT HOME
c 82010732034 SE<date removed>0708073011   DEPOT     JAN M    512036300
d 0000null_ptr 07 s2 <date removed>0708073015 y1 <date removed>0708073013 a2 <date
removed>0708073014 p2 <date removed>070807 … …
r 60010732944<date removed>07080753320099631665 ALLISON A /dev/add02
t 65010733415 Escalation Exhausted (Inform Supervisor) Line 99631665  Event 512036302 … …
```

The records above are reasonably obvious once one can understand each records' contents from the single letter code at the start of each. Again, more detail is to be found in mdshsrec.h. As a few brief comments on the examples above :

	Message Code	Description of Message
1	m – message	A pager message text
2	c – contact	'Contact' messages, handled by skymds27, need (and receive) some sort of advanced processing..
3	d – destinations	Messages can be sent to multiple destinations ('07')> This record indicates followed by 7 pairs of - destination queue (see the queue section of this manual), and - yyyymmddHHmmss date-time stamp
4	r – retrieval	A message retrieved from history storage
5	t–contact message	What was actually sent as the result of a "Contact" special handling. In this case it is a message to a Call Centre operator to contact their supervisor - to ensure that the mail gets through.

Table 24 : Meaning of Secondary History file fields

7.2.2 CDR - CALL DETAIL RECORD

CDRs are the raw billing-related data within <System Name>. Call Detail Records are written by skymds01 - the Call Centre interface program - and often by skymds20 - the program that extracts messages from the Bulk Queue for allocation to distribution queues. skymds01 writes CDRs for all types of calls, including those that are incomplete, while skymds20 adds CDRs as part of its queue processing.

On each local machine, all CDR data is written into /var/mds/data/cdr, to text files named CDRMccyymmdd (e.g. CDRM<date removed>0720 - M and O for Melbourne, S for Sydney …). At the end of each day these are copied to /var/mds/data/cdr in Adelaide (server lnka). From here they are combined (into CDRALLyyyymmdd) from whence a utility transfers this (large) daily collection of billing information to the (Melbourne) LMS server for uploading to an Oracle database for salubrious financial processing. About 250,000 to 300,000 CDR billing records are transferred each day, about 80-90,000 from Melbourne.

The CDR file format is defined in mdscdrec.h, with mdsacrec.h defining an extended version that wraps and extends a standard CDR record. mdscdr.h is a long list of definitions, constants and settings for the CDR process.

7.2.2.1 CDR File Structure

As noted, the /var/mds/data/cdr/CDRMyyyymmdd is a local CDR billing file, whose structure is held in mdscdrec.h. Following is the detail of one CDR line, followed by a full "CDR Matrix", stipulating all possible CDR line meanings.

An example of a CDR line :

<date removed>0720001700|00|00|300984|O|2|05|0003119999|3||L|0000238031||m5|0000079|5|1|35123900|N|L|92090000|/dev/mla31|L|L||~

Can be interpreted as :

Field Value	Meaning
<date removed>0720001700	Date, time
00	Transaction Type
00	Sub Type
300984	Operator ID
0	What machine (O is Melbourne #2)
2	External
05	"CPU" - external computer
0003119999	In dial Number
3	In dial Type. 3 means remote access
[N/A]	2 indicates personalised Service
L	Subscriber type : 'L' means <Company Name>
0000238031	Pager number
[N/A]	Contact (special handling) ID
m5	RF frequency or email address
0000079	message length
5	Which Frequency. 5 means 149.6125 MHz
1	Transmission rate. 1 means 1200 baud
35123900	Pager Client Number
N	Priority? (No)
L	Subscriber Type (<Company Name>)
92090000	Where the call was made from
/dev/mla31	The Port
L	Subscriber Type
L	Subscriber sub-type

Table 25 : Meaning of CDR file fields

See over for a more detailed CDR Matrix.

7.2.2.2 CDR File Structure Matrix

1:Date	**6:Source**	**11:Indial Subscriber Type**	**16:Frequency**	**21:Indial Client Number**
2:Transaction Type	**7:Sub Source**	**12:Pager/Mobile No.**	**17:Baud Rate**	**22:Port ID**
3:Trans Sub Type	**8:Indial Number**	**13:Cont. Id/File /Batch Num**	**18:Pager Client Number**	**23:Indial Subscriber Type**
4:Access Code	**9:Indial Type**	**14:Destination Delivery**	**19:Pager Priority**	**24:Pager Subscr. Sub Type**
5:Machine Id	**10:Indial Service Type**	**15:Volume**	**20:Pager Subscrib Type**	**25:Source Address**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Y	**00 Paging**	00 Alpha	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	RF dest	Length	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y		01 Numeric	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	RF dest	Length	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y		02 Tone	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	RF dest	Length	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	**01 Incalls**	01 Inactive	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	Y	Y	Y	N	N
Y		02 Invalid	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N
Y		03 Call Duration	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	No of Seconds	N	N	N	N	N	Y	Y	Y	N	N
Y	**02 Faxing**	04 On Line Fax Pager	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Fax	Length	N	N	Y	Y	Y	Y	Y	Y	Y	N
Y		05 On Line Fax In dial	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Fax	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		06 Batch Fax Pager	N	Y	N	N	N	N	N	N	Y	N	Fax	Num Pages	N	N	Y	Y	Y	N	N	N	Y	N
Y		07 Batch Fax In dial	N	Y	N	N	Y	Y	Y	Y	N	N	Fax	Num Pages	N	N	N	N	N	Y	N	Y	N	N
Y		08 Customised Fax Reports	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Fax	Num Pages	N	N	N	N	N	Y	Y	Y	N	N
Y	**03 Email**	04 On Line Email Pager	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Email	Length	N	N	Y	Y	Y	Y	Y	Y	Y	Y
Y		05 On Line Email In dial	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Email	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		06 Batch Email Pager	N	Y	N	N	N	N	N	N	Y	N	Email	Num Pages	N	N	Y	Y	Y	N	N	N	Y	Y
Y		07 Batch Email In dial	N	Y	N	N	Y	Y	Y	Y	N	N	Email	Num Pages	N	N	N	N	N	Y	N	Y	N	N
Y		08 Batch Email Text	N	Y	N	N	Y	Y	Y	Y	N	N	Email	Size of File	N	N	N	N	N	Y	N	Y	N	N
Y	**04 Message To TMS**	00 TMS Message	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y	**05 Message To Text**	00 TEXT Message	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y	**06 Manual Reminder**	00 Manual Reminder	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		01 Booked Page Add	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	Y	N	N
Y		02 Booked Page Change	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	Y	N	N
Y		03 Booked Page Delete	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	Y	N	N
Y	**07 Contact Processed**	00 Advisory	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		01 Call Patch	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		02 Email	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		03 Escalation	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		04 Fax	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		05 Form	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		06 Group	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		07 Manual Reminder	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		08 Pager Contact	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		09 Reminder	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N
Y		10 Roster	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Length	N	N	N	N	N	Y	Y	Y	N	N

Table 26 : More detailed CDR Matrix

7.3 QUEUES

<System Name> is a messaging system and while, like most other systems, the majority of her information is held in a database system, all the messages themselves traverse <System Name> from queue to queue. The central repository of all <System Name> messages is the "Bulk Queue", from where messages are passed to one of a large number of "destination queues". The majority of messages in Sydney (say) will travel to local (Sydney) destination queues, though, after being passed by skymds23 / skymds24 to another city they may end up in any queue, of any type around Australia or be held in a *local* copy of some other city's queue. Each site individually maintains a copy of every Australian queue.

7.3.1 THE BULK QUEUE

All messages in <System Name> pass through the bulk queue. From this queue :

- each message is distributed to the appropriate "distribution" queue for delivery
- Billing data is gathered as CDR ("Call Detail Record") history
- The messages are individually recorded for later retrieval both locally and on the History Server.

The format of a Bulk Queue record is detailed in the header file mdsqdrec.h, while the Bulk Queue itself is instantiated as /var/mds/data/mdsqqb.txt . Many programs insert messages into the Bulk queue, using the functionality coded in mdsqmain.c, the queue format being specified in the header mdsqmain.h. The daemon sykmds20 extracts messages from the Bulk Queue for allocation to local destination queues.

7.3.2 THE DESTINATION QUEUES

7.3.2.1 Queue Naming

Examples : mdsq**m1**.txt – melbourne pager queue
mdsq**em**.txt – email queue
mdsq**q3**.txt – a Bulk SMS queue

The following describes the complete queue-naming logic used by <System Name> - from which any queue should be able to be classified. Each queue is given a two letter name, and as the above examples show, this two letter name is concatenated to "mdsq" to give the actual queue name "mdsqXX.txt", which can be found in /var/mds/data/*.

The system for allocating names is as follows :

Queue Attribute	How the Queue is named
City Names :	Melbourne : m and o Adelaide : a VIC, NSW, WA rural : g VIC, NSW, QLD, rural : w and x All Australia : y and z Perth : p Canberra : c Sydney : s and x NSW and QLD Coast : v Northern Territories : n Tasmania : t (t + letter = telstra) SAGRN : g Brisbane : b History : h
Frequencies :	There are 6 frequencies, numbered, well, 1,2,3,4,5 and … 6 a6 is a special frequency owned by the South Australian Government (SAGRN)
Pager Queues :	The city letter followed by a frequency, as above m1 or s4, a5 or c3. Some city letters may be followed by further letters instead of numbers : Dual Frequency or Telstra.
Dual Freq Queues : (Pager)	Canberra, Perth, Adelaide and Brisbane have message queues that handle pages of two or three different queues. These queue names combine a city letter with some other letter - of no special importance - to identify the queues. The current examples are : Adelaide : al : Frequency 4 and 5 Canberra : cb : Frequency 4 and 5 Perth : pb : Frequency 4 and 5 Brisbane : bn : Frequency 1 and 5 Perth : pa : Frequency 2 and 5

Table 27 : Queue naming rules

Queue Attribute	How the Queue is named	
Pager via satellite	"eq" - equatorial satellite	
	Many queues are directed to the 'eq' queue :	
	n2, n4 : Northern Territory Queues	t2, t4 : Tasmanian Queues
	ta,tb,tc : Telstra South Australia	tf, tg,th : Telstra Rural Victoria
	v : NSW and QLD Coast	w and x : VIC, NSW and QLD rural
	All of these, thus, have the single mdsqeq.txt queue as their physical destination, the two letter representation (e.g. v4 (Queensland Central Coast) and t2 (Tasmania)) being maintained for internal, conceptual use. Further, at a later time, one or more of these satellite transmission may come to be delivered by a less celestial procedure, so we want to keep these representations. Think of the messages that are transmitted by satellite as having two queues : one descriptive (and non-existant), the other – the equatorial queue – being the true distribution queue for the message	
<Company Name> SMS Queues	ka : <Company Name> Pager	ja : New Zealand
Carrier SMS queues	h2 : Optus	h3 : Vodaphone
Bulk SMS Queues	q1 to q4 : SMS "priority" queues.	bs : Bulk SMS queue
email	em, e2	
special external queues	mi : forwarding to Mitsubishi	m2 : forwarding to New Zealand
	vo : Vodaphone SMPP	op : Optus SMPP
	a6 : SAGREN – South Australian Government.	
	sa, sb : Sydney Priority Pager	ma : Melbourne Priority Pager
special internal queues	co : "contact" – special processing	du : Dummy, null queue
	rm : Reminders for the Call Centre	qm : maintenance queue
Unused	i : A large number of 'i' queues are no longer used	

Table 28 : Queue naming rules (continued)

7.3.3 QUEUE REFERENCE

The Queue Reference listing on the next few pages details all the queues you should ever find in a <System Name> system. The tables are constructed as follows :

Queue	the internal name for the queue
to Queue	Where the queue is re-directed to
Queue File	The physical file holding the queue. Most of these queues will be present at most sites.
Name	the four letter human-readable name of the queue
freq	what frequency band (of 5) is the broadcast made on, "eq" meaning "equatorial", indicating that the transmission will be via satellite. Any particular pager will receive on only one of the six frequencies, the frequency (as a 'channel') being matched to the pager number in the <System Name> system.
Location/Dest	what area will the pager message be broadcast in. A page processed in Sydney may be for Perth.

< two pages of data removed >

7.3.3.1 Dual Frequency and the "Equatorial" (Satellite) Pager Queues

For economy in areas of lower pager traffic, encoders (and transmitters) can be set up to process pager messages for transmission on multiple frequencies. In this case, a single queue for a pager message encoder will contain messages to be broadcast on multiple frequencies - a dual frequency pager queue.

queue	Queue File	Name	Location/Dest	freq
al	mdsqal.txt	ADEL	Adelaide	4 and 5
bn	mdsqbn.txt	BNES	Brisbane	1 and 5
cb	mdsqcb.txt	ACT	A.C.T.	1 and 2
pa	mdsqpa.txt	PERA	Perth	2 and 5
pb	mdsqpb.txt	PERB	Perth	4 and 5
eq	mdsqeq.txt	EQUA	Equatorial satellite	

Table 29 : Dual Frequency Queues

7.3.3.2 SMS Queues

Queue	Queue File	Name	Location/Dest
h2	mdsqh2.txt	SMS2	Optus SurePage
h3	mdsqh3.txt	SMS3	Vodaphone CallScreen
ja	mdsqja.txt	SNZ	NZ SMS
ka	mdsqka.txt	SAU	<Company Name> SMS
op	mdsqop.txt	OPT1	Optus SMPP
vo	mdsqvo.txt	VODA	Vodafone SMPP

Table 30 : SMS queues

7.3.3.3 Bulk SMS

Queue	Queue File	Name	Location/Dest
bs	mdsqbs.txt	BULKS	Bulk SMS
q1	mdsqq1.txt	PQU1	SMS priority queue 1
q2	mdsqq2.txt	PQU2	SMS priority queue 2
q3	mdsqq3.txt	PQU3	SMS priority queue 3
q4	mdsqq4.txt	PQU4	SMS priority queue 4

Table 31 : Bulk SMS queues

7.3.3.4 Other Queues

A few queues don't fit nicely into the main groupings above. These are :

- The Bulk Queue – all messages pass through the bulk queue
- The Maintenance Queue – data for the synchronisation of the <System Name> systems.
- Internal Queues, that is, internal buffers inside the <System Name> system
- Other Queues – Fax, email, special Customer
- SAGRN. Queue a6 is for the South Australian Government Radio Network.

Queue	Queue File	Name	Location/Dest	
qb	mdsqqb.txt	BULKQ	Bulk queue	Bulk Queue
qm	mdsqqm.txt	MNTQ	Maintenance queue	Maintenance
co	mdsqco.txt	CONT	Contact processing	Internal
du	mdsqdu.txt	DUMY	Nowhere, nothing, zilch	Internal
e2	mdsqe2.txt	EM2	Email	Other
em	mdsqem.txt	EML	Email	Other
f2	mdsqf2.txt	FA2	Fax	Other
fx	mdsqfx.txt	FAX	Fax	Other
mi	mdsqmi.txt	MIT1	Mitsubishi	Other
rm	mdsqrm.txt	REM	Tone/voice reminders	Internal
a6	mdsqa6.txt	ADE6	Adelaide	Sth Aust Govt

Table 32 : Other <System Name> queues

8 Running <System Name> : external interfaces

8.1 PROTOCOLS

8.1.1 OVERVIEW

<System Name> is a powerful messaging and communication system, which supports many data communication protocols. The following is a list of these protocols:

- Pager entry terminal (PET) protocol
- Message transaction protocol (MTP)
- Message entry and retrieval protocol (MERP)
- Short message peer to peer (SMPP) protocol
- Telocator network paging protocol (TNPP)
- XACOM protocol

This section outlines each of the protocols listed above.

8.1.2 PAGER ENTRY TERMINAL PROTOCOL

Pager entry terminal protocol (PET) is a communication protocol that enables the remote customer computers (or devices) to directly connect into <System Name> paging network, which eliminates the need for message delivery via a human operator in call centre. With the PET connection, all ASCII characters in the defined format can be transmitted between the remote computers and the network of paging hosts.

The original PET specification provides many features, some of which are optional, usually at the character and field level. Manual remote entry devices and multiple block message transactions are not supported from the original PET specification. Most responses from the paging host allow for optional message.

The <System Name> system supports the PET protocol both on public dialup ports (multiple modems on a rotary line) and private dedicated leased line port. These services are both currently available in most capital cities in Australia. The protocol is exactly the same for both public dialup port and leased line port, however with dedicated ports, the time-out periods are more relaxed.

Direct access services can be connected by any means that provides a direct, error free data transfer from the customer premises to <System Name> system. Typical service types include Leased Lines, DMS, and DDN data services.

- Leased Lines are primarily a “pair of wires” between the customer premises, and <System Name> system, in which no modems or other hardware is provided. Customers must provide modems suitable for direct connection, data transmission, and the grade of service for fault restoration is lower than for DMS services.
- DMS services are simple direct data service with premium service for faults and problems. NTU modems are supplied and maintained by Telstra, and the service is available in a variety of speeds and connection types. These services are only available in metropolitan areas, but offer a very reliable error free data transmission medium.
- DDN services are a higher premium data service which allow for direct, end to end connections, at a variety of speeds and connection specifications. DDN services are substantially dearer than DMS, and usually only used by users outside the metropolitan areas.

Use of the PET protocol for remote access to the <System Name> system involves the following steps:

- Have the modem attached to your computer, dial the appropriate phone number for the public or dedicated PET port. Check the <Company Name> for the telephone number.

- Logon to the <System Name> system with your user ID and password. The system will prompt you with error message and ask you try again if the logging fails. The maximum number of attempts is three.
- Send messages to and receive messages from <System Name> system directly. These operations of receiving and sending can be repeated as many times as you like.
- Log off the system once the message delivery or receiving operations completed.

8.1.3 MESSAGE TRANSACTION PROTOCOL

Message transaction protocol (MTP) is a data communication protocol developed by <Company Name> Telecommunications. The protocol provides a generic interface between multiple hosts and the <System Name> system. It is a message transaction based protocol that allows for both customer, and application specific transaction, to be easily defined and implemented.

The MTP protocol is ASCII character oriented, which is transmitted via a standard RS-232 asynchronous port or via a TCP/IP connection. The protocol is based on a client/server architecture, where one host requests a transaction, and the other host processes the request and replies back with the required details. Inactivity polling is also used for the purpose of alarming if the connection between hosts, or one of the hosts is down.

The protocol supports variable length packets, comprising of a packet header, and a message transaction data block. The header is common in all data formats for both requests and replies. The header also contains a serial number used to identify repeated or missed packets.

The MTP protocol supports the following transactions:

- Polling transaction
- Cap page transaction
- Line transaction
- Test transaction
- Error transaction

All supported transactions types are optional, except for the Polling transaction that is required in all implementations. The polling transaction is used to validate the communication <Company Name> between the client and server, and to generate a system alarm if required.

8.1.4 MESSAGE ENTRY AND RETRIEVAL PROTOCOL

Message entry and retrieval protocol (MERP) is a communication protocol that enables the remote customer computers to directly connect into <System Name> paging network. With the MERP connection, all ASCII characters in the defined format can be transmitted between the MERP user pagers and networked hosts. The protocol supports many transaction types, and is more powerful and flexible than previously defined protocols.

The MERP protocol is based on a client/server architecture, where client host requests a transaction, and the server host processes the request and replies back with the required details. Inactivity polling is also used for the purpose of alarming if the connection between the client and server or one of the hosts is down. The protocol supports variable length packets, comprising of a packet header, and a message transaction data block. The header is common in all data formats for both requests and replies. The header also contains a serial number used to identify repeated or missed packets.

The original MERP specification provides features to send messages to a pager by using either its pager id or cap code, to retrieve previously sent messages, and to perform a watchdog or heartbeat transaction to test the status of the connection. A user code, sequence numbers and a cyclic redundancy check (CRC) are used to ensure that transactions are valid and correct. A reset sequence number transaction is also included should the sequence number between the MERP user and the <System Name> system get out of sync. At the current date, the message retrieval transaction is not supported. However, the format and usage of the request and reply transactions are already defined in readiness of this feature being implemented in the future.

The <System Name> system supports the MERP protocol on private dedicated leased line ports only. There is no public dialup ports. MERP service is currently available in most capital cities in Australia. Like the PET protocol, direct access services can also be provided with Leased Lines, DMS, and DDN data to act as message inputs into the <System Name> system or as message destinations.

Like the PET users, direct access services can be connected by any means that includes Leased Lines, DMS, and DDN data services. For completeness, we list these data services.

- Leased Lines are primarily a "pair of wires" between the customer premises, and <System Name> system, in which no modems or other hardware is provided. Customers must provide modems suitable for direct connection, data transmission, and the grade of service for fault restoration is lower than for DMS services.
- DMS services are simple direct data service with premium service for faults and problems. NTU modems are supplied and maintained by Telstra, and the service is available in a variety of speeds and connection types. These services are only available in metropolitan areas, but offer a very reliable error free data transmission medium.
- DDN services are a higher premium data service which allow for direct, end to end connections, at a variety of speeds and connection specifications. DDN services are substantially dearer than DMS, and usually only used by users outside the metropolitan areas.

Use of the MERP protocol for remote access to the <System Name> system involves the following steps:

- Have the modem attached to your computer, dial the appropriate phone number for dedicated MERP port. Check the <Company Name> for the telephone number.
- Logon to the <System Name> system with your user ID and password. The system will prompt you with error message and ask you try again if the logging fails. The maximum number of attempts is three.
- Send messages to and receive messages from <System Name> system directly. These operations of receiving and sending can be repeated as many times as you like.
- Log off the system once the message delivery or receiving operations completed.

8.1.5 SHORT MESSAGE PEER TO PEER PROTOCOL

Short message peer to peer protocol (SMPP) is a data communication protocol that defines the data format and transmission events required for interface to the short message service centre (SMSC) for transactions relating to short messages in the GSM network. The protocol is a generic protocol that can be used for data communications between any short message entity (SME) such as a <Company Name> message hosts or voice mail system.

SMPP protocol is a complicated protocol that supports 19 different transaction types. Transactions are of a variable length and the fields that a transaction contains can be either of a fixed or variable length. Each transaction can have between 4 and 22 fields.

Using the SMPP protocol, an SMS application system called the external short message entity (ESME) may initiate am application layer connection with SMSC over a TCP/IP or X.25 network connection and may then send short messages and receive short messages to and from the SMSC respectively. The ESME may also query, cancel, or replace short messages.

SMPP protocol supports digital cellular network technologies including GSM, IS-95, ANSI-136, and iDEN, and it provides a fully featured set of two-way messaging functions including the following:

- Transmit messages from an ESME to single or multiple destinations via the SMSC
- An ESMC may receive messages via the SMSC from other SME's (e.g., mobile stations)
- Query the status of a short message stored on the SMSC
- Cancel or replace a short message stored on the SMSC
- Send a registered short message
- Schedule the message delivery date and time
- Select the message mode, i.e. datagram or store and forward
- Set the delivery priority of the short message
- Define the data coding type of the short message
- Set short message validity period
- Associate a service type with a message, e.g. voice mail notification

8.1.6 TELOCATOR NETWORK PAGING PROTOCOL

Telocator network paging protocol (TNPP) is a data communication protocol that is used for communications between paging terminals or other types of equipment required to implement a paging system network. The protocol is ASCII character oriented that is transmitted via a standard RS-232 port.

The protocol supports variable length packets and group of different page and data blocks within the same data packet. The protocol uses three types of error recovery methods, two for full duplex applications and one for broadcast/simplex application.

A common header is used in all of the data formats to indicate the source and destination of each data packet. The header also contains a serial number that is used for identifying packet repeats and an inertia value that represents the number of nodes the packet is allowed to pass through. The header length is variable and is delimited from the reminder of the packet with the start of text flag.

After the common header, an optional header extension can be used to extend the source and destination address fields. If a device does not use this feature, the header extension data between the first 12 bytes of the header and start of text flag should be ignored. After the start of text flag the data block is sent. Multiple data blocks can be transmitted within the packet by terminating each block with an end of text block flag and terminating the last data block with an end of text flag followed by the block check character bytes. Each data block can optionally have a block modifier added. This modifier is called ETE request and is used for multiple packet messages and end-to-end control.

The protocol uses standard RS-232 asynchronous data communication. The character format used is eight bit, no parity, one stop bit. The data rate can vary from 300 to 9600 baud and will be determined by the application and the available data distribution network.

A packet consists of the following:

- Start of header flag
- Header
- Start of text
- Data blocks
- End flag
- Block check code

8.1.7 XACOM PROTOCOL

XACOM protocol is an equipment/device protocol that is used for communications with XACOM ENCODER devices that emit the messages to satellites.

XACOM protocol is the simplest protocol used in the <System Name> system. The protocol simply sends a message to Xacom encoder device and then waits up to 5 seconds for the reply from the device. The protocol generates a log message to indicate that if a response is received in 5 seconds. Otherwise, an error message is displayed and the protocol retries to send the message to the device. The protocol rings the bell and terminates the message sending process once the number of retries is equal to or more than 3.

9 Running <System Name> : runtime and environment

What is it actually like on that <System Name> server?
What should you look for?
What is `/opt/mds/bin/remterm -p 2027 -t /dev/dum26 -m 30 -T 60 -P ml`?

By now, you should have a good feel for a <System Name> system. The next few pages are for when you are sitting in-front of terminal, trying to make it all make sense: It looked so clear on paper! The following are taken from the <System Name> programs running on a typical installation (Melbourne, mid-<date removed>), each with a brief description.

9.1 RUN-TIME ENVIRONMENT

The following are the important components of the <System Name> run-time environment. These values should all be set – either in the production environment, start.env or set and exported by start.mds. The values supplied provide good defaults, or at least clues as to the type of values should be set.

Env_variable	Description	Example
DATABASE_DIR	Some old (Informix) database files	/opt/mds/mds.dbs
INFORMIXDIR	Where the Informix executables are	/usr
LOG_SCR	What terminal to display log messages on	/dev/tty1p0
LPDEST	Where to route line-printer jobs	ml_net_prn1
MANPATH	A long list of paths to man files	/usr/share/man/ …
MDS_BIN_DIR	All the <System Name> executables	/opt/mds/bin
MDS_C_DIR	<System Name> source code files	/opt/mds/c
MDS_DATA2_DIR	Another data directory - though set may not exist	/var/mds/data2
MDS_DATA_DIR	The root of the data directories, queues, database and history.	/var/mds/data
MDS_DOC_DIR	Various <System Name> documentation	/opt/mds/doc
MDS_EMAIL_DIR	the email spool file location	/var/mds/data/email
MDS_FAX_DIR	faxes are written here before being transferred to the Fax server	/var/mds/data/bfax
MDS_LOG_DIR	the directory for logs to be written to	/var/mds/data/logs
MDS_MACHINE_ID	M,O Melbourne S,X Sydney A Adelaide B Brisbane …	O
MDS_PRINTER	What to use as a printer	cat >/home/cont1/print.log
MDS_SHMIDS_DIR	Location of the agreed IDs for Shared Memory blocks	/opt/mds/shmids
MDS_TEXT_DIR	Message storage directory	/var/mds/data/text
MDS_TMP_DIR	The temp directory	/opt/mds/tmp
OUR_CITY	What machine we are on.	mdssms01
PATH	The usual huge long list of directories to search	/usr/bin:…
TRUFAX_DIR	Another Fax directory	/opt/mds/trufax

Table 33 : Important <System Name> Environment Variables

9.2 PROCESSES - WHAT YOU SHOULD SEE ON A RUNNING SYSTEM

As noted above, the following is the output from a 'ps -ef' process listing on a production system. The 100 or so root process have been omitted.
These will help you make sense of your running system. The most important programs to look out for are:

- mdsalarm: Monitors the network for fault and/or problem conditions
- skymds00. This is the log-on process. One copy stays running for each user, 63 when this ps was done
- skymds01. The main program that runs the call-centre user interface to <System Name>.
- skymds02: connects external users' computers to <System Name> : PET Protocol
- skymds03: reads messages to each of one or more destination queues. The queues are passed on the command-line and displayed in the following list
- skymds20: extracts messages from the Bulk Queue for delivery to a sub-queue
- skymds70: connects external users' computers to <System Name> : MERP Protocol

Process	Brief Description
/opt/mds/bin/keepopen /dev/tty1p0	Keep a terminal open for log messages
/opt/mds/bin/mdsalarm -D -o SSVIC/SSADE -q 30 -d 100 -u 70 -	Monitor the network for problem conditions and raise appropriate alarms
/opt/mds/bin/mdsem01 -R -t /dev/dum73	email-to-pager processor
/opt/mds/bin/mdsem02 -R -i kaja -d SAU -t /dev/dum74	email-to-SMS processor
/opt/mds/bin/mdsem03 -R -i bska -m 50000 -d BULKS -t /dev/du	email-to-BulkSMS. Send the text to one of four Priority Bulk SMS queues
/opt/mds/bin/mdsmtps -t /dev/mlc22	MTPS server. For special uses, customised solutions.
/opt/mds/bin/mdssms01 -t /dev/dum77 -n 30 -p 5	Extract Bulk SMS messages from the Bulk-SMS queue and send to the main Bulk Queue
/opt/mds/bin/mdswq -s 5	Monitor the queues
/opt/mds/bin/pm_dtc -x -d /opt/mds/bin/dtc_conf/x25007	??? : 2 copies
/opt/mds/bin/remterm -p 2027 -t /dev/dum26 -m 30 -T 60 -P ml	Supplies telnet connections to <System Name> : 11 copies
/opt/mds/bin/skymds00	Logon screens for terminals : 63 copies
/opt/mds/bin/skymds00j 0	Client Maintenance Screen : 4 copies
/opt/mds/bin/skymds01 0	Message Entry Screen : 10 copies with parameter 0, 22 copies with parameter 1
/opt/mds/bin/skymds02 -D -s 4 -S 12 -t /dev/mla29	Processes direct connections to <System Name> that use the PET protocol : 26 copies
/opt/mds/bin/skymds03 -D -t /dev/mld15 -d NZL1 -l 3091 -P 7	Destination Queue : Pager messages to be forwarded to New Zealand
/opt/mds/bin/skymds03 -n -t /dev/dum044 -d BULKS	Destination Queue : Bulk SMS queue
/opt/mds/bin/skymds03 -t /dev/dum001 -d ADE2 -l 3001	Destination Queue : Adelaide Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum002 -d ADEL -l 3002	Destination Queue : Adelaide Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum003 -d BNE2 -l 3003	Destination Queue : Brisbane Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum004 -d BNE4 -l 3004	Destination Queue : Brisbane Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum005 -d BNES -l 3005	Destination Queue : Brisbane Dual Frequency Pager queue
/opt/mds/bin/skymds03 -t /dev/dum006 -d ACT4 -l 3006	Destination Queue : ACT Frequency 4 Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum007 -d ACT5 -l 3007	Destination Queue : ACT Frequency 5 Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum008 -d ACT -l 3008	Destination Queue : ACT Dual Frequency Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum009 -d EQUA -l 3009	Destination Queue : "Equatorial" - Satellite transmission queue
/opt/mds/bin/skymds03 -t /dev/dum010 -d MEL1 -l 3010	Destination Queue : Melbourne Frequency 1 Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum011 -d MEL2 -l 3011	Destination Queue : Melbourne Frequency 2 Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum012 -d MEL3 -l 3012	Destination Queue : Melbourne Frequency 3 Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum015 -d PERA -l 3015	Destination Queue : Perth Dual Frequency Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum016 -d PERB -l 3016	Destination Queue : Perth Dual Frequency Pager Queue
/opt/mds/bin/skymds03 -t /dev/dum017 -d SYD1 -l 3017	Destination Queue : Sydney Frequency 1 queue
/opt/mds/bin/skymds03 -t /dev/dum018 -d SYD2 -l 3018	Destination Queue : Sydney Frequency 2 queue
/opt/mds/bin/skymds03 -t /dev/dum019 -d SYD3 -l 3019	Destination Queue : Sydney Frequency 3 queue
/opt/mds/bin/skymds03 -t /dev/dum020 -d SYD4 -l 3020	Destination Queue : Sydney Frequency 4 queue
/opt/mds/bin/skymds03 -t /dev/dum021 -d SYD5 -l 3021	Destination Queue : Sydney Frequency 5 queue
/opt/mds/bin/skymds03 -t /dev/dum022 -d SMS2	Destination Queue : SMS queue - to Optus SMSC
/opt/mds/bin/skymds03 -t /dev/dum023 -d SMS3	Destination Queue : SMS queue - to Vodaphone SMSC

Table 34 : Processes on a <System Name> System

Process	Brief Description
/opt/mds/bin/skymds03 -t /dev/dum026 -d FAX	Destination Queue : A fax queue
/opt/mds/bin/skymds03 -t /dev/dum027 -d DUMY	Destination Queue : Advisories
/opt/mds/bin/skymds03 -t /dev/dum032 -d EML	Destination Queue : Melbourne email queue 1
/opt/mds/bin/skymds03 -t /dev/dum034 -d REM	Destination Queue : Reminders for the Call Centre Operators
/opt/mds/bin/skymds03 -t /dev/dum038 -d FA2	Destination Queue : Melbourne email queue 2
/opt/mds/bin/skymds03 -t /dev/dum039 -d EM2	Destination Queue : Melbourne email queue 2
/opt/mds/bin/skymds03 -t /dev/dum040 -d HAM1	Destination Queue : Hammesley (Client Pager Netwrok)
/opt/mds/bin/skymds03 -t /dev/dum041 -d MIT1	Destination Queue : Forward to Mitsubishi
/opt/mds/bin/skymds03 -t /dev/dum042 -d ADE6	Destination Queue : Adelaide Frequency 6 queue
/opt/mds/bin/skymds03 -t /dev/dum046 -l 3192 -d SAU	Destination Queue : <Company Name> SMS queue.
/opt/mds/bin/skymds03 -t /dev/dum047 -d VODA	Destination Queue : <Company Name> SMPP queue (not Vodaphone)
/opt/mds/bin/skymds03 -t /dev/mlb26 -d MEL5 -l 3014	Destination Queue : Melbourne Frequency 5 pager queue
/opt/mds/bin/skymds03 -t /dev/mlc04 -d SNZ -C -l 3092	Destination Queue : SMS messages forwarded to Telecom New Zealand
/opt/mds/bin/skymds03 -t /dev/mlc27 -d MEL4 -l 3013	Destination Queue : Melbourne Frequency 4 pager queue
/opt/mds/bin/skymds06 -t /dev/dum00 -g 180	Timed Event processor
/opt/mds/bin/skymds08 -T -n 3 -t /dev/mlc30	Interface to XACOM pabx/indial processor
/opt/mds/bin/skymds10 -t /dev/vcg03 -P -a 3101 -b 6301 -l 30	Message delivery module. Receive msg from IP and deliver message to a TNPP device.
/opt/mds/bin/skymds11a -D -t /dev/dum60 -dOPTUS_SAU -l 3457	SMPP Client.
/opt/mds/bin/skymds20 -t /dev/dum04	Extract messages from the bulk queue (mdsqqm.txt)and put them in the real queues.
/opt/mds/bin/skymds23l -a MM -b AA -p 2006 -v -h lnka -t /de	Extract messages from the Maintenance Queue and send to skymds24 in Adelaide
/opt/mds/bin/skymds23l -a MM -b MH -p 2003 -h lnkh -t /dev/d	Extract messages from the Maintenance Queue and send them to the History Server
/opt/mds/bin/skymds23l -a MM -b MO -p 2001 -h lnko -t /dev/d	Extract messages from the Maintenance Queue and send to the other Melbourne server
/opt/mds/bin/skymds23l -a MM -b SS -p 2021 -v -h lnks -t /de	Extract messages from the Maintenance Queue and send to skymds24 in Sydney
/opt/mds/bin/skymds24l -p 2079 -r -t /dev/dum08	receive Bulk Queue messages from other servers : 4 copies
/opt/mds/bin/skymds27 -t /dev/dum36 -d CONT -l 96252082	Extracts messages from the "Contacts" queue for special pr altered handling
/opt/mds/bin/skymds28 -p 2033 -t /dev/dum20	Network CISAM file server.
/opt/mds/bin/skymds67a -D -t /dev/dum70 -d OPTUS_SAU	Process packets from SMSC (SMPP V3.4)
/opt/mds/bin/skymds70 -o -s 10 -t /dev/mlb18	Process MERP protocol direct connections to <System Name> : 8 copies
/opt/mds/bin/skymds81 -t /dev/dum41	Network Server Module.
/opt/mds/bin/skymds83 -t /dev/mlc06 -s 10 -n 30 -m 9 -T 5	Network Delay monitoring.
/opt/mds/bin/skymds84 0	Port Maintenance Screen.
/opt/mds/bin/skymds89 -c10 -d5 -i15 -t/dev/mlb23	Get Zach time and set a HP system clock every hour.
/opt/mds/bin/skymds96a -D -n SSVIC,SSADE -t /dev/dum67	Process pager messages from the Oracle database
/opt/mds/bin/tstty -t mla09 -h mla -p 3009 -w	Terminal program : 103 copies

Table 35 : Processes on a <System Name> System (continued)

10 <System Name> Glossary

A	The letter code for the Adelaide <System Name> server.
Advisory (ADV)	A message that the Call Centre operator reads out to a caller. Unlike Temporary Messages *q.v.* Advisories are a permanent arrangement
Airmail	<Company Name> Airmail. A two-way paging service offered by <Company Name> Communications.
A-Party	The sender (the A-party) must be registered with both <System Name> and <SYSTEM NAME 3>, and the sender (the A-Party) is charged for the communication. A CPP (Calling Party Pays) protocol.
APD	???
AT&T	A huge American telco, in a former life a gigantic monopoly. Broken up into the "Baby Bells", one of which (Bell South) was a former owner of <Company Name> Communications.
Audit	a <System Name> record of changes in a client's messaging set-up.
Austas	Oldest <Company Name>'s ancestor. A telephone answering company founded in 1961, that both grew to be the largest in Australia and the foundation of <Company Name> Corporation.
AUSTPAC	The public Australian X.25 network.
B	The letter code for the Brisbane <System Name> server.
Bell South	A former owner and, to a large extent, builder of <Company Name> Communications.
Blocking File	A file containing permission information for A-Party and B-Party message senders and destinations
Booked Page	Pager message pre-set to be sent at arranged times/dates
B-Party	The destination of the message (the B-Party) must be registered with <System Name>. Call costs are covered by the B-Party's existing <System Name> payment arrangements.
Brooke Decoder	Like an advanced pager, a Brooke decoder receives and decodes pager messages on any frequency
C	The programming language that the <System Name> system is written in.
Call Centre	A room or rooms of people employed to answer telephones, usually computer assisted. <Company Name>, especially its <System Name> system, is focused on supporting and maximising call centre resources.
Call Interflow	The ability to process any call through any <System Name> server.
Call Patch (CPA)	Make the Call Centre Operator place a call in response to a caller's <System Name> option choice.
Cap Code	A unique number built in (not assigned) to each pager. In <System Name>, one can address messages to a particular pager's cap-code instead of via the <Company Name>-assigned pager number
Carrier	A company that owns a Telecommunications transmission network. <Company Name> operates as a pager message carrier for other telecommunications companies, while using Optus and Vodaphone as carriers for its SMS messages.
CAS	Corporate Answering Service

CDMA Carrier Detects Multiple Access. A Cell-phone data transfer protocol, used to refer to phones and systems that implement the CDMA protocol.

CDR Call Detail Record. Data about a call kept for billing purposes.

C-ISAM The type of database used by <System Name>.
C : the tables are directly updateable by C programs
ISAM : Indexed Sequential Access Method

Contact (1) A system of advanced, customised processing of messages.
Contact (2) A synonym for 'service'. A user with three services (e.g. pager, email and Fax) has three contacts

CPP Calling Party Pays. Referred to as A-Party model in <System Name>.

CTI Computer Telephony Integration. A computer system that interacts with a PABX to extract technical information from the telephone line while a call is being made.

curses a method of writing dialogs, sub-windows, text entry prompts entirely using letters and punctuation, e.g. '--------' to draw a line. <System Name> uses curses for its message-entry interface,.

DataPage A paging company, founded in 1985. DataPage was one of the many companies conglomerated into <Company Name> Communications.

DDN Digital Data Network. DDN lines provide a fixed bandwidth between two points within a wide area network (WAN).

Delay Pager Provides the ability for messages entered between certain times to not be delivered till a later time.

Distribution Transmitters Transmitters that distribute the paging messages from a <System Name> site to a network of paging transmitters in a city. One distribution transmitter per frequency.

DMS Telstra's "Digital Metropolitan Service". Used by <System Name> for sending (ASCII) signals to the satellite dish and by external customers with permanent connections to <System Name>

duress A synonym for 'contact' *q.v.* as a result of an alarm or concern state in the '<Other Company 2>' employee monitoring system.

Email (EML) A 'contact' (*q.v.*). Send an email from <System Name> to an external destination

Encoder A hardware device that translates paging messages (text) into signals to be broadcast by the distribution and paging transmitters.

Environment Environment Variables. On a Unix computer, a list of names with associated text values used to convey information to computer programs.

EPPIX A computer system to which <System Name> connects. <System Name> is responsible for providing call detail records to the EPPIX system so that <Company Name> Messenger customers can be billed. See CDR.

eq The two letter code referring to the 'equatorial' *q.v.* satellite transmission queue.

Equatorial Used in <System Name> to refer to the Satellite transmission of messages. From the name of the company: "Equatorial Satellite Systems" who were the former owners and operators of the system.

Equatorial Queue The queue of messages awaiting transmission (via the X.25 network, the Sat-MUX, and Telstra's DSP network) to <Company Name>'s Satellite Transmission system for delivery to areas outside the main centres

Equatorial Satellite Systems The Company, bought by <Company Name>, that owned the Satellite Broadcast system that <Company Name> now operates.

ETHERNET A data-<Company Name> layer protocol for use over a local area network (LAN). Ethernet

networks typically span an office or a building.

Escalation (ESC) An advance <System Name> message delivery system where messages are repeatedly sent by various mechanisms until and acknowledgement of 'OK I got the message' is received.

ESME External Short Message Entity

Event An Escalation or an escalation acknowledgement

f1 to f6 Labels for the six frequencies on which <System Name> systems transmit messages.
f1 = 149.8875 MHz f2 = 149.8375 MHz f3 = 148.7875 MHz
f4 = 148.9625 MHz f5 = 148.6125 MHz f6 = 148.8125 MHz

Facilities Response Centre A work and resource management system developed for Myer Stores Ltd (<Other Company 1>.).

Faxes (FAX) A 'contact' (*q.v.*). Send a fax

FEMD Front End Mediation Device. An Optus system that <System Name> interfaces with for the purpose of activating, deleting, and updating the SMS customer information.

Follow Me A service that re-routes paging messages to the network of another city while the pager holder is in that city.

Forms (FRM) A 'contact' (*q.v.*). Process the data entered on a customised form

FRC Facilities Response Centre *q.v.*

GL3000 The messaging system used by SAGRN *q.v.*.

Group Messages A single message is sent to a number of pagers simultaneously.

Group Paging (GRP) One paging in dial is assigned to a group of related pagers. The caller tells the call centre operator which of the pagers he/she wants to send a message to.

GSM Global System for Mobile communications [originally: *Groupe Speciale Mobile* a pan-European ISDN compatible digital cell-phone standard (low-capacity high-mobility system.]
Frequency bands: 905-914 MHz, 950-959 MHz, hi speed data transmission (19.2 up to 64 kb/s).

History The collection of details of all the previous year's <System Name> messages

History Server The machine dedicated to the collection and conglomeration of <Company Name>'s history data. (named 'lnkh')

HP 9000 A computer, the brand (Hewlett Packard) and model : 9000 series, that the <System Name> systems uses.

IIS Internet Information Server. A Microsoft internet data request and supply management system.

in dial The phone number *from* which a person is calling a <System Name> call centre.

infopack (~pak) Some advice to the Call Centre Operator for a particular in dial.

Informix The brand of database. <System Name> uses a C-ISAM (*q.v.*) Informix database.

InstaPage A former Australian paging company. Like Austas, DataPage and others, became part of the <Company Name> Corporation.

Interflow Call Interflow. The ability to process any call through any <System Name> server.

ISDN	Integrated Services Digital Network. An international standard for a network that handles digital data (no analogue signals). The network is capable of providing a range of services as it can handle voice, fax, and computer data.
IVR	Intelligent Voice Recognition, or Interactive Voice Response. The caller interacts verbally with a computer and a computer voice, the computer managing the "conversational" interaction.
<System Name>	a messaging system, delivering messages *from* telephone, via a call centre or directly, from the internet and 'special-delivery' sources *to* pagers, fax machines, mobile phones and email accounts.
<System Name 2>	A Web-based bionic wrapper of, and addition to, <System Name>
LMS	"<Company Name> Mediation System". An Oracle system managing billing for usage of the <System Name> system
M	The letter code for the primary Melbourne server. The other Melbourne server is referred to by the letter O.
mds??	**NB: A complete listing of <System Name> programs appears elsewhere in this documentation.**
mdsalarm	Monitors <System Name> for error conditions, and sends alarm messages appropriately.
mdsem01	Extracts the B-Party (normal) messages from the email message queue, posting them to the Bulk Queue.
mdsem02	Extracts the A-Party (Sender Pays) messages from the email message queue, posting them to the Bulk Queue
mdsem03	Processes those emails that are to become Bulk-SMS messages, generating up to 50,000 SMS messages and storing them in one or more of four "priority" (Bulk SMS) queues.
mdsemgw	Runs the email Gateway.
mdsms01	Extract SMS messages from the four "priority" (Bulk SMS) queues, transferring them to the Bulk Queue.
Meatspace	The ever shrinking domain that isn't cyber-space, i.e. reality
MERP	Message Entry and Retrieval Protocol. A protocol for allowing a customer computer connected to the <System Name> system to act as a source and destination of messages.
Metagram	A former Australian paging company. Like Austas, DataPage, and others, Metagram has become part of the <Company Name> Corporation.
MHz	A 'million per second'. Used to represent electromagnetic (transmission) frequencies, e.g. 149.8875 MHz, meaning 149,887,500 wave-peaks per second. Not 'mHz', 'Mhz', or 'mhz'.
MTPS	Message Transfer Protocol System, a network data transfer protocol developed by <Company Name> and used for transferring data over some leased-line connections.
Myers	A major <Company Name> client, for whom extensive custom applications have been written.
O	The letter code for the secondary Melbourne server. The other Melbourne server is referred to by the letter M
Optus	Another Australian Telecommunications company with which <Company Name> has some dealings.
OS/2	The operating system (like, say, Microsoft Windows, Unix) that the Interactive Voice Response software runs on.
P	The letter code for the Perth <System Name> server.

PABX	Private Automatic Branch eXchange – the in-house telephone system management system.
Pager	A device to which textual messages can be sent. These devices are typically small and can be worn by a person. A sound such as a beep is usually used to indicate the arrival of a message.
Paging (PAG)	Send a normal pager message as a 'contact' (*q.v.*).
PAM	An interface, coded in C, to an arbitrary authentication/security system.
Paging Transmitters	The transmitters located through-out each major city that first receive pager messages for re-broadcast to people's pagers. Each paging transmitter is, generally, set to transmit on a single frequency, though some switch between two frequencies.
Personalised Paging	One pager in dial per pager, as opposed to a standard in dial and individual pager numbers to be quoted by the caller.
PET	Pager Entry Terminal. A protocol for allowing a customer computer connected to the <System Name> system to act as a source and destination of messages.
pid	The (Process Identification) number a Unix system uses to identify each running process.
Priority	For Bulk-SMS. There are 4 queues, each of which sends Bulk SMS messages with a different urgency, priority '1' being the most urgent, priority '4' being the least urgent.
Provisioning	The allocation of a new phone-number, and the process of making other telecommunications companies aware of the new telephone number
PSTN	Public Switched Telephone Network. The general telephone network that people can use to make calls within Australia.
Queues	Flat (text) files that buffer messages internally within the <System Name> system. See section 7.3.
Reminders (REM)	Messages to be sent to call centre operators themselves, usually prompting some action, e.g. calling a customer
root	The super user on a Unix computer system who has full access to every part of the system without any security checking.
Rosters (ROS)	Messages are handled in different ways at different times of the day/week/year.
Router	A device that connects two or more TCP/IP networks together, enabling data transfer between them.
S	The letter code for the primary Sydney server. The other Sydney server is referred to by the letter X
<Other Company 2>	A worker safety tracking system implemented as a External Database contact in <System Name>. Will be moved to <System Name 2>.
SAGRN	The South Australian Government Radio Network. <Company Name> processes and forwards messages to this network.
Sat-MUX	Satellite Multiplexer
sendmail	The Unix operating system that manages emails. sendmail runs the mdsemgw (<System Name> email gateway) program each time an email arrives at any <System Name> domain.
Semaphore	A simple structure containing an ID number and a counter, used by programs to track resource usage, the counter being incremented and/or decremented as the resource is used. Explicitly supported by the Unix operating system.
Shared Memory	A block of program memory that two or more programs can simultaneously access. For transferring information between programs. Explicitly supported by the Unix operating system.

Signal	A method of sending simple coded commands to programs while they are running. Almost exclusively used for program control and status manipulation. Explicitly supported by the Unix operating system.
<SYSTEM NAME 3>	A computer system to which <System Name> connects. Statistical information is sent to this system for billing purposes. Also customer activation, deletion and update transactions are sent between the two systems.
skymds??	**NB: A complete listing of <System Name> programs appears elsewhere in this documentation**
skymds01	The program that manages all the Call Centre operator screens, integrating the execution of the other programs that run behind the scenes
skymds02	Processes direct connections transferring messages in the PET format.
skymds03	skymds03, using various modules, transfers messages from the bulk queue to various external destinations.
skymds06	skymds06 generates timed events, e.g. pager messages automatically sent at the same time each day.
skymds08	Interface to XACOM pabx/indial processor. skymds08 interacts with the IVR (interactive voice response) server to obtain pager message details from those who have had their pager-message call processed by a computer.
skymds10	Interface to the South Australian Government Radio Network (SAGRN) system. Uses the TNPP protocol.
skymds11t	Sends SMS messages to the SMS Centres of other Telecommunication companies.
skymds11r	Receives sent-SMS details from external SMS Centres.
skymds18	Used to transfer occasional messages from <Company Name>'s <SYSTEM NAME 3> billing system to <System Name>'s bulk queue.
skymds20	Extracts messages from the Bulk Queue for delivery to distribution queues
skymds27	Manages and generates messages for alternative processing. Referred to as the "Contact" system.
skymds70	Processes direct connections transferring messages in the (<Company Name> Proprietary) MERP format.
<Former Company Name> (1)	A former Australian paging company. Like Austas, DataPage and others, became part of the <Company Name> Corporation. The "sky" features in about half of the <System Name> programs : skymds03, skymds78, skymds11a …
<Former Company Name> (2)	The system paging system inherited from <Former Company Name>. <System Name> was built on top of <Former Company Name>..
<Former Company Name> (3)	The company that the combined InstaPage, (former) <Former Company Name>, and VoiceCall was (re) named.
SMPP	Short Message Peer to Peer. The protocol used to send SMS messages
SMS	Short Message Service. A service available for GSM mobile phones that allows a textual message to be sent to the phone and displayed on the handset screen.
SMSC	Short Message Service Centre. The centre that is responsible for handling the delivery of the SMS messages to the mobile phones.
SMTP (1)	Simple Mail Transfer Protocol [layer 6, Internet standard]
SMTP (2)	Short Message Transport Protocol

start.mds	The script that starts a <System Name> system, correctly executing and establishing each program
STARTEL	An old, retired system that <Company Name> at one time used to support <Other Company 3> in Australia. This functionality has since been moved to an 'External Database' contact application. Will move to <System Name 2>.
system reference number	A number generated by <System Name> (skymds01) that is given out to a caller as their reference number regarding this call. Currently produced only by customised forms.
TCP/IP	Transmission Control Protocol / Internet Protocol. Two protocols in the internet suite of protocols. Typically used to refer to a network that is using the protocols and applications belonging to the internet suite.
Telstra	Another Australian Telecommunications company with which <Company Name> has some dealings
Temporary Message	A message that the Call Centre operator reads out to a caller. Unlike Advisories *q.v.*, Temporary Arrangements, as the name suggests, are a short-term arrangement, e.g. "I'm on holiday, please call John Smith on …"
Timed events	The <System Name> system that records and triggers all timed message sending; more than you may think Implemented by skymds06.
TMS	"Text Message Service", a 'contact' (*q.v.*) that instructs the Call Centre Operator to record a message from the user for later recall for the client.
TNPP	Telocator Network Paging Protocol. An inter-paging-system communication protocol used to connect to the SAGRN *q.v.* GL3000 system.
Two-way SMS	Using a small keyboard (or not, in some cases), either attached to a cell phone or built into a pager, a customer can reply to a pager message as soon as it arrives, on the pager (or cell-phone itself). Similar to two-way SMS messaging, but using metropolitan wireless networks instead of the cell-phone networks.
TXT	"Text", a 'contact' (*q.v.*), like TMS, but stores the messages in a text file, the file itself delivered.
VIP	A home services franchise system that has a set of contact (*q.v.*) functionality custom made.
Vodaphone	Another Australian Telecommunications company with which <Company Name> has some dealings
<Other Company 3>	The car dealership for whom a <System Name> custom solution has been developed.
VoiceCall	An Australian Telecommunications company that was bought by Bell South to become part of <Company Name> Communications.
X	The letter code for the secondary Sydney server. The other Sydney server is referred to by the letter S
X.25	packet switched data service, with channels up to 64kb/s, based on the three lower Levels of the OSI model. The 25th standard in the X series of standards. X.25 is an international standard for packet switched data networks.
Xacom (1)	A brand of devices used by <System Name>. Examples include the encoders that translate text pager messages to pager signals and the CTI server
Xacom (2)	A communications protocol used for communicating with various Xacom brand devices.
XENIX	A Unix implementation, like HP-UX, BSD, Line and other versions of Unix
zach time	A device that provides accurate time signals to the <System Name> system