

Technical Manual

2nd August 2004

Contents

1	Introduction	4
2	Xamime Control	5
2.1	Overview	5
2.2	Starting Xamime	5
2.3	Stopping Xamime (and restarting sendmail without Xamime)	5
2.4	Invoking Xamime manually	5
3	Regular Expressions	6
4	File Types	8
4.1	Introduction	8
4.2	Test Methods	8
4.2.1	Dynamic Text Detection	8
4.2.2	Fixed Binary Detection	8
4.3	filetype.list	9
4.3.1	Field explanations	9
4.4	Compiling filetype.list into filetype.spec	10
5	File Descriptions	11
5.1	Introduction	11
5.2	Files	11
6	Main Menu Enhancements	16
6.1	Introduction	16
6.1.1	Main Menu Extensions	16
6.1.2	Main Menu Side-bar	16
7	Report Scripts	19
7.1	Introduction	19
7.2	Report Script files	19
7.3	Parameters	19
7.4	Invoking Method	19
8	AntiVirus	20
8.1	Introduction	20
8.2	Fixed AntiVirus Engine Method	20
8.2.1	Invoking	20
8.2.2	Output Filtering	20
8.2.3	Filtering Example	20
8.3	Generic AntiVirus interface	21
8.3.1	Introduction	21
8.3.2	Generic AntiVirus requirements	21

9 Postfilter	22
9.1 Introduction	22
9.2 Postfilter URI data format	22
10 Debugging	25
10.1 Introduction	25
10.2 Global Debugging	25
10.2.1 Activating	25
10.2.2 Output	25
10.3 Command Line Debugging	25
11 Error Messages	28
11.1 Introduction	28
11.2 Things to check	28
11.2.1 Disk utilization	28
11.2.2 Mail processing	29
11.2.3 System CPU and memory resources	29
12 Additional Tools	31
12.1 confcompile	31
12.2 buildxamcf	31
12.3 xmapasswd	32

1 Introduction

This manual is intended to be used as a technical level reference for dealing with Xamime on a non-WWW interface level. Most operations of Xamime are performed via the XAdmin interface. However, there may be situations/events which require low-level maintenance or adjustments which cannot be directly performed through the XAdmin interface.

This manual is also intended to be used as an information source for various commonly needed data tables.

2 Xamime Control

2.1 Overview

This chapter deals with the process of starting and stopping Xamime, as well as detailing how to invoke Xamime manually for the purpose of testing/debugging.

2.2 Starting Xamime

Xamime can be started with the following command. NOTE the killall statement is optional, and only required if there are other sendmail processes running.

```
killall1 sendmail; sendmail -bd -q1h -C /usr/local/xamime/xamime.cf
```

The flags *-bd -q1h* are dependent on what your typical installation for sendmail is. The *-bd* invokes sendmail as a daemon (sends it to background). The *-q1h* forces sendmail to re-queue every 1 hour.

2.3 Stopping Xamime (and restarting sendmail without Xamime)

Sometimes it is required to stop Xamime from filtering email, perhaps in order to ascertain the location of various routing issues or other such.

```
killall sendmail; sendmail -bd -q1h
```

Check additionally in your startup scripts for Xamime modifications (typically */etc/init.d/rc.sendmail*) if you need to disable Xamime for a greater duration of time.

2.4 Invoking Xamime manually

For the purpose of debugging, or perhaps as part of a periodic script, Xamime can be invoked directly without calling Sendmail initially. Xamime accepts three (3) parameters on the command line. It receives data intended for delivery [RFC822 mailpack] via STDIN.

```
cat mailpack | xamime <conf file> <sender address> <receiver address>
```

¹killall should not be used on Solaris systems. The killall on Solaris can be fatal to all processes running.

3 Regular Expressions

Regular expressions form a major component of Xamime operations. Regular expressions are used for nearly all text matching routines. It is important to learn how to use regular expressions efficiently as while they are very powerful they can also bring your CPU to maximum load if abused.

Xamime uses regex matching in the following areas:

- ACL Name (matching to email address)
- Sender Test
- Receiver Test
- Text Test
- Filename Test
- Generic AntiVirus output filtering

Of all the above tests, the Text Test ranks the highest in terms of potential load. If your system is suffering under a lot of CPU load, check to see what Text Tests you have active and see if you can perhaps optimize them (ie, conglomerate separate tests into one OR test, such as `\(wordA|wordB|wordC)`).

All regex testing is done in case insensitive mode. This cannot be changed. If you require specifically a case sensitive regex search, consider using an External Test to perform the operation.

Xamime will report any errors encountered in the ACL regular expressions in the system log files with reports similar to:

```
ERROR: Regex compilation resulted in code 13 for regex '*.com'
```

Table 3.1: Compact Regular Expression Guide

Regular Expression	Definition
\t \n \r	Match a tab, newline, carriage return
\\ * \+ \?	Match escaped character literally, ignoring the \
[...]	Match any characters contained within the brackets
[^...]	Match any characters not contained within the brackets
.	Match any character excluding the newline (\n)
\w \W	Match any word, or NON-word
\s \S	Match any whitespace or NON-whitespace
^ \$	Match expression to the beginning or end of a line
\b, \B	Match at word boundary (or NON-boundary)
?	Match zero or one time
+	Match previous term one or more times
*	Match term zero or more times
{n}	Match previous term precisely n times
{n,}	Match previous term n or greater times
{n,m}	Match previous term at least n times, but no more than m times
a b	Match either a or b

4 File Types

4.1 Introduction

Although the filetype.spec/list file is not considered an end-user administrable item of Xamime, its specification has been included here for those administrators whom which to expand further on the available file types available for detection. It should be strongly noted here that any alterations made will be ignored and potentially overwritten with the release of any new versions of Xamime.

4.2 Test Methods

Xamime currently uses two different methods of file type testing, dynamic text and fixed binary. The requirement of two different style of testing methods arises from the desire to detect the vast array of file types available to computing.

4.2.1 Dynamic Text Detection

Dynamic Text Detection testing came about in order to fill the need to detect and “type” many file types which do not exhibit a fixed byte-spacing structure. Such formats include HTML, Shell scripts and programming source code (amongst many others).

The test type field designation for Dynamic Test Detection is “1” (as opposed to “0” for binary). The test-length field sets a byte count limit for the detection range. Should the file type not be “detected” within that range of bytes, the test is deemed unsuccessful.

Each word required to be detected (in sequence) is separated by a caret (^). If the word is optional, it is further prefixed using a tilde (~). If the word is not to be detected, it is prefixed using a exclamation (!).

An example of a Dynamic Text Detection to detect C header files could be:

```
~extern^#define^~extern^!main(
```

Dynamic Text Detection can be used on binary files and still match word sequences without having to worry about null/0 byte sequences causing short reads or premature test failures. Xamime will prefilter the testing block of file, replacing any null characters with ASCII 1

4.2.2 Fixed Binary Detection

Unlike the Dynamic Text Detection, Fixed binary (as it name indicates) relies on detecting specific sequences of bytes in specific static locations. As most binary sequences involve characters outside of the normal ASCII 7-bit range, Xamime provides two methods for encoding byte values

Decimal notion

Decimal values are encoded into single bytes using a slash prefixed three digit notion.

```
\ddd - \034 \138 \232
```

Hexadecimal notion

Hexadecimal values are encoded into single bytes using a slash-x two [hex]digit notion.

```
\xHH - \x40 \x39 \x9C
```

Additional operators

Additionally, it is often required to apply an AND bit mask to a byte value and compare for an identical result. To apply an AND test to a fixed binary sequence, the required byte must be prefixed with a “&” symbol.

```
&BYTE - &\x80 &J &\032
```

4.3 filetype.list

4.3.1 Field explanations

The text contents of Xamime filetype.list are colon separated single record per line entries. Each field will be described here

ID

This field is no longer used. However, for the sake of backward compatibility, it is still required to be present with a value of '-’.

OFFSET

0-offset based byte value indicating at which byte [in the file] to commence testing from.

TEST-TYPE

- 0 - Use Fixed Binary test
- 1 - Use Dynamic Text test

TEST-LENGTH (applicable only to Dynamic Text test)

The maximum number of bytes to test from the start of the file. The absolute maximum default length is 10KB (10240 bytes).

GR[123]

Progressive category heirachy of the file type. This permits an ACL-item to select a group of files based purely on the category in which a file exists, as opposed to a specific file type.

NAME

Often considered as a GR4 field, This field should be considered an absolute and unique identifier for a file type

TESTFOR

The sequence of bytes or words for which must be tested for by either the Fixed Binary or Dynamic Text tests

COMMENT

A comment of up to 512 bytes in size used to describe the file type being tested for

4.4 Compiling filetype.list into filetype.spec

Due to the performance demands, the Xamime program loads the file-types in a compiled form from a file called filetype.spec. The file filetype.spec is a compiled version of filetype.list. To compile *filetype.list*, it is required to use the *buildtypes* program [included with Xamime distributions]. Typical use is as follows:

```
./buildtypes filetype.list filetype.spec
```

A normal run of *buildtypes* will result in a similar output to what follows:

```
# ./buildtypes filetype.list filetype.spec
Reading in text database...
65 types read in...
Clearing database...
Reading in binary database...
65 types read in from binary database
Types database verified.
```

5 File Descriptions

5.1 Introduction

Xamime as a package contains numerous files, each which perform a specific task, or are used/created as part of the running process of either Xamime or XMAdmin.

5.2 Files

.gif

All gif format files (non-LZW compressed) are used by the XMAdmin program to provide iconic representations of the many functions in XMAdmin (ie, delete, deny-status, pass-status, view etc). These icons can be replaced/redrawn as desired. Gif is currently being used due to the lack of PNG-transparency support in numerous browsers. Once transparency is supported more widely (and reliably) the default file format will be PNG.

.groups

Text files containing space, command and carriage-return delimited email addresses. The addresses are matched on a strict basis with no regular-expression permitted. Groups are used in ACL name entries to permit the grouping of multiple dissimilar email addresses under one ACL name.

aclitempage.dhtml

This file is loaded every time you view/edit/create an ACLitem. The file contains a majority of the HTML code required to build the ACLitem page. It is not recommended that you modify this file unless you know what you are doing. Should the file become non-functional it can be refreshed from the original install package. This file will be updated when ever a new set of email tests are added to Xamime.

acls

Is a subdirectory containing all current ACLs. Within this directory exists a file *acls.meta* which contains meta information about all the ACLs.

buildtypes

Is a program used to compile *filetype.list* [text format] into *filetype.spec*. See section 4.2 on page 8 for details on file types.

confcompile

Confcompile converts a `xamime.conf` (text) file into an Xamime native readable binary format. This program is used during the initial Xamime installation to convert the parameters generated for Xamime through OS probing into the required `xamime.cfg` file. *You cannot use `xamime.conf` (text) as an xamime configuration file, Xamime will fail to operate.* See section 12.1 on page 31.

genericav.conf

Configuration file containing the binary formatted details for the GenericAV interface. These file should not be modified directly. If you wish to alter the GenericAV configuration, do so via the Xamime WWW interface.

help

This directory contains help files that are used in Xamime. As more help files are added, they will be installed here.

localdomains

A text file containing single domains per-line. Each domain contained within is considered a “local domain” by Xamime. This file is often replaced by the `/etc/mail/local-domains` sendmail file.

logs

Is a subdirectory in which Xamime and XMAdmin logs are stored.

mailadmin, mailreceiver, mailsender

Are template shell scripts used to generate and deliver blocked email reports to the administrator, receiver and sender respectively. This template is copied to an ACL directory for each new ACL created. It is recommended that the administrator modify this script to best suit the message response desired. note - The script can also be rewritten in Perl or any other language, so long as it accepts the required parameters and delivers the blocked message. (See also 7 on page 19)

mainmenu-sidebar.url

This file contains data used to generate the Xamime WWW interface sidebar. This file is plain text and is created by the administrator. Please refer to 6.2 on page 18 for more details.

mainmenu.extensions

This file contains HTML code which will be inserted into the Xamime WWW interface main menu. Please refer to 6.1.1 for more details.

Mxamime.MDA

This file contains the Xamime mail agent definition as required by Sendmail. The file is generated during the installation. The file is retained for reference purposes.

normal.css

Cascading Style Sheet used by XMAAdmin to enforce a common look and feel. This file can be modified by the administrator to provide a look/feel as desired.

options.conf

This file contains configuration information for the Xamime Extended Options. Do not edit this file directly, use the Xamime WWW configuration interface.

postfilter.conf

This file contains configuration information for the Xamime postfilter. Do not edit this file directly, use the Xamime WWW configuration interface.

postfix.conf

This file contains configuration information for communicating with the Postfix MTA server (Xamime Postfix version). Do not edit this file directly, use the Xamime WWW configuration interface.

prefilter.conf

This file contains configuration information for the Xamime Prefilter. Do not edit this file directly, use the Xamime WWW configuration interface.

storage.conf

This file contains configuration information for the Xamime Spanning storage. Do not edit this file directly, use the Xamime WWW configuration interface.

tmp (Xamime Levels 1, 2 and 3)

Is a directory containing all unpacked/in-process email.

- Directories with a suffix of `_B` are blocked and retained.
- Directories with a suffix of `_P` are passed and retained.
- Directories without a suffix are in process,

Should a directory without a suffix persist for more than a minute, it is often worth claiming the mailpack out of the directory, and sending it to the Xamime technical administration email account (see *Xamime WWW site for suitable contact details*) for further inspection.

tmp (Xamime Level 4)

The spanning storage system of Xamime Level 4 extends the `./tmp` directory into 2 more levels. The Xamime Level 4 retained email path format is:

```
xamime/tmp/{selected partition}/{coarse timecode}/inf*
```

- The selected partition is a algorithmically selected partition from one listed in the Xamime storage configuration
- The coarse timecode is a composition of the year, month, day and hour (YYYYMMDDHH).

An example path is:

```
xamime/tmp/2/2004070216/inf1093532234_3322
```

xamime

Is the main filtering program. MTA servers feed all email through Xamime.

xamime.cf

Altered version of the system's original `sendmail.cf`. The `xamime.cf` file contains alterations which cause sendmail to direct all email through xamime.

xamime.cfg

Xamime main configuration file compiled into a binary format by the `confcompile` utility (See 12.1 on page 31). The reasoning behind converting to a binary format rather than the previous text-only configuration file (See `xamime.conf`) is for reasons of speed. A binary format configuration file can be read in with a single read command and immediately has all the contents available in the correct format, conversely, a text format configuration file has to be read in and parsed line by line, which consumes many more times the CPU resources.

xamime.conf

Xamime configuration file. This file contains global parameters used by Xamime. This file is no longer “actively” used. Rather it serves as a source file for compiling the `xamime.cfg` file (on initial install).

xamime.licence

Details the ownership status of the Xamime installation. The Xamime licence file is a simple 3 line text file which is formatted as follows:

- Licence Number (ie 20040201L4)
- Licence Owner (ie PLD)
- Release Code (long digit sequence supplied by PLD Software, uniquely generated)

xmadmin

WWW interface administration tool for Xamime.

xmadmin.access

Controls the access rights of administrators in XMAdmin

xmadmin.conf

XMAdmin configuration file. This file contains global parameters used by XMAdmin

xmadmin.passwd

MD5 encoded encrypted password file for administrator users

xmapasswd

Command line interface user/password modification tool (similar to *passwd*)

6 Main Menu Enhancements

6.1 Introduction

In order to provide a degree of customization of the Xamime WWW interface, it is now possible to add additional menu entries using the MAIN MENU EXTENSIONS facility and to add a separate loading side window using the MAIN MENU SIDE-BAR.

6.1.1 Main Menu Extensions

Main menu extensions allows an extra segment of HTML code to be inserted into the main menu page between the EDIT CONFIGURATION and LOGIN menu entries. XMAAdmin looks for the file *mainmenu.extensions* in the Xamime home directory (*/usr/local/xamime*), if the file is present, it will be loaded. Below is a sample *mainmenu.extensions* file:

```
<UL>
<LI><a href=http://www.freshmeat.net>Freshmeat </a>
<LI><a href=http://www.news.com.au>Australia News</a>
</ul>
```

See figure 6.1 on the next page as an example of the above file included in the Xamime main menu.

6.1.2 Main Menu Side-bar

The MAIN MENU SIDE-BAR allows the Xamime WWW interface to have an embedded IFRAME HTML tag located on the right hand side of the main menu. The file *mainmenu-sidebar.url* file contains the information required to make the IFRAME useful. The file contains two separate lines of data.

1. <TD> tag parameters for the inclusion of the IFRAME
2. <IFRAME> tag parameters

The details contained in the file *mainmenu-sidebar.url* are placed in the WWW page as such:

```
<TD ${first line of mainmenu-sidebar.url}>
<IFRAME ${second line of mainmenu-sidebar.url}>
```

an example of the contents of the file is depicted below:

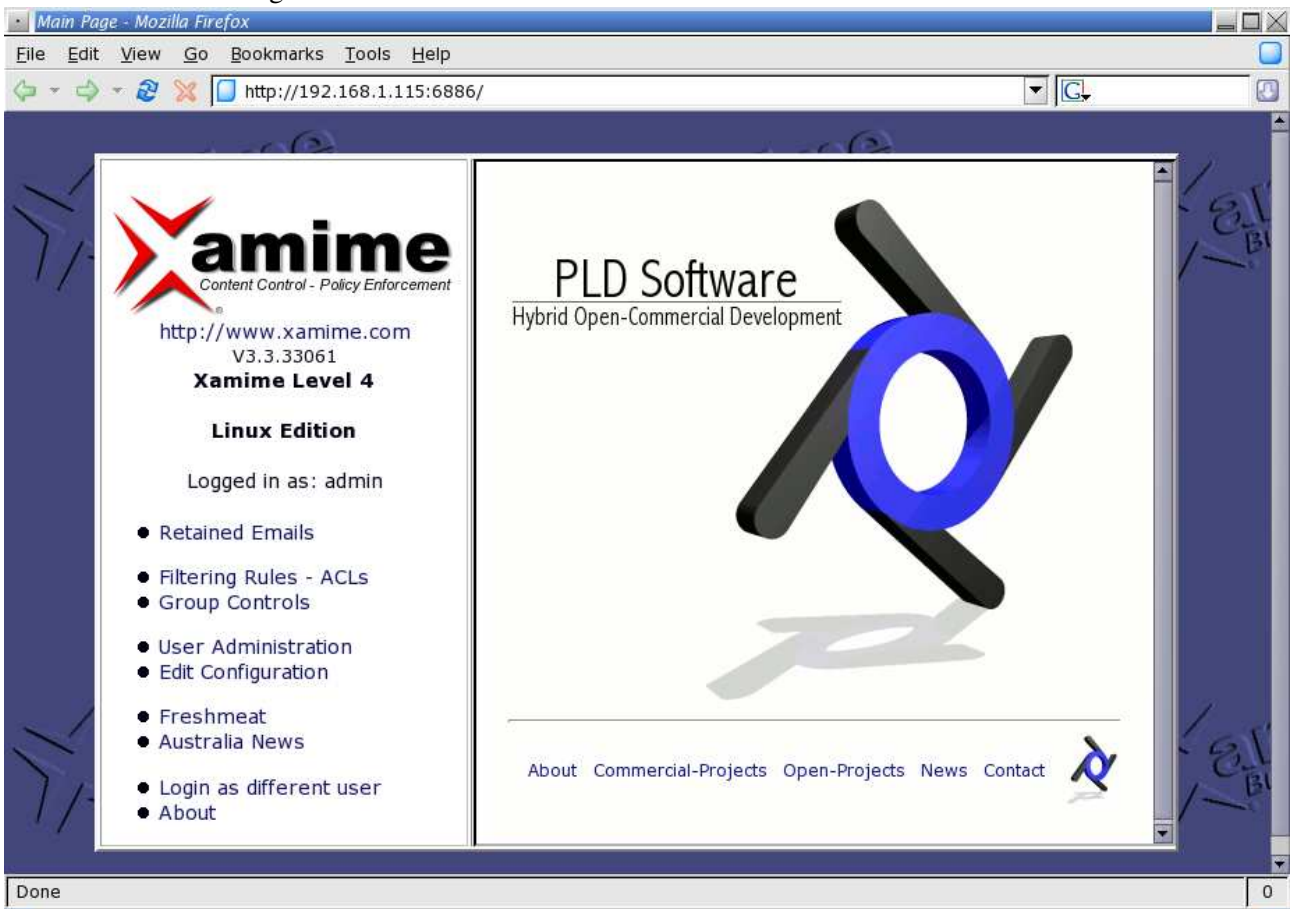
```
width=500 bgcolor=#ffffff
src='http://www.pldaniels.com' width=100% height=100% border=0
```

The resulting WWW page is shown in figure 6.2.

Figure 6.1: Xamime Main Menu with Menu Extensions



Figure 6.2: Xamime Main Menu with Side-bar and Extensions active



7 Report Scripts

7.1 Introduction

Xamime employs an unique script for delivering blocked email reports to each of sender, receiver and administrator. Each script is stored in the ACL subdirectory to which it is applicable.

7.2 Report Script files

When a new ACL is created, the report script files are copied from the Xamime home directory (typically */usr/local/xamime*). The scripts are entirely administrator configurable and not limited to being shell-script only (it's perfectly applicable that the mail report delivery files be written in C or Perl). The filenames for the mail report delivery scripts for sender, receiver and administrator are *mailsender*, *mailreceiver*, *mailadmin* respectively

7.3 Parameters

Each script is called with seven (7) parameters

1. sender email
2. receiver email
3. report name (filename containing the report file which Xamime produced, full path)
4. email temporary directory (directory in which the email was unpacked/tested, full path)
5. RFC compliant date string (Date string used to ensure that the headers of the created email are RFC822 compliant)
6. administrator email address
7. Sendmail default (non-Xamime) configuration file (full path)

7.4 Invoking Method

Xamime invokes each of the three (3) blocked report delivery scripts via a *system()* call. The return code is that which is returned by the delivery script. Typically the script will return the result code from the sendmail delivery.

8 AntiVirus

8.1 Introduction

This section is not about specific AntiVirus programs, but, rather how Xamime deals with invoking AntiVirus programs and filtering their results. There are now two methods of using AntiVirus in Xamime. The first is the fixed AntiVirus engine solution which uses predefined configurations - this is located on the main configuration page. No more AntiVirus engines will be added to this page. The second method is the Generic AntiVirus interface. The Generic AntiVirus interface enables Xamime to cope with an ever expanding and changing market of available engines without forcing the administrator to wait for the latest Xamime updates.

8.2 Fixed AntiVirus Engine Method

8.2.1 Invoking

Xamime invokes external executable AntiVirus programs through the use of the *execve()* C proceeding a *fork()* call. The output from the AntiVirus program is captured by *dup2()*'ing the STDOUT file stream to *antivirus.report*.

8.2.2 Output Filtering

Due the the varying nature of the output of AntiVirus programs, Xamime employs the use of a post-executional filtering call to strip out all non-virus report data. This is required to provide a relatively consistent output for blocked email reports. In addition, removal of the non-virus data ensures that end-users receiving blocked email reports are not overwhelmed with spurious and confusing data.

Full paths are removed in order to prevent broadcasting information which may be of potential use to a cracker.

8.2.3 Filtering Example

Example NOD32 execution output prior to filtering

```
p Loading module...OK
p Scanning log p NOD32 Version 1.106 (20010919)
p Command line: /extra/mailpacks/rm1 date: 29.11.2001 time: 10:56:11
/extra/mailpacks/rm1/Navidad.exe - Win32/Navidad worm /extra/mailpacks/rm1/FRE
number of diagnosed files: 148 number of viruses found: 3 termination time: 10
```

Example of NOD32 output after filtering

```
Navidad.exe - Win32/Navidad worm
FREE_xxx_sites.TXT.pif - Win32/MTX.A
eicar.com - Eicar test file
```

8.3 Generic AntiVirus interface

8.3.1 Introduction

With the more recent releases of Xamime, including the Generic AntiVirus configuration tool, it is no longer required of Xamime to explicitly support a particular AntiVirus program.

8.3.2 Generic AntiVirus requirements

To configure a new AntiVirus program for Xamime via the Generic interface, the following information will be required:

- Binary path
- Binary flags
- Clean/No-virus return code(s)
- Virus detected return code(s)
- Error return code(s)
- Other return code(s)
- Text regular expression for the filtering of lines containing virus information (optional)
- Text regular expression for the extraction of the filename containing the virus information (optional)

Please refer to the Xamime Administration manual for further details about setting up the AntiVirus program in the Generic AntiVirus interface.

9 Postfilter

9.1 Introduction

The Xamime postfilter provides an open interface to the task of getting data to a third party solution. While the Postfilter does allow you to modify the files stored for the email, it is more commonly used to transfer data to SQL databases and reporting systems. More fields may be added to the data stream over time as new features are added to Xamime.

9.2 Postfilter URI data format

The Postfilter data-stream, as delivered to the invoked binary via STDIN is encoded using the URI format. The URI encoded data can be quickly decoded using most languages including PHP and Perl. It should be noted that for recent versions of PHP, it is an absolute requirement that PHP is configured with the *'-disable-cli'* parameter else the data input from STDIN will fail.

Table 9.1: URI Data Specification

Variable Name	Comments	Data example
utstamp	Unix timestamp integer	1093422281
eid	Email ID	1093422281_3322
retained_path	Full path to the email on the Xamime system	/usr/local/xamime/tmp/ 2/20040701/inf1093422281_3322
timestamp	Human readable timestamp	20031025160313 (2003-10-25, 16H03:13)
acl_key	Key of the ACL used to block/pass the email (if applicable)	106290648412129
acl_name	Regex text of the ACL for address filtering	@pldaniels.com, #itgroup
acl_comment	Text comment of the ACL	Global ACL
block_status	Email block status (PASS/DENY/None)	Passed Blocked
block_comment	Comment from the ACLitem used to PASS/DENY	Email is blocked
sender	Email address of the sender	pld@pldaniels.com
receiver	Email address of the receiver	receiver@domain.com
global_status	See table 9.2 on the following page	199
sender_blocked	Email was blocked due to a sender test	1
receiver_blocked	Email was blocked due to a receiver test	1
explicit_blocked	Email was blocked due to a explicit block (DENY ALL)	1
tagging_blocked	Email was blocked due to a tagging test	1
virus_blocked	Email was blocked due to a virus test	1
type_blocked	Email was blocked due to a type test	1
name_blocked	Email was blocked due to a filename test	1
text_blocked	Email was blocked due to a file contents/text test	1
anynamed_blocked	Email was blocked due to attachments in the email	1
sized_blocked	Email was blocked due to email size limit exceeded	1
external_blocked	Email was blocked due to an external test returning non-zero	1
zipbomb_blocked	Email was blocked due to zip decoding size exceeding limits	1
mailpack_size	Email mailpack size in Kilobytes (1024bytes)	224
cpu_usage	CPU seconds required to process the email	12
	<i>Attachment file data (multiple instances)</i>	
filename	Name of the file. No leading path	eicar.com
filesize	Size of the file in bytes	49
hit_status	If an ACLitem blocked the file, what the status is	Virus Detected
comment	Comment from the ACLitem hit.	Deny all Viruses

Table 9.2: Global block status codes

Global Block Code	Associated test (ACLitem test)
100	File type test
101	File name test
102	File contents / text test
103	File size test
104	Attachment presence test
105	External test
106	Email sender test
107	Email receiver test
108	Virus test
109	Tagging test
110	Source IP test
199	Zipbomb test

10 Debugging

10.1 Introduction

Things do occasionally go wrong on all computer systems. It is up to the debugging tools to attempt to locate and resolve the problem as quickly and simply as possible. Xamime offers two separate types of debugging, a global debugging system activated from XMAdmin (can very quickly fill up system log files), or, the preferred method of invoking Xamime directly from the command line

10.2 Global Debugging

10.2.1 Activating

Xamime provides debugging output to syslogs when invoked. The debugging detail is useful for tracing down where problems might be occurring, both at a binary level and ACL[item] level. Activating the debugging requires changing the debugging switch in the XMAdmin main configuration panel to “yes” (See 2.2 on page 5)

10.2.2 Output

Below is an example of Xamime/XMAdmin syslog debug output.

```
XAMIME:main: Regenerated filetype list...
XAMIME:main: Reading ACL #1
XAMIME:main: Comparing arbx == <worker@hserver.pldaniels.com or arbx>
XAMIME:main: Receiver hit!
XAMIME:XAMIME_process_acl: Start
XAMIME:XAMIME_process_acl: Copying Admin address
XAMIME:XAMIME_process_acl: Status = a
XAMIME:XAMIME_process_acl: Processing test...(Virues be blocked)
XAMIME_process_acl: item is DENY
XAMIME:XAMIME_aclitem_test: Start (v)
XAMIME:XAMIME_aclitem_test: Virus test
XAMIME:XAMIME_aclitem_test: Testing for Viruses...
XAMIME:XAMIME_aclitem_test: Result = 0
XAMIME:XAMIME_process_acl: end of test.
XAMIME:XAMIME_process_acl: Status = a
XAMIME:XAMIME_process_acl: Processing test...(no text files.)
```

10.3 Command Line Debugging

Xamime can be invoked from the command line for the purpose of debugging. The command line debugging mode is the preferred method of locating faults with Xamime. Xamime supports several debugging modes as listed in following table.

Debugging Flag	Description
<code>-debug-acls</code>	Show the ACL testing path of the email. This mode is useful for debugging which ACLs and ACLitems are being invoked.
<code>-debug-delivery</code>	Show details of the final delivery process (if the email is passed for final delivery)
<code>-debug-fine</code>	Show a high level of debugging details, including build up, mailpack decomposition, ACL processing and tear down processes. The output from this debugging mode can be very large, in the order of several hundred Kbytes.

All debugging modes generate their output to STDOUT. A typical debugging execution is setup as follows:

```
./xamime xamime.cfg sender@somewhere.com receiver@domain.com  
--debug-fine < mailpack > debug.log
```

Figure 10.1: Example `-debug-acls` output

```

HServer:root:/extra/development/xamine-33/validator# cat mailpacks/test.multijpeg | ../xamine /usr/local/xamine/xamine.cfg root@pldnet.com arb@pldnet.com -d 2002
Debug mode 2002 active [ACL trace mode]

[324234foo]From: root@pldnet.com, To: arb@pldnet.com, Conf: /usr/local/xamine/xamine.cfg
      5276text10 size=      1 type=unknown
      Teamwork1.jpg size=    30731 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      ironman.jpg size=    27340 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      5276text8 size=      1 type=unknown
      fun6.jpg size=    32441 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      fun5.jpg size=    26643 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      5276text7 size=      1 type=unknown
      fun4.jpg size=    36869 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      fun3.jpg size=    25684 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      5276text5 size=      1 type=unknown
      fun2.jpg size=    22358 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      fun1.jpg size=    97708 type=IMAGE:COMPRESSED:JPG:JFIF Jpeg:1
      5276text2 size=      722 type=DOCUMENT:EMAIL:HTML:HTML:84
      5276text1 size=     179 type=unknown
      5276text0 size=      46 type=unknown
      5276headers size=   4098 type=unknown

Commencing ACL processing [Total ACLs = 3]
ACLName: "pldaniels@pldaniels.com" [1075007024;10129012438501;-1:1][sender = root@pldnet.com, receiver = arb@pldnet.com]
ACLName: "." [1075007024;10128090356219;0:1][sender = root@pldnet.com, receiver = arb@pldnet.com]
  |--Processing using sender match
  | |--RAF mode = 0
  | | |--Admin Address = postmaster
  | | | |--ACLitem [000]: Mode='d' Test='n' Comment='Block Party Virus'
  | | | | Test Result = 0
  | | | |--ACLitem [001]: Mode='d' Test='x' Comment='KAK Virus'
  | | | | Test Result = 0
  | | | |--ACLitem [002]: Mode='d' Test='v' Comment='Block Virus Content'
  | | | | Test Result = 0
  | |--Processing using receiver match
  | | |--RAF mode = 0
  | | | |--Admin Address = postmaster
  | | | | |--ACLitem [000]: Mode='d' Test='n' Comment='Block Party Virus'
  | | | | | Test Result = 0
  | | | | |--ACLitem [001]: Mode='d' Test='x' Comment='KAK Virus'
  | | | | | Test Result = 0
  | | | | |--ACLitem [002]: Mode='d' Test='v' Comment='Block Virus Content'
  | | | | | Test Result = 0
  | | | | | done.

ACLName: "#staff" [1075007024;10128855764744;0:1][sender = root@pldnet.com, receiver = arb@pldnet.com]
  |--Processing using receiver-group match
  | |--RAF mode = 0
  | | |--Admin Address = postmaster
  | | | |--ACLitem [000]: Mode='d' Test='r' Comment='No Explicit Emails'
  | | | | Test Result = 0
  | | | |--ACLitem [001]: Mode='d' Test='n' Comment='Deny foofoobar.bat'
  | | | | Test Result = 0
  | | | |--ACLitem [002]: Mode='t' Test='t' Comment='Allow Documents'
  | | | | Test Result = 0
  | | | |--ACLitem [003]: Mode='t' Test='t' Comment='Allow Compressed Images'
  | | | | Test Result = 100
  | | | |--ACLitem [004]: Mode='p' Test='g' Comment='Allow Tagged'
  | | | | Test Result = 109
  | | | | A Test HIT has occurred, Breaking out of ACL processing.

SUMMARY: Email is -EXPLICIT-PASSED-. Pass test code = 109

(FAKE)>>>Deliver from root@pldnet.com to arb@pldnet.com
HServer:root:/extra/development/xamine-33/validator# █

```

11 Error Messages

11.1 Introduction

When problems occur during the Xamime filtering process, errors will typically be sent to the system log file (`/var/log/mail.log;messages;maillog`). Xamime tries to provide as much information as possible in the error to help the administrator rectify the issue causing the problem(s). If it is unclear as to the cause of the errors, it's recommended that the administrator contact PLD Software directly.

11.2 Things to check

11.2.1 Disk utilization

disk space

Check that you have sufficient disk space available for your spool directories and your Xamime retained email. Without sufficient free space email cannot be saved and often logging can not be stored (often causing a lot of confusion as no errors are thus logged). Free space can be checked using:

```
df -h
```

anything less than 10% free space is considered dangerous on most systems.

inodes

Often overlooked, the disk inode availability determines if a file can be created/allocated on the filesystem. Even if `df -h` reports sufficient free space inodes can often be exhausted. The most common cause of the depletion of inodes is small files (less than 4Kbytes in size). The inode availability status can be checked using:

```
df -i
```

directories storage limit (Xamime Levels 1, 2 and 3)

Most file systems can store many hundreds of thousands of files per directory but have a limited ability to store directories. Each file system has its own limit, though a typical value is approximately 32,000 entries. If email is being rejected by Xamime and retained in the in-bound email spool, check the number of directories in the *tmp* directory:

```
ls /usr/local/xamime/tmp | wc -l
```

If this value is at 32,000 (or 64,000 for ReiserFS), you will need to either;

- Remove old retained email
- Create a new *tmp* directory and move the original to a new location (*tmp.old*)
- Upgrade to Xamime Level 4 (the Spanning storage system avoids the filesystem limitations)

11.2.2 Mail processing

Sendmail out-bound queue increase

Because Xamime uses two separate sendmail configuration files (*xamime.cf* and */etc/mail/sendmail.cf*), it is possible for situations to arise where email which has been processed by Xamime refuses to be sent. Symptoms include:

- Build up of *mailq* entries (email waiting to be delivered to its next hop)
- Running *sendmail -q -v* returns after a delay showing no, or limited numbers of email delivered

The cause of this condition is because both the Xamime cf and original sendmail cf have the same QueueLA and RefuseLA values. The solution to this problem is to either reduce the xamime.cf QueueLA and RefuseLA values, or increase the sendmail.cf QueueLA and RefuseLA values. It is preferable to increase the sendmail.cf values as this will not require an entire sendmail restart (the effect will be immediate).

11.2.3 System CPU and memory resources

Surging system load

Symptoms include:

- Load goes through spike cycles
- Sendmail periodically refusing to accept new connections
- Number of messages per minute/hour are lower than anticipated

A common cause of this condition is when sendmail does not have its in-bound connection rate throttled. Check the *xamime.cf* file and activate the *ConnectionRateThrottle=* parameter. A typical value of 1 or 2 is usually sufficient (relating to 3600 or 7200 email per hour respectively).

If throttling does not appear to ease the condition sufficiently, check to see if there are other SMTP servers feeding the Xamime server which might be using *connection caching*. Connection caching is a facility provided by sendmail servers to allow multiple email to be sent over the same connection in a sequential manner. Connection caching effectively bypasses the *ConnectionRateThrottle=* setting.

Memory starvation

Symptoms include:

- Applications refusing to start
- High disk I/O
- Server generally behaving strangely
- Spam-filtering not working correctly

The use of SpamAssassin with Xamime can often lead to memory starvation. Each SpamAssassin process can consume well over 50MB of ram in order to handle all its rules. Possible solutions include:

- Reduce the number of email being processed concurrently (see about throttling at 11.2.3 on the preceding page)
- Install more memory
- Deactivate any unwanted services (most Unix distributions come with many services running that are not required)

12 Additional Tools

12.1 confcompile

confcompile is used to convert a text-only based Xamime configuration file into its [usable] binary version. Normally *confcompile* is invoked during the Xamime installation process and hence is rarely used again. Typical use of *confcompile* may be if you lose or corrupt your existing *xamime.cfg* file.

Usage of *confcompile* is as follows:

```
confcompile <input configuration (text)> <output file>
```

Where *input configuration* is the *xamime.conf* file and *output file* is the *xamime.cfg* file. The output of an example run is presented below:

```
./xamime-release# ./confcompile xamime.conf xamime.cfg
Reading...done.
Writing...done.
Verifying...Verified.
./xamime-release#
```

12.2 buildxamcf

buildxamcf is a tool used to create the *xamime.cf* file from an existing *sendmail.cf*. Essentially the tool adds into an existing *sendmail.cf* a ruleset 0 hook and a new delivery agent definition. This program is included only in the Xamime installation package.

Usage of *buildxamcf* is as follows:

```
./buildxamcf <sendmail.cf> <xamime.cf> <xamime home path>
```

An example run of *buildxamcf* is shown:

```
./xamime-release# ./buildxamcf /etc/mail/sendmail.cf ./xamime.cf /usr/local/xamime
Creating MDA file...done.
Applying updates to current /etc/mail/sendmail.cf
to create ./xamime.cf...done.
./xamime-release#
```

12.3 xmapasswd

xmapasswd is a command line tool used to setup accounts for XMAdmin. Although this is typically done via the XMAdmin interface, there may be times where manual intervention is required.

Usage of *xmapasswd* is as follows:

```
./xmapasswd -u <username> -p <password> -f <passwd file>
```

-u : Username

-p : Password to be used for the username

-f : location of the password file (typically xadmin.passwd)

Index

access, 15
aclitempage.dhtml, 11
acls, 11
AntiVirus Filtering, 20

Binary detection, 8
buildtypes, 10, 11
buildxamcf, 31

cf, 14
conf, 14, 15
confcompile, 12, 31
ConnectionRateThrottle=, 29
css, 13

Debugging, 25
Decimal, 9

Extentions, 16

filetype, 8
filetypes, 10

genericav.conf, 12
groups, 11

help, 12
Hexadecimal, 9

inodes, 28

licence, 14
list, 10
localdomains, 12
logs, 12

mailadmin, 12, 19
mailq, 29
mailreceiver, 12, 19
mailsender, 12, 19
mainmenu-sidebar.url, 12
mainmenu.extensions, 12
Menu, 16
Mxamime.MDA, 12

offset, 9

passwd, 15
PHP -disable-cli, 22

QueueLA, 29

RefuseLA, 29
Regex, 6
ReiserFS, 28

Sidebar, 16
spec, 10

Text detection, 8

xamime.cf, 29
xamime.cfg, 14
xmapasswd, 15, 32