
Mailgate Ltd.

SpamWeasel Free User Manual



Microsoft is a registered trademark and Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corporation.

Copyright © 2001 Mailgate Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in and for or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Mailgate Ltd.

Edited by Lani K. and David D. Thompson.

Contents

GETTING STARTED.....	1
WELCOME TO SPAMWEASEL.....	1
SPAMWEASEL INTRODUCTION	2
HOW SPAMWEASEL WORKS	3
INSTALLING SPAMWEASEL.....	4
SYSTEM REQUIREMENTS	4
SOFTWARE INSTALLATION	5
SPAMWEASEL CONFIGURATION.....	6
E-MAIL CLIENT CONFIGURATION	7
USING SPAMWEASEL	9
USING SPAMWEASEL	9
HOW SPAM IS FILTERED	10
THE CONFIGURATION SCREENS.....	11
<i>Action Tab</i>	11
<i>Rules Tab</i>	14
<i>Patterns Tab</i>	15
<i>Patterns List Dialog</i>	16
<i>Words Tab</i>	17
<i>Words List Dialog</i>	17
ABOUT THE FILTER RULES	19
ABOUT RULES	19
ABOUT PATTERNS	21
ABOUT WORDS	22
SOLVING PROBLEMS	23
USING THE LOG FILE	23
UNDERSTANDING PORTS	27
PORT <i>IN USE</i> PROBLEMS	28
REGISTRATION & SUPPORT	29
REGISTERING SPAMWEASEL	29
GETTING SUPPORT	30
THE MESSAGE VIEWER.....	31
INTRODUCTION TO THE MESSAGE VIEWER.....	31
USING THE MESSAGE VIEWER.....	32
<i>The Menu Bar</i>	32
<i>Message File Source Panel</i>	33
<i>SpamWeasel Pending Queue</i>	33
<i>SpamWeasel Archives</i>	33
<i>Message Listing Panel</i>	34
<i>Viewing a Message</i>	34
<i>Copying a Message</i>	34
<i>Moving a Message</i>	35
<i>Deleting a Message</i>	35
TECHNICAL REFERENCE	36
USING WILDCARDS	36
<i>Using Wildcard Expressions</i>	36
USING MACROS.....	37

<i>Using Macro Expressions</i>	37
USING SCRIPTING	38
<i>Introduction to Scripting</i>	38
<i>Scripting Syntax & Commands</i>	38
<i>Scripting Functions</i>	44
WINDOWS REGISTRY	64
WINDOWS REGISTRY	64
REGISTRY - SPAMWEASEL	65
REGISTRY - PRIORITY	66

Figures

FIGURE 1 - HOW SPAMWEASEL WORKS	3
FIGURE 2 - ACTION TAB.....	11
FIGURE 3 - RULES TAB.....	14
FIGURE 4 - PATTERNS TAB.....	15
FIGURE 5 - EXAMPLE PATTERN DIALOG	16
FIGURE 6 - WORDS TAB	17
FIGURE 7 - EXAMPLE WORDS LIST DIALOG.....	17
FIGURE 8 - MESSAGE VIEWER.....	32

Getting Started

Welcome to SpamWeasel

Getting Started with SpamWeasel

The Welcome to SpamWeasel help page will be displayed every time you start the program until you save your settings. After saving your configuration, you will still be able to access this information through the normal help system.

Starting and Stopping SpamWeasel



SpamWeasel will normally start automatically when you start your computer. When the program is running you will see the SpamWeasel icon in your system tray (normally located in the bottom right hand corner of your screen).

To start the SpamWeasel, double click on the Desktop icon or select SpamWeasel in your Start Menu.

To stop the program, right click on the System Tray icon and select Exit.

Configuring SpamWeasel

Before you can use SpamWeasel you need to configure your computer. There are two steps to this, configuring SpamWeasel and changing your E-mail client to work with SpamWeasel.

To configure SpamWeasel, you should review the default settings and then save the configuration by clicking OK in settings dialog. You can access the configuration dialog at any time by double clicking on the Desktop or System tray icons. For more detail on this see

SpamWeasel Configuration on page 6.

For SpamWeasel to work, you need to also change the POP collection (incoming mail) settings in your E-mail client software. For more details see the page **E-mail Client Configuration** on page 7. You may also like to visit the SpamWeasel area of our web site www.mailgate.com where you will find details on how to configure the more common E-mail client programs.

SpamWeasel Introduction

What Is Spam Mail

In the true sense there are two types of Spam - *Unsolicited Commercial Email* (UCE) and *Unsolicited Bulk Email* (UBE). This involves the spammer obtaining your email address and using it to send you mails you have not asked for, often by using means to hide the origins of the message.

For many people though Spam means "*any email I did not ask for or do not want*". Some of this mail could be regarded as *solicited* because often it is caused by completing on line registrations or by signing up to list servers. Sometimes getting your address de-listed again can be difficult so you continue to receive un-wanted email.

What is SPAMWeasel

Mailgate's SpamWeasel is a simple program that will check all your E-mail as it is collected from your Internet Service Provider (ISP) against a set of rules to try to filter out Spam E-mails.

The filter is designed to identify true Spam by looking for certain characteristics of this type of mail. It can also be adjusted to block other un-wanted mail.

Some rules are simple checks on the structure or content of the mail, others make use of pattern and word data lists to support the rule. These lists are referenced by individual rules and are fully user maintainable.

Once a mail has been identified as potential Spam, the software can be configured to perform user configurable actions. The actions taken can be linked to a certainty or priority associated with each rule. This allows an easily identified Spam mail to be treated differently than mail that may or may not be Spam.

Refer to **How Spam is Filtered** on page 10 for full details of the filtering process.

How SpamWeasel Works

In a basic Internet mail set-up you will use an E-mail client program to collect, read, create and send your mails.

This program will generally use the POP3 protocol to collect your mail from your Internet Service Provider (ISP) and the SMTP protocol to send your outgoing mail.

SpamWeasel is a POP3 proxy program that sits between your E-mail client and your ISP. When you check for incoming mail, your mail software passes this request to SpamWeasel, not directly to your ISP. SpamWeasel then passes the request to your ISP.

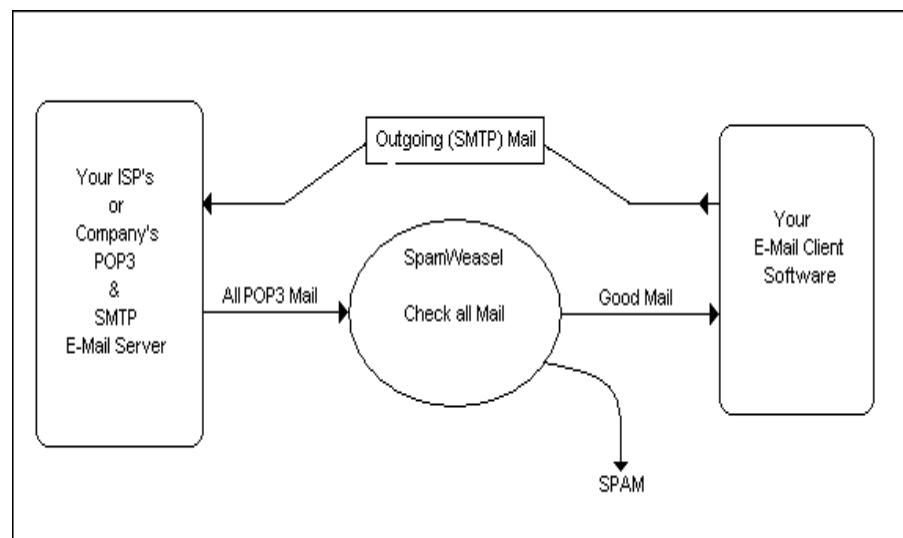


Figure 1 - How SpamWeasel Works

If there is any mail to be collected, it will be passed to SpamWeasel which then checks the mail data against all its rules. Mail that is then identified as Spam will be processed according to the actions configured in the product. All other mail is simply passed on to your E-mail client as normal.

See **E-mail Client Configuration** on page 7 to see what changes you need to make in your E-mail client software to get it to talk to SpamWeasel. The section **Using SpamWeasel** (page 9) has details on **How Spam is Filtered** (page 10).

For more technical information on how the system works, refer to **Understanding Ports** on page 27.

Installing SpamWeasel

System Requirements

System Requirements

SpamWeasel will run on Windows 95/98/ME, NT4 2000 and XP.

A minimum of 32Mb of memory is recommended.

Disk space requirements are dependant on the number of mail messages and the user settings for archiving and purging periods.

Software Requirements

To use SpamWeasel you will need the following:

- i. A TCP/IP mail account with an Internet Service Provider (ISP).
- ii. An E-mail client program.
- iii. A means of connecting to the Internet.

Before installing SpamWeasel you should complete and test the configuration of your E-mail client and ISP account. You will need a note of the POP account settings used.

Software Installation

Before you install SpamWeasel, make sure you have a working E-mail setup.

To install SpamWeasel, run the self-extracting executable set-up program. This program, after you have read and confirmed your acceptance of the software licence, will install all the necessary components onto your hard disk.

Once the install process has completed you must:

- i. Configure SpamWeasel.
- ii. Change your E-mail client settings to work with SpamWeasel.

SpamWeasel Configuration

Starting and Stopping SpamWeasel



SpamWeasel will normally start automatically when you start your computer. When the program is running you will see the SpamWeasel icon in your system tray (normally located in the bottom right hand corner of your screen).

If SpamWeasel is not running, to start it either double click on the Desktop icon or select SpamWeasel in your Start Menu. Whenever you start SpamWeasel the registration dialog will be displayed until you register the product.

To stop the program, right click on the System Tray icon and select Exit.

Configuring SpamWeasel (Initial Settings)

To initially configure SpamWeasel:

1. If SpamWeasel is not running, double click on the Desktop icon or use your Start menu to start it.
2. Double click on the Desktop or System tray icon. This will open the Configuration Screens.
3. Click on the **Action Tab** (page 11) and review the Global Settings for the Spam Report and Archive period. If you have other POP mail software which acts as a server/proxy (e.g. some Antivirus programs), you may need to adjust the Listen on port - See **Understanding Ports** on page 27 for more detail.
4. Click on the **Rules Tab** and review the rules, disabling any that are not required. In particular consider the two rules that check the To: addresses against 'My Addresses'. Only one of these should be active.
5. Click on the **Patterns Tab** (page 15) and review any lists that should contain localised entries, for example - Friendly E-mail Addresses.
6. Review any other settings as you wish, then click OK to save your settings.
7. Right click on the System Tray icon and select Exit to close the program, then re-start SpamWeasel as above.

You should now configure your E-mail Client (page 7) to work with SpamWeasel if you have not already done this.

E-mail Client Configuration

Configuring your E-mail Client.

In order for your E-mail client will work with SpamWeasel, you will need to change the settings for any POP collections (Incoming mail) you wish to be passed through the filter. Do not make any changes to your SMTP (Outgoing mail) settings.

To configure your client to work through SpamWeasel you normally only need to make two changes:

1. POP3 Server - Note your current setting then change this to 127.0.0.1
2. POP3 Account Name - Change this by adding an @ and then your ISP's POP server address to the end of the current setting. You can also specify a non-standard port by further adding *:portnumber* to the end of this setting.

Notes

1. Password - Leave this setting.
2. You may need to re-start your E-mail client for the changes to take effect.
3. If you have multiple accounts configured in your E-mail client, you may need to repeat these changes for each account you wish to be passed through SpamWeasel.
4. If you collect quite large numbers of emails when you connect to your ISP you may need to increase the timeout setting in your client software.
5. If your ISP requires Secure Password Authentication for incoming (POP3) mail, you must disable this and add authentication control to the account name in 2. above. There are currently two options \$APOP if your ISP uses APOP authentication and \$MSN if you are using an MSN account which requires authentication. For example, for MSN your account name setting may look like `jdoe@pop3.email.msn.com$MSN`
6. Some E-mail clients (some Netscape versions in particular) do not allow using '@' as the separator between the POP Account name and the ISP's server address. You can configure SpamWeasel to use a different character (we recommend '?') in place of the @ by changing the Separator Char setting on the Actions Tab.
7. For advanced users - you may use any name in the POP3 server address, for example *localhost*, provided your TCP/IP configuration will correctly resolve this to 127.0.0.1

Different E-mail clients can use different names for the above settings. If you are not sure how to access these settings, please refer to your client software help or manual. You can also visit the SpamWeasel FAQ area on our web site www.mailgate.com where you will find details for configuring the more common E-mail packages and information on any special requirements that may be required.

Example settings

Your Current Settings:

Incoming POP3 Server - pop.myisp.net

Account Name - jdoe

Password - pass

Change to:

Incoming POP3 Server - 127.0.0.1

Account Name - jdoe@pop.myisp.net

Password - pass

Using SpamWeasel

Using SpamWeasel

The SpamWeasel is installed with a full set of pre-defined rules. These are listed in the **Rules Tab** (page 14) and you should review which rules you wish to use and adjust their priority flags as required.

Some rules reference data lists. These lists are provided with sample data and you should edit them according to your requirements through the **Patterns Tab** (page 15) and the **Words Tab** (page 17) .

When the Filter identifies a mail as Spam, the mail is given the priority level of the rule used and the SpamWeasel will perform the actions defined in the **Action Tab** (page 11) according to this priority level. Use this tab to define the required actions for each priority level in use.

We recommend you read **How Spam is Filtered** on page 10 before finalising your configuration. This will assist in ensuring you get the best from the software.

How Spam is Filtered

The SpamWeasel checks every mail that passes through the system against a set of rules. This is a matching process where the mail is checked against a set of criteria defined by the rules. The rules may be viewed and adjusted in the **Rules tab** (page 14). For details on how the rules work see **About Rules** on page 19.

Some rules need to use variable data to achieve the desired results. Such rules reference data lists which can be reviewed and edited in the **Patterns** (page 15) and **Words** tabs (page 17). For full details on these lists see **About Patterns** on page 21 and **About Words** on page 22.

There are three possible results after each rule is processed:-

1. The mail is matched by the rule and is identified as spam.
2. The mail is matched by the rule and is identified as not spam.
3. The mail is not matched by the rule.

Should a match be made, then this is noted along with the matching rule's priority level. Rule processing continues until either all the rules have been tried or the highest achievable priority level has been reached.

Mail messages that fail to be matched by any rules are passed on by the SpamWeasel for normal processing.

For those mail messages that have been matched, the status of the highest priority level associated with the mail is checked. If this status indicates the mail is **not** spam then it is also passed on for normal processing. If the status indicates that it **is** spam, then the SpamWeasel will apply the actions for the mail's priority level as defined on the **Action Tab** (page 11).




When you first use the SpamWeasel, you should take care with the actions you define. It is advisable to run the system for a while carefully checking you are correctly identifying your spam mails to confirm your settings before using actions to block mail deliveries.

If a mail matches one of your rules SpamWeasel adds a field to the message header called X-SpamWeasel. This field will show the name of the operative rule used. So if you are not sure why a mail has been marked as Spam, check the message header details.

The Configuration Screens



For more information about any entry on the configuration screens, select the  button (top right) then click on the entry you want to know about.

Action Tab

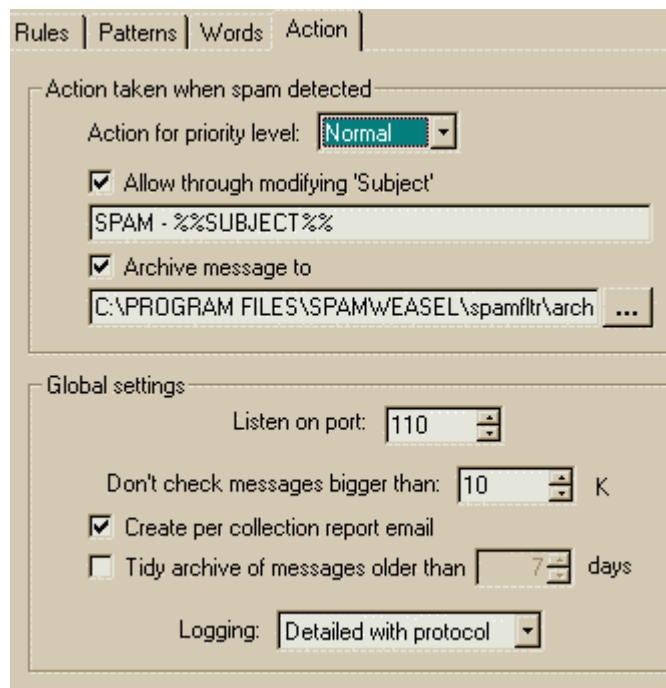


Figure 2 - Action Tab

The Action Tab is used to define the actions to be taken by the SpamWeasel when a mail has been identified as Spam. The tab is divided into two sections. The upper section is used to define what happens to a Spam mail. The lower section sets the general operation of the software.

► Action taken when spam is detected

When a mail is checked by the SpamWeasel and is matched by one of the rules, the rule's priority level is associated with the message. (See **How Spam is Filtered** on page 10). In this part of the Action Tab you can set what actions you wish to be performed for each of the priority levels in use.

To set these actions, first select the priority level you wish to configure using the **Action for priority level** dropdown. You will now see displayed the current actions that will be performed on any Spam mail that has been given this priority level and you can make any changes required.

For each priority level you can:-

1. Have the Spam Filter pass the mail through to its recipient(s) after modifying the subject line, by checking the **Allow through modifying 'Subject'** option. The **subject** can be modified using the Mailgate Macro language (**see Using Macro Expressions** on page 37) facility. Uncheck this option if you wish SpamWeasel to drop Spam mail messages matched by rules with the current priority level.
2. Have the SpamWeasel archive a copy of the message using the **Archive message to** option. The message copy will be added to a daily archive file in the specified folder. The file name format used is **SF<yymmdd>.TXT** where <yymmdd> is the current date.



You should review the actions for all of the priority levels you have associated with the rules in the **Rules Tab** (see page 14).



If you set the actions for a priority level to NOT pass the mail through by unchecking Allow through modifying 'Subject' and check Archive message to, then you will not receive copies of mails found to be Spam. These messages will however be written to the daily archive file. You can review the contents of the archive files and retrieve messages in them by using the **Message Viewer** (see page 31) utility included with SpamWeasel.

► Global settings

The Global settings section is used to set the general operation of the software.

- i. With the **Listen on port** setting you can change the TCP/IP port used by the SpamWeasel from the default of 110. This may be necessary if you have other POP3 related software on your machine. If you need to change this setting, you will also need to change your E-mail client.
- ii. Some E-mail clients (some Netscape versions in particular) do not allow using '@' as the separator between the POP Account name and the ISP's server address (see Client Configuration). You can configure SpamWeasel to use a different character (we recommend '?') in place of the '@' by changing the **Separator Char** setting. If you change this setting you must ensure you change your E-mail client settings to match.
- iii. Most Spam messages are small in size. With the **Don't check messages bigger than** setting you can have large messages bypass the rules checking process to preserve performance.
- iv. By selecting the **Create per collection report email** you can have the SpamWeasel send a report of Spam message activity after each collection.
- v. The **Tidy archive messages older than** option is used to make SpamWeasel remove old archive files automatically. If not checked, old archive files will be kept indefinitely.
- vi. By selecting the **Create daily log email** option you can have the SpamWeasel send an email copy of the previous day's log file when you first collect mail the following day.

- vii. The **Tidy log files older than** option is used to make SpamWeasel remove old log files automatically. If not checked, old log files will be kept indefinitely.
- viii. Use the **Logging** option to set the level of detail written to the SpamWeasel log file. See **Using the Log File** on page 23 for more details.



In many cases the identification of Spam mail is not an exact science. You should consider making settings that allow you to monitor and review the identification process when you first setup the module and when your rules and list settings are tested you can set your actions with confidence.

Rules Tab

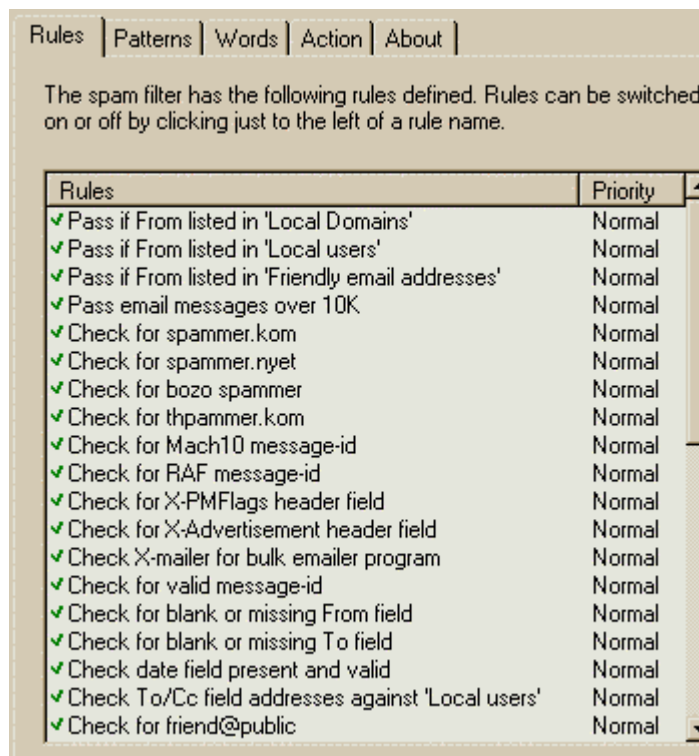


Figure 3 - Rules Tab

The Rules Tab displays a list of the rules used by the SpamWeasel when checking the mail traffic. Use the scroll bar to move through the list of rules.

In the Rules List you can :-

▶ Enable/Disable a rule

Click to the left of the rule. Enabled rules are marked with a green check and disabled rules with a red cross.

▶ View the rule details and code

Double click on any rule and a pop-up window will display the details and code for the selected rule. See **About Rules** on page 19 for more details on the rule language and how you can maintain your own rules.

▶ Change the rule priority

Right click on the current rule priority and a pop-up menu will be displayed. Select the required priority. For more information on the use of priorities see **How Spam is Filtered** on page 10.



Remember to configure actions in the Action Tab (see page 11) for all the priorities you are using.

Patterns Tab

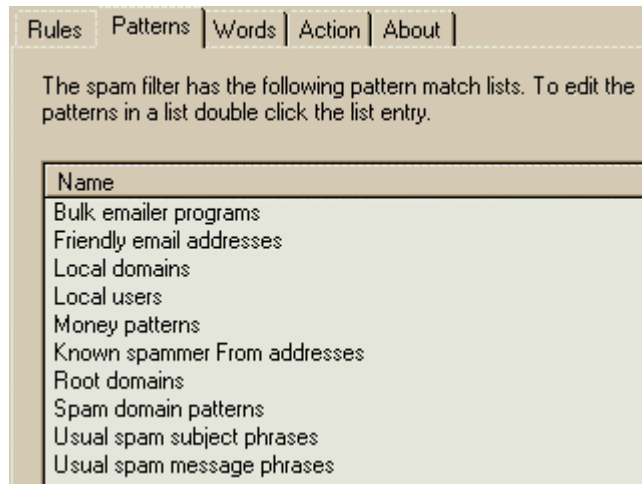


Figure 4 - Patterns Tab

The Patterns Tab displays a list of the installed pattern lists used by individual rules.

The Patterns list box shows the name given to each list, and these names are used by the rules to reference the appropriate list.

Double click on an entry to display the **Patterns List Dialog** (see page 16) details. In this dialog the list contents may be viewed or changed.



Each Pattern list is stored as a file in the SpamWeasel system folder. The names displayed in this tab are the file names used. See **About Patterns** on page 21 for more details.

Patterns List Dialog

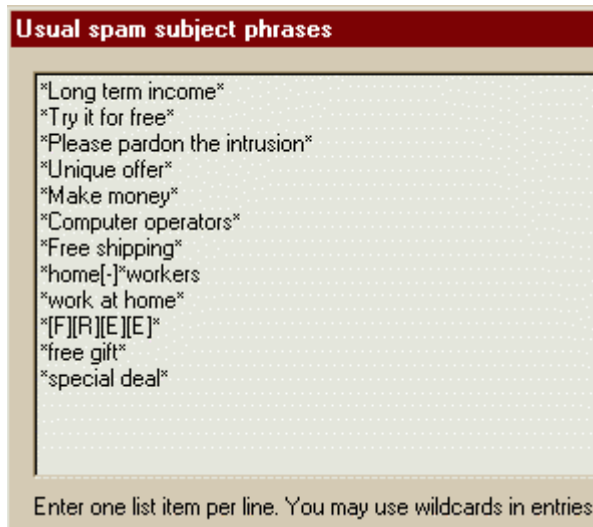


Figure 5 - Example Pattern Dialog

The Patterns List Dialog displays the contents of a Patterns List. To access the dialog, double click on a list name in the **Patterns Tab** (see page 15).

Patterns may be added to the list in any order according to the following rules :-

- i. Each pattern entry must be placed on a new line.
- ii. The use of wildcards is allowed.
- iii. Patterns are not case sensitive.

When adding patterns to a list, remember to check which rules use the list and which parts of the mail are being checked for a pattern match. This context may affect the entries you wish to make.

When you have finished editing the list, click OK to save your changes.

Words Tab

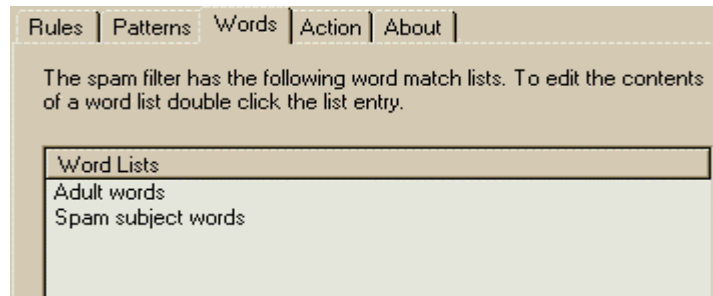


Figure 6 - Words Tab

The Words Tab displays a list of the installed words lists used by individual rules.

The Word Lists box shows the name given to each list, and these names are used by the rules to reference the appropriate list.

Double click on an entry to display the **Words List Dialog** details (below). In this dialog the list contents may be viewed or changed.



Each Words list is stored as a file in the SpamWeasel system folder. The names displayed in this tab are the file names used. See **About Words** on page 22 for more details.

Words List Dialog



Figure 7 - Example Words List Dialog

The Words List Dialog displays the contents of a Words List. To access the dialog, double click on a list name in the **Words Tab** (above).

Words may be added to the list in any order according to the following rules :-

- i. Each word entry must be separated by spaces.
- ii. Any number of words may be placed on a line.
- iii. The use of wildcards is not permitted.
- iv. Word matching is not case sensitive.

When adding words to a list, remember to check which rules use the list and which parts of the mail are being checked for these words. This context may affect the entries you wish to make.

When you have finished editing the list, click OK to save your changes.

About the Filter Rules

About Rules

The SpamWeasel is configured with a standard set of rules. When a mail is passed through the extension each rule is processed in turn until the mail is either matched by a rule or all the rules have been tried. See **How Spam is Filtered** on page 10 for more details.

The rules set can be viewed in the **Rules Tab** (see page 14) and by double clicking on any rule the associated code will be displayed. The rules use an internal scripting language, details of which can be found in the Technical Reference section under **Scripting** on page 38.

The rules are contained in three simple text files, userpre.sfr, system.sfr and userpost.sfr which are located in the spamfltr folder under the SpamWeasel system folder (by default c:\Program Files\SpamWeasel). When the rules are loaded by the filter the files are read in the above order.

The software is shipped with a standard set of rules contained in system.sfr which should not be changed. If you wish to create your own rules these should be placed in either userpre.sfr or userpost.sfr. To add a new rule you may edit these files with a suitable text editor.

The SpamWeasel functions are defined using the script language external function directive. The definitions are contained in the file spamfltr.inc. This file is read when the rules are loaded and if you wish to create your own functions, these may be included in this file. Otherwise you should not change this file.



Always take a backup copy of your existing files before making any changes. Only edit the files if you are sure you know what you are doing! If you make changes to either of the system files, these must be backed up before re-installing or upgrading the software. Your changes may need to be re-done after installing.

Rule Location - You should consider carefully the location of your rule. To minimise system resource requirements, rules which are

more likely to find a match should be placed to the top of the rules list by using the file userpre.sfr. Rules which are rarely matched may be added to the end of the list by using userpost.sfr. Remember, rule processing stops if a high priority match is found.

Rule Structure - Each rule starts with the statement - #rule "Rule Name" - where "Rule Name" is the name displayed in the Rules Tab.

Generally this will be followed by a number of comment lines starting with - //. These are used to describe your rule.

Next comes the rule code itself using the scripting language.

Finally the rule is terminated with the statement - #endrule

The line // ----- is simply included to make reading the file easier.

Example

```
#rule "My Example Rule"
```

```
//
```

```
// This rule will reject any email with 'Spam' in the subject.
```

```
//
```

```
field$ = HeaderFieldValue("Subject")
```

```
if WildcardMatch(field$,"Spam") then
```

```
    IsSpam()
```

```
endif
```

```
#endrule
```

```
//-----
```

In this example the content of the mail header field 'Subject' is obtained and then it is checked for the word "Spam". If this is found then the mail is marked as being a Spam mail.

Note - When you have completed you changes to the rules file you must stop and re-start the SpamWeasel to register your changes.

About Patterns

SpamWeasel rules can make use of data lists which may be accessed using either the **Patterns** or the **Words Tabs**.

In the **Patterns Tab** (see page 15) you will find a list of all the Patterns lists installed.

Each patterns list is contained in a simple text file called <ListName>.lst which is located in the spamfltr folder under the SpamWeasel system folder (by default c:\Program Files\SpamWeasel). <ListName> is the name you see in the Patterns Tab and it is also used by the rules to identify the list.

To add a new list, create a text file and give it a name as above. You will need to close and re-open the configuration screen for your new list to appear in the Patterns Tab.

► List Contents

You can edit a list's contents either through the Patterns Tab or by using your text editor.

Patterns may be added to the list in any order according to the following rules :-

- i. Each Pattern must be placed on a new line.
- ii. The use of wildcards is allowed.
- iii. Patterns are not case sensitive.

About Words

SpamWeasel rules can make use of data lists which may be accessed using either the Patterns or the Words Tabs.

In the **Words Tab** (see page 17) you will find a list of all the Words lists installed.

Each words list is contained in a simple text file called <ListName>.wrđ which is located in the spamfltr folder under the SpamWeasel system folder (by default c:\Program Files\SpamWeasel). <ListName> is the name you see in the Words Tab and it is also used by the rules to identify the list.

To add a new list, create a text file and give it a name as above. You will need to close and re-open the configuration screen for your new list to appear in the Words Tab.

► List Contents

You can edit a list's contents either through the Words Tab or by using your text editor.

Words may be added to the list in any order according to the following rules :-

- i. Each word entry must be separated by spaces.
- ii. Any number of words may be placed on a line.
- iii. The use of wildcards is not permitted.
- iv. Word matching is not case sensitive.

Solving Problems

Using the Log File

SpamWeasel writes information about each mail collection to a daily log file.

The log files are kept in the Log folder under the SpamWeasel system folder (by default c:\Program Files\SpamWeasel). The file name format used is **SW<yymmdd>.LOG** where <yymmdd> is the current date. You can view the log file with any text file viewer, like Notepad.

On the Action Tab, you can adjust the level of detail written to the log file by changing the logging setting. When you first install the filter it is advisable to set this to log all details and then review the log file to help adjust your settings.

Each line in the log file has a number of columns which are used as follows:

Column 1 - Identifies the product SpamWeasel

Column 2 - The type of log line.

Column 3 - Time of the event.

Column 4 - Operating System Thread.

Column 5 - The log message.

Example - SPAMWEAS I 09:12:15 0xffff1090d <Logged Message>

In the following extracts the first four columns have been removed for clarity.

The following shows an example of the SpamWeasel starting and checking the installed rules.

```
Spam Weasel version 1.0.4 initializing
spamfltr: Extension version 1.0.23 starting
spamfltr: Parsing C:\PROGRAM
FILES\SPAMWEASEL\spamfltr\userpre.sfr
spamfltr: Parsing C:\PROGRAM
FILES\SPAMWEASEL\spamfltr\system.sfr
```

```
spamfltr: Parsing C:\PROGRAM
FILES\SPAMWEASEL\spamfltr\userpost.sfr
spamfltr: Parsing rule 'Pass if From listed in 'Local
Domains''
spamfltr: Initialised rule 'Pass if From listed in 'Local
Domains''
```

This next extract is a typical collection where no mail is waiting. The client has 'leave a copy of mail on the server' switched on. This is managed by the UIDL command.

Note also the direction of data flow is indicated as follows:

>c - Data sent to the E-mail client

<c - Data from the E-mail client

<s - Data from the ISP's server

>s - Data sent to the ISP's server.

```
>c +OK Spam Weasel
<c USER testing@pop3.isp.net
<s +OK pop server ready
>s USER testing
<s +OK testing welcome
>c +OK testing welcome
<c PASS pass
>s PASS pass
<s +OK mailbox is locked
>s LIST
<s +OK 2 4173
<s 1 2690
<s 2 1483
<s .
>s UIDL
<s +OK unique id list follows.
<s 1 234886982063044
<s 2 234898982064031
<s .
>c +OK mailbox is locked
<c STAT
>c +OK 2 4173
<c UIDL
```

```
>c +OK Unique id list follows
>c 1 234886982063044
>c 2 234898982064031
>c .
<c QUIT
>s QUIT
<s +OK mailbox updated
>c +OK mailbox updated
```

After the SpamWeasel has requested the waiting message list from the ISP's server it compares this to locally stored information and then requests any mail not already collected. In this extract, message 2 is new and the log shows this being collected then passed to the client.

```
>s UIDL
<s +OK unique id list follows.
<s 1 234886982063044
<s 2 234898982064031
<s .
>s RETR 2
<s +OK 1469 octets
Spooling message to file
>c +OK mailbox is locked
<c STAT
>c +OK 2 4145
<c UIDL
>c +OK Unique id list follows
>c 1 234886982063044
>c 2 234898982064031
>c .
<c LIST 2
>c +OK 2 1469
<c TOP 2 0
>c +OK top of message follows.
Spooling top of message
<c RETR 2
```

```
>c +OK 1469 octets
Spooling from file message to client
<c QUIT
>s QUIT
```

This final extract shows what is reported to the log when the SpamWeasel identifies a message as being spam.

```
>s RETR 2
<s +OK 994 octets
Spooling message to file
spamfltr: Spam! - Rule "Check Subject against 'Usual spam
subject phrases'" msg00617 remote@isp.server postmaster
Test
```

This information is also written to the message header using a field X-SpamWeasel

Understanding Ports

SpamWeasel communicates with your E-mail client program and your ISP using the TCP/IP protocol.

► About Ports

Applications which use TCP/IP to communicate use *ports* to establish a uniquely defined link between the application and the IP address. At any time only one application can be associated with a given port.

If the application is one that waits for a TCP/IP connection, then it will require full use of the port it listens to. When a remote system wishes to connect, it will need to provide both the IP address for the waiting machine and the port for the waiting application. To make things easy for regularly used services such as mail, FTP, etc there are a number of *well-known ports* which are normally used for these. (For example POP3 - 110, SMTP - 25 and FTP - 21)

In some systems there may be multiple applications installed which have similar purposes. In this case they may all default to using the same port. Only one application will be able to run because the others can not access the port once it is in use. In this case you will need to move these other applications to use different ports. This is generally not a problem provided both ends of a connection can be configured with the same port number.

► Ports and SpamWeasel

SpamWeasel is designed for use with mail collection using the POP3 protocol. The *well-known port* for POP3 is 110.

To communicate, SpamWeasel makes use of two port settings.

Firstly, SpamWeasel waits for a request from your E-mail client software. As this is a POP3 protocol process, the default setting for this port is 110. You can change this port on the **Action Tab** (see page 11), but you will also need to be able to change to the same setting in your E-mail client.

Secondly, SpamWeasel needs to connect to your ISP's POP server. You will need to specify the IP address and port for this server in the Account name setting that is passed to SpamWeasel by your E-mail client. See **E-mail Client Configuration** on page 7. The port number part of the setting is optional and SpamWeasel will default to connecting to port 110. Unless you know your ISP's POP service uses a different port, you can assume the default is OK.

If you have other applications on your machine which are part of your mail system and work like SpamWeasel, for example some Antivirus software, you will need to adjust the ports used so that each process has its own number.

SpamWeasel will indicate a port problem by reporting 'unable to bind' in the log file when the software first starts. See **Using the Log File** on page 23 for more information.

Port *In Use* Problems

When SpamWeasel starts, it tries to connect to the ***Listen on Port*** set on the **Actions Tab** on page 11 to wait for a connection from your mail client software.

If this port is not available for some reason, SpamWeasel displays an error message. You will need to resolve this problem in order for SpamWeasel to work correctly with your mail client. The error number displayed is that reported by the Winsock TCP/IP stack. The most common code is 10048, which means '*The address is already in use*'.

To learn more about IP Ports see **Understanding Ports** on page 27. There are two common reasons for port problems:

There is other software on your computer which is already using the desired port. Often the easiest solution to this is to set SpamWeasel and your mail client to communicate using a different port, for example 1110. To do this change the **Listen on Port** setting on the **Actions Tab** (see page 11) and the POP3 port setting in your mail client (often in advanced settings).

There is other software on your computer which is blocking the use of the desired port, for example a PC firewall program. You should refer to the documentation for such software to find out how to allow the use of your desired port.

Once you have made any changes you should stop and restart (see page 6) SpamWeasel to ensure the error has been cleared.

Registration & Support

Registering SpamWeasel

When first started and every so often while in use, SpamWeasel will prompt you to register your copy. Registration is free and any information you give us will only be used for registration purposes.

To register SpamWeasel you need to visit our web site at www.mailgate.com and follow the links to the SpamWeasel Registration page. You can do this by using the Get Registration Code button on the prompt dialog when it appears or by right clicking on the System tray icon and selecting **Register...**

When you have obtained your registration code, enter this into the prompt dialog and click Continue.

Getting Support

SpamWeasel is available free to anyone, provided the licence conditions are accepted and followed.

Because the SpamWeasel is free, no direct support is available.

If you do have a problem and you can not find the answer by referring to this help, then you should check out our web site - www.mailgate.com - to see if there is an answer to your problem.

We would be interested to hear your comments since they help us provide answers to common problems on the web site. Please use the form on the web site or email us at SpamWeasel@mailgate.com.

As information about new spam sites becomes available we will publish updated rule and pattern files. You should check out the SpamWeasel pages on the web site from time to time.

The Message Viewer

Introduction to the Message Viewer

The Mailgate Message Viewer is a utility program to allow direct access to message data stored in the files used by all Mailgate products including the MailGate Mail and Proxy server, its extension modules and the single user Spam mail filter, SpamWeasel.

With SpamWeasel, the Message Viewer is installed as part of the product. To access the program, right click on the System Tray icon



and select Archive Viewer. You may also wish to create a shortcut to this program on your desktop.

Using the Message Viewer you can retrieve any message from the archive files (page 33) in which copies of Spam mail are stored by using the **SpamWeasel Pending Queue** (page 33).

In addition you can:

- **View** (page 34) any message stored in a selected location.
- **Copy** (page 34) a message from one location to another.
- **Move** (page 35) a message from one location to another.
- **Delete** (page 35) a message from its stored location.

See the pages listed above to review how to perform specific tasks or go to **Using the Message Viewer** (page 32) for more general information.

Using the Message Viewer

When first started the Message Viewer will display a screen similar to the one below.

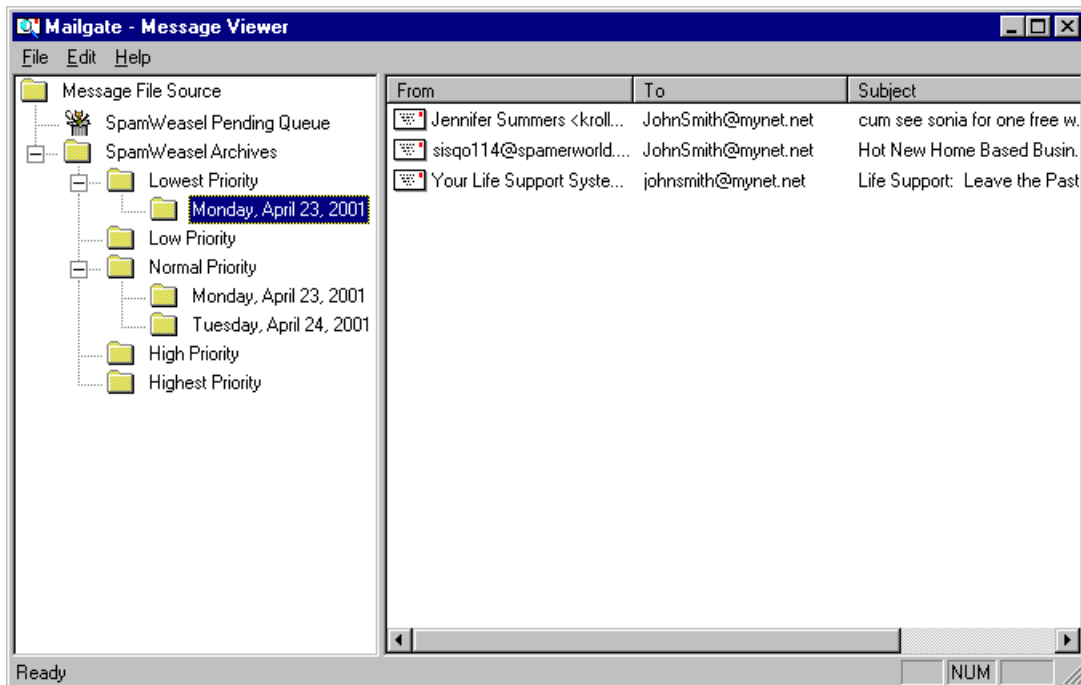


Figure 8 - Message Viewer

The Menu Bar

The Message Viewer menu bar gives access to the program options.



Most of these options are also available by right clicking on an individual message.

Message File Source Panel

The left hand side of the Message Viewer screen displays the range of message sources visible to the Viewer on this machine.

The message sources are presented in a tree structure which can be expanded or collapsed by clicking on the + and - icons. The entire tree can be collapsed and expanded by double clicking on the Message File Source folder at the top of the tree.

For the SpamWeasel there are two main parts to the tree, the **SpamWeasel Pending Queue** (see page 33) which can contain messages waiting to be passed to your email client and the **SpamWeasel Archives** (see page 33) where copies of Spam messages can be stored in daily files.

SpamWeasel Pending Queue

The SpamWeasel Pending Queue can contain messages which are waiting to be passed to your Email client program.

The primary function of the pending queue is to accept messages retrieved from the archives and these messages are passed directly through to the client without re-checking during the next mail collection.

To retrieve an archived message, highlight the required message then copy (page 34) or move (page 35) it to the Pending Queue. The message will be delivered to your Email client when you next make a collection.

SpamWeasel Archives

The SpamWeasel Archives can contain copies of messages identified as Spam.

The archives folder can be expanded to show a folder for each of the available Priority levels. The folders for those priorities in use can be further expanded to show the range of daily archive files currently held in the SpamWeasel system.

Select one of these files to display a list of the stored messages in the message listing panel.

For more details on the configuration and use of priorities and archiving, please refer to the SpamWeasel help system.

Message Listing Panel

When a valid message store has been selected in the **Message Source Panel** (see page 33), a list of all the messages stored in that store is displayed in the Message Listing Panel on the right of the Message Viewer screen.

For each message the viewer will display -

- i. The From address
- ii. The To address
- iii. The Message Subject
- iv. The Message Date

Viewing a Message

To view a stored message:

1. Click on the envelope icon for the required message to highlight it.
2. Select Open either from the File menu item or the popup menu displayed by a right click.

OR

1. Double click on the required message.

The message content will now be displayed.

Note - The Message Viewer displays the raw message data only, so the full message header will be visible and any attachments will display in their email encoded form.

Copying a Message

To copy a stored message to another location:

1. Click on the envelope icon for the required message to highlight it.
2. Select Copy either from the Edit menu item or the popup menu displayed by a right click or press Ctrl-C to copy the message to the clipboard.
3. Select the destination location in the Message File Source tree.
4. Select Paste from the Edit menu item or press Ctrl-V.

You can repeat steps 3 and 4 to make multiple copies of the source message.

You can also use Drag and Drop to copy a message:

1. Click on the required message and hold the mouse button down.
2. Drag the cursor to the destination location in the Message File Source tree.
3. Press and hold the Control Key then release the mouse button.

The message copy will now be added to the end of the destination location's list.

See **Message File Source Panel** on page 33 for more detail on using message locations. In particular if you wish to recover an archived message, you should copy it to the **SpamWeasel Pending Queue** (see page 33) then perform a send and receive in your mail client.

Moving a Message

To move a stored message to another location:

1. Click on the required message and hold the mouse button down.
2. Drag the cursor to the destination location in the Message File Source tree.
3. Release the mouse button.

The message will now be moved from the source list and added to the end of the destination location's list.

See **Message File Source Panel** on page 33 for more detail on using message locations. In particular if you wish to recover an archived message, you should move it to the **SpamWeasel Pending Queue** (see page 33) then perform a send and receive in your mail client.

Deleting a Message

To delete a stored message:

1. Click on the envelope icon for the required message to highlight it.
2. Select Delete either from the Edit menu item or the popup menu displayed by a right click or press the Delete key.

The message will now be deleted.

Technical Reference

Using Wildcards

Using Wildcard Expressions

Some entries in SpamWeasel allow the use of wildcard patterns rather than creating long lists. When creating these entries, you can use special characters to control the matching process.

The following can be used within SpamWeasel to create wildcard patterns :-

?	Matches any single character. T?m matches Tim, Tom and tam.
*	Matches zero or more characters. Hol*d matches holed and hold.
[]	Matches any single character that appears within the square brackets. For example, [0123456789] will match any single digit from 0 to 9. You can also specify a range using a dash - for example, [0-9].
^	Indicates the start of a new line. For example, *^Received:* will only return a match if the Received: is found at the start of a new line.
_	Indicates the match must occur on a single line. For example, *_filename*.exe* will only return a match if the strings filename and .exe occur on a single line.
[^]	Matches any single character that does not appear within the square brackets. For example, [^abc] will match any single character except "a", "b" and "c".
\	Matches the following character literally. For example, * will match a single asterisk and \\ will match a single backslash.
!	Negate - The matching will return true if the negated pattern is NOT matched. Use this option with care in pattern lists.

Using Macros

Using Macro Expressions

When manipulating mail data it is often desirable to access the current mail information. This can be specified in the set-up screens by using the macro facility.

When making a setting which supports the use of Macros, existing mail header fields and other data may be referenced by specifying the field in macro format. This is done by surrounding the required field name with %% (example %%SUBJECT%%). When processed the macro specifier is replaced by the content of the required field.

Example

The SpamWeasel allows you to change the subject of an email which has been identified as SPAM. The setting which defines what the change should be can be configured with :-

```
SPAM - %%SUBJECT%%
```

This setting will cause the software to write a new subject field in the mail, adding 'SPAM - ' in front of the existing subject.

Generally the Macro facility can reference any existing mail header fields. However, when configuring to use macros, be aware that not all header fields are always available in your mails. Availability can depend on your ISP's server system which may be subject to change.

The following fields are regularly found in mail headers and are good candidates for use with macros:-

FROM - The originator's email address.

DATE - The original email's date.

SUBJECT - The original subject.

MESSAGE-ID - The original message ID.

Using Scripting

Introduction to Scripting

The script language was originally developed as a simple, extendable and flexible method of controlling automated processes under Microsoft Windows NT. This language is fully incorporated into the SpamWeasel as the Mailgate Scripting language.

The language consists of a simple set of BASIC like **commands** (see page **Error! Bookmark not defined.**) which provide all the necessary programming constructs for program development. It also allows the addition of application specific functionality via the use of external function libraries. These are standard Windows NT Dynamic Link Libraries (DLLs) which can be developed in any language that can produce DLLs (such as C, C++ or Delphi) to provide addition features to the base script language.

Within SpamWeasel there is also a standard set of **functions** (see page **Error! Bookmark not defined.**) available to the user to perform the actions required for checking mail data.

Scripting Syntax & Commands

The Mailgate Scripting Language consists of a simple set of BASIC like commands which provide all the necessary programming constructs for program development.

The script language is not case sensitive.

The following pages give details of these commands and examples of the language syntax:-

- Integer and string variables

- Integer and string assignments

- Integer and string arithmetic statements

- Integer Boolean statements

- If .. then ... else constructs

- Labels and jumps

 - Subroutines

 - Repeat ... until constructs

 - For ... to/downto ... step...next constructs

 - Inclusion of external source files

 - Definition of external functions

 - Using comments

Integer and string variables

Variables are defined when an assignment to them is first made and they then exist for the duration of the script execution. Variables are of two types, either integer or string. They are named with alphanumeric characters, names are non-case sensitive and must start with a alphabetic character. Integer variables are indicated by ending the name with a '%' and strings with a '\$'. String and integer variables of the same name are permitted.

Some example variable names:-

a%	Integer variable 'A'
A%	Integer variable 'A', same variable as previous
a\$	String variable 'A', exists as a separate entity to integer variable 'A'
test123\$	String variable 'TEST123'
123test\$	Invalid variable name - names must not start with a digit.

Integer and string assignments

Variables are created when assignments are made to a previously uncreated variable. The following statements would create and initialise some variables:-

```
a%=1
b%=2
test$ = "Hello"
```

You can also initialise new variables to the value of an already existing variable of the same type:-

```
a%=1
b%=a%
a$="hello"
b$=a$
```

You cannot assign integer values to strings or vice versa.

Integer and string arithmetic statements

String variables support the operation '+' which acts as 'string' concatenate. The following script segment shows this:-

```
a$="Hello"
b$="there"
c$="folks"
d$=a$+" "+b$+" "+c$
```

The value of **d\$** after executing this script would be "Hello there

folks". The '+' operator is the only one supported by string variables.

Integer variables support the following arithmetic operators:-

+ - / * and or xor not

The logic operators **and**, **or**, **xor** and **not** do bitwise operations between two integer values, for example '**7 and 3**' would result in the value 4. The integer arithmetic evaluator also supports parenthesis and operator precedence as per the standard rules. Some example arithmetic assignments:-

```
a%=1
```

```
b%=2
```

```
c%=3
```

```
result%=a*(b+c%) and 121
```

```
notres%=not result%
```

```
d%=-c%
```

Integer Boolean expressions

Non-zero values are considered true and zero values false. Two constants exist, TRUE and FALSE which have the value -1 and 0 respectively. Boolean expressions can only exist in if...then...else and repeat...until statements. The following comparison operators are supported:-

= < > <= >= <>

These will evaluate to either the TRUE or FALSE (-1 or 0). The Boolean operators and, or, xor and not work as logic statements on these values. For example:-

```
(a%=1) and (b%=2)
```

would evaluate to TRUE if a% was equal to 1 and b% was equal to 2.

If ... then ... else ... endif

The if statement comes in two forms, the single line form and the multiple line form. If the whole if..then..statement is on one line the '**endif**' is not needed. If the statement is split a number of lines the statement must be terminated with '**endif**'. For example:-

```
if a%=1 then b%=2
```

```
or
```

```
if a%=1 then
```

```
    b%=2
```

```
endif
```

The else section of the statement is optional. You can separate statements with the '&' character which acts as a statement separator. You may also use this statement separator anywhere in a

script. You can join lines of a script to be treated as a single line with the '_' character, again this can be used anywhere in a script.

Some more example if statements:-

```
if a% < 3 then b%=1 & c%=2 else b%=2 & c%=1
```

```
if a% < 3 then b%=1 _  
    c% = 2
```

```
if a% < 3 then  
    b%=1  
    c%=2  
else  
    b%=2  
    c%=1  
endif
```

```
if a%=1 then  
    if b%=2 then  
        c%=1  
    endif  
    d%=1  
endif
```

Notice that nesting of statements is supported.

Labels and jumps

Unlike BASIC the script language does not have line numbers. It supports jumps but you do this by defining labels in the script. A label is given a text value and is defined by preceding it with a ':'. For example:-

```
:testlabel
```

would define the label '**testlabel**' at the particular point in the script. You can jump to labels with the '**goto**' statement. The '**goto**' statement is followed by the label name terminated by a ':'. For example:-

```
goto testlabel:
```

would jump to the above defined label. Think of the ':' symbols joining up to remember where to put the ':'. It is in fact not necessary to use the '**goto**' command and this can be omitted but was included for clarity. The following would be equivalent to the above '**goto**' statement.

testlabel:

The most common use of jumps is in 'if' statements. For example:-

```
if a%=1 then goto testlabel:
if a%=2 then testlabel:
```

Both of the above statements would jump to the label if the condition evaluated to true.

Subroutines

Labels can also be used to mark the start of a subroutine.

Subroutines are called by the '**gosub**' statement and control returns to the calling point when the '**return**' statement is encountered, for example:-

```
gosub test:
end
:test
a%=1
b%=2
return
```

Notice the use of '**end**' to stop the program execution 'dropping through' to the subroutine. All variables are considered global and so subroutine script code can access variables created outside the subroutine and vice versa.

Repeat ... until constructs

The repeat ... until construct repeats the enclosed statements until the logical expression evaluates to true. For example:-

```
count%=1
repeat
  a%=1
  b%=a%*2
  count%=count%+1
until count%=10
```

You can nest repeat ... until statements if desired.

For ... to/downto ... step ... next

This statement offers the familiar BASIC looping construct, some examples:-

```
for i%=1 to 10
  a%=i%*2
next
```

```

for i%=a%*2 to a%*3 step 2
    j%=2
next i%
for i%=10 downto 1 step 2
    j%=2
next

```

Notice that the inclusion of the loop variable after the 'next' is optional.

Inclusion of external source files

The **#include** directive includes a source file into the current script at the current point. This is for including lists of external functions or script code for previously written useful subroutines. Include directives can be nested across files up to a limit of 32. Example of include statement:-

```
#include "external.inc"
```

External function definitions

The external function definition directive is the key to the scripting systems power and flexibility. It defines the call interface to routines in a DLL file which have been developed in a suitable language. Once defined the functions can be used as part of the script language. The external definition statement has the following form:-

```
#external <name> ([<parameters>]) [as [async ]<type>] in
<library>
```

where

<name> is the name of the function in the external library.

<parameters> lists the parameters expected. Parameters can be either **string** or **integer** and there may be up to 20. These are listed in the **<parameters>** section separated by commas, for example **string,integer,integer**.

async indicates that the function completes asynchronously. Details of this mechanism will be provided in the developers guide to the scripting language.

<type> give the return type of the function which can be **string** or **integer**. Functions that don't return a value can be defined by omitting the **as <type>** section of the statement.

<library> gives the filename of the external function library.

Thus some example external function definitions would be:-

```
#external OpenFile(string) as integer in "c:\external\external.dll"
#external CloseFile(integer) in "c:\external\external.dll"
#external Output(string) in "c:\external\external.dll"
#external IntToStr(integer) as string in "c:\external\external.dll"
```


Functions that return integers can be used in integer expressions and those that return strings in string expressions. For example:-

```
file%=OpenFile("test.txt")
a$=IntToStr(32)
```

Comments in Scripts

Comments can be added to scripts by preceding them with `//` as per the C++ convention. This turns the rest of the current line into a comment. Comments do not effect the execution speed of the script. For example:-

```
// This is a comment
// So is this
a%=1 // Set a% to the value 1
```

Scripting Functions

There are a number of standard functions available when using the MailGate Scripting Language. These are listed below with full details on the following pages.

String Functions

```
length(<string>)
left(<string>,<count>)
right(<string>,<count>)
mid(<string>,<start>,<count>)
pos(<string>,<substring>)
ascii(<string>)
chr(<integer>)
value(<string>)
str(<value>)
hex(<value>)
```

File I/O Functions

```
fopen(<filename>,<mode>,<sharing>)
fclose(<handle>)
feof(<handle>)
ftell(<handle>)
ferror(<handle>)
```

fread(<handle>,<count>)
fwrite(<handle>,<message>)
fseek(<handle>,<offset>,<from>)

Windows Registry Functions

regqueryval(<hive>,<key>,<value>)
regquerystr(<hive>,<key>,<value>)
regsetval(<hive>,<key>,<valuename>,<value>)
regsetstr(<hive>,<key>,<valuename>,<value>)

Event Log Functions

logevent(<message>)
debug(<string>)

Spam Mail Functions

IsOK()
IsSpam()
HeaderFieldExists(<FieldName>)
HeaderFieldValue(<FieldName>)
ParseAddress(<AddressString>)
GetFirstAddress(<FieldName>)
GetNextAddress()
MessageSize()
IsValidDate(<DateString>)
WildcardMatch(<String>,<MatchPattern>)
WildcardMatchHeader(<MatchPattern>)
WildcardMatchBody(<MatchPattern>)
MatchesListItem(<PatternList>,<String>)
HeaderMatchesListItem(<PatternList>)
BodyMatchesListItem(<PatternList>)
FindWordInString(<WordList>,<String>)
FindWordInHeader(<WordList>)
FindWordInBody(<WordList>)

Function Length(<String>)

Parameters -

Value	Type	Description
<String>	string	String to get length of

Return Type - Integer

Operation

Returns the length of the passed string expression as an integer value.

Function Left(<String>,<Count>)

Parameters -

Value	Type	Description
<String>	string	String to extract left portion from
<Count>	integer	Number of characters of <string> to return

Return Type - String

Operation

Returns a string of the <count> leftmost characters from <string>. If the length of <string> is less than <count> the whole of <string> is returned.

Function Right(<String>,<Count>)

Parameters -

Value	Type	Description
<string>	string	String to extract right portion from
<Count>	integer	Number of characters of <string> to return

Return Type - String

Operation

Returns a string of the <count> rightmost characters from <string>. If the length of <string> is less than <count> the whole of <string> is returned.

Function Mid(<String>,<Start>,<Count>)

Parameters -

Value	Type	Description
<String>	string	String to extract portion from
<Start>	integer	Start position to extract portion
<Count>	integer	Number of characters of <string> to return

Return Type - String

Operation

Returns a string of <count> characters starting from position <start> from <string>. If there are less than <count> characters after position <start> in <string> a string of the available characters after <start> is returned. If <start> is a position after then end of <string> a blank string is returned.

Function Pos(<String>,<SubString>)

Parameters -

Value	Type	Description
<String>	string	String to search
<Substring>	string	Search string

Return Type - Integer

Operation

Returns the position of the first occurrence of <substring> in <string>. If <substring> does not exist in <string> returns zero.

Function Ascii(<String>)

Parameters -

Value	Type	Description
<String>	string	String to get code for

Return Type - Integer

Operation

Returns the ASCII code for the first character of the passed string expression as an integer value.

Function Chr(<Integer>)

Parameters -

Value	Type	Description
<Integer>	integer	ASCII code to convert to string

Return Type - String

Operation

Returns a string containing the single character for the passed ASCII code.

Function Value(<String>)

Parameters -

Value	Type	Description
<String>	string	String to convert

Return Type - Integer

Operation

Returns the integer value represented by the passed string. If the string does not represent an integer value zero is returned.

Function Str(<Value>)

Parameters -

Value	Type	Description
<Value>	integer	Value to convert to string

Return Type - String

Operation

Returns a string of digits representing <value>.

Function Hex(<Value>)

Parameters -

Value	Type	Description
<Value>	integer	Value to convert to string

Return Type - String

Operation

Returns a string of hex digits representing <value>.

Function Fopen(<Filename>,<Mode>,<Sharing>)

Parameters -

Value	Type	Description
<Filename>	string	Name of file to be opened
<Mode>	string	<p>The string class=pmode specifies the type of access requested for the file, as follows:-</p> <p>class=op"r" Opens for reading. If the file does not exist or cannot be found, the class=op>fopen call fails.</p> <p>class=op"w" Opens an empty file for writing. If the given file exists, its contents are destroyed.</p> <p>class=op"a" Opens for writing at the end of the file (appending) without removing the EOF marker before writing new data to the file; creates the file first if it doesn't exist.</p> <p>class=op"r+" Opens for both reading and writing. (The file must exist.)</p> <p>class=op"w+" Opens an empty file for both reading and writing. If the given file exists, its contents are destroyed.</p> <p>class=op"a+" Opens for reading and appending; the appending operation includes the removal of the EOF marker before new data is written to the file and the EOF marker is restored after writing is complete; creates the file first if it doesn't exist.</p> <p>class=opt Open in text (translated) mode. In this mode, class=kn CTRL+Z is interpreted as an end-of-file character on input. In files opened for reading/writing with class=op "a+", class=op fopen checks for a class=kn CTRL+Z at the end of the file and removes it, if possible. This is done because using class=op>fseek and class=op ftell to move within a file that ends with a class=kn CTRL+Z, may cause class=op>fseek to behave improperly near the end of the file.</p> <p>Also, in text mode, carriage return-linefeed combinations are translated into single linefeeds on input, and linefeed characters are translated to carriage return-linefeed combinations on output. When a Unicode stream-I/O function operates in text mode (the default), the source or destination stream is assumed to be a sequence of multibyte characters. Therefore, the Unicode stream-input functions convert multibyte characters to wide characters (as if by a call to the class=op>mbtowl function). For the same reason, the Unicode stream-output functions convert wide characters to multibyte characters (as if by a call to the class=op>wctomb function).</p> <p>class=op b Open in binary (untranslated) mode; translations involving carriage-return and linefeed characters are suppressed.</p>
<Sharing>	integer	<p>Controls sharing of the file with other processes whilst open by the calling process. Can be one of the following values:-</p> <ul style="list-style-type: none"> 0 or 16 - deny read and write access 32 - deny write access 48 - deny read access 64 - deny none

Return Type - Integer

Operation - Opens the file and returns a value to be used as the file handle in the other file functions. Returns zero if error.

Function Fclose(<Handle>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function

Return Type - Logical Integer

Operation

Closes the file. Note that all files opened by a script will automatically be closed when the script session terminates.

Function Feof(<Handle>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function

Return Type - Logical Integer

Operation

Checks if at the end of the passed file. Returns -1 (TRUE) if position is end of file or zero if not.

Function Ftell(<Handle>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function

Return Type - Integer

Operation

Returns the current position in the passed file. Returns -1 if error occurs.

Function Ferror(<Handle>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function

Return Type - Integer

Operation

Returns the current error state of the passed file. Returns zero if no error.

Function Fread(<Handle>,<Count>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function
<Count>	integer	Number of bytes to read from file

Return Type - String

Operation

Returns a string buffer containing the requested number of bytes from the passed file. If an error occurs or there are not enough bytes available in the file the returned string may be shorter than the requested count. Use feof and ferror to determine further information if this occurs.

Function Fwrite(<Handle>,<Message>)

Parameters -

Value	Type	Description
Handle	Integer	A file handle value returned by the fopen function
<Message>	string	Buffer to write to file

Return Type - Integer

Operation

Writes the passed buffer to the passed file. Returns the number of bytes written which, if write successful, will be the length of the passed string. If an error occurs the returned value may be less than the length of the passed string.

Function Fseek(<Handle>,<Offset>,<From>)

Parameters -

Value	Type	Description
<Handle>	integer	A file handle value returned by the fopen function
<Offset>	integer	Number of bytes to move in file
<From>	integer	Indicates the type of move to make as follows:- 0 = move to <offset> bytes from start of file 1 = move to <offset> bytes from current location in file 2 = move to <offset> bytes from end of file

Return Type - Integer

Operation

Moves the current file position as indicated by the parameters. Returns zero if successful or non-zero if error.

Function Regqueryval(<Hive>,<Key>,<Value>)

Parameters -

Value	Type	Description
<Hive>	integer	Value indicating the registry hive to read from as follows:- 0 = HKEY_LOCAL_MACHINE 1 = HKEY_CURRENT_USER 2 = HKEY_CLASSES_ROOT 3 = HKEY_USERS
<Key>	string	Full path giving key to read from, e.g. "software\IDSL\MailGate"
<Value>	string	Name of value to read

Return Type - See Below

Operation

Returns a 32bit integer value giving the DWORD value at the given registry location. Returns zero if error occurs.

Function Regquerystr(<Hive>,<Key>,<Value>)

Parameters -

Value	Type	Description
<Hive>	integer	Value indicating the registry hive to read from as follows:- 0 = HKEY_LOCAL_MACHINE 1 = HKEY_CURRENT_USER 2 = HKEY_CLASSES_ROOT 3 = HKEY_USERS
<Key>	string	Full path giving key to read from, e.g. "software\IDSL\MailGate"
<Value>	string	Name of value to read

Return Type - String

Operation

Returns a string giving the REG_SZ value at the given registry location. Returns zero length string if error occurs.

Function Regsetval(<Hive>,<Key>,<Valuename>,<Value>)

Parameters -

Value	Type	Description
<Hive>	integer	Value indicating the registry hive to read from as follows:- 0 = HKEY_LOCAL_MACHINE 1 = HKEY_CURRENT_USER 2 = HKEY_CLASSES_ROOT 3 = HKEY_USERS
<Key>	string	Full path giving key to write to, e.g. "software\IDSL\MailGate"
<Valuename>	string	Name of value to write
<Value>	integer	32bit value to write to registry as REG_DWORD data

Return Type - Integer

Operation

Writes the given data value to the registry as a REG_DWORD item. Returns zero if successful or Win32 error code if not.

Function Regsetstr(<Hive>,<Key>,<Valuename>,<Value>)

Parameters -

Value	Type	Description
<Hive>	integer	Value indicating the registry hive to read from as follows:- 0 = HKEY_LOCAL_MACHINE 1 = HKEY_CURRENT_USER 2 = HKEY_CLASSES_ROOT 3 = HKEY_USERS
<Key>	string	Full path giving key to read from, e.g. "software\IDSL\MailGate"
<Valuename>	string	Name of value to write to
<Value>	string	String to write to registry as REG_SZ data

Return Type - Integer

Operation

Writes the given data value to the registry as a REG_SZ item. Returns zero if successful or Win32 error code if not.

Function Logevent(<Message>)

Parameters -

Value	Type	Description
<Message>	string	string to write to SpamWeasel event log

Return Type - None

Operation

Writes the given string to the SpamWeasel event log as a script event (type column '~').

Function Debug(<String>)

Parameters -

Value	Type	Description
<String>	string	String to write to debug

Return Type - None

Operation

Write <message> to the system debug console. If the SpamWeasel is being run under a debugger the debugger will display <message>.

Function IsOK()

Parameters - None

Return Type - None

Operation

Calling this function identifies the current mail as NOT Spam and registers this along with the current rule's priority level.

Function IsSpam()

Parameters - None

Return Type - None

Operation

Calling this function identifies the current mail as Spam and registers this along with the current rule's priority level.

Function HeaderFieldExists(<FieldName>)

Parameters -

Value	Type	Description
<FieldName>	string	The header field to look. Do not include the ':'

Return Type - Logical Integer

Operation

Looks for the specified mail header field. Returns True if the field exists otherwise returns False.

Example

```
if HeaderFieldExists("From") then
    <action>
endif
```

<Action> is performed if the From: field is found in the mail header.

Function HeaderFieldValue(<FieldName>)

Parameters -

Value	Type	Description
<FieldName>	string	The header field whose value is required. Do not include the ':'

Return Type - String

Operation

Returns the value associated with the specified header field.

Example

```
if HeaderFieldValue("From") = "Spammer" then
    IsSpam()
endif
```

If the From: field in the mail header exactly matches the string "Spammer" then the mail is identified as Spam.

Function ParseAddress(<AddressString>)

Parameters -

Value	Type	Description
<AddressString>	string	String from which to extract a valid email address.

Return Type - String

Operation

Extracts a valid email address from the specified string. In many cases an email header field will contain a name along with the email address.

Example

```
ParseAddress("(Doe, John) <jdoe@domain.com>")
returns the string jdoe@domain.com
```

Function GetFirstAddress(<FieldName>)

Parameters -

Value	Type	Description
<FieldName>	string	The header field to extract the first address from. Do not include the ':'

Return Type - String

Operation

Some mail header fields (particularly the To: and CC: fields) can contain multiple addresses. This function extracts the first valid email address from the specified field.

Example

```
GetFirstAddress("To")
```

will return the first valid address found in the To: header field. See GetNextAddress() for reading multiple addresses.

Function GetNextAddress()

Parameters - None

Return Type - String

Operation

Special function used in conjunction with GetFirstAddress() to enable reading of multiple addresses from the To: and CC: header fields.

Example

```
addr$ = GetFirstAddress("To")
:loop
    if length(addr$) = 0 then loopend:
    if <test the address> then goto notspam:
    addr$ = GetNextAddress()
    goto loop:
:loopend
IsSpam()

:notspam
```

This example demonstrates the reading of multiple addresses from the To: header field. Using the label jump capability, a loop is defined. The first address is extracted and the loop entered. An exit from the

loop will occur either if the extracted address is 0 length (i.e. there is no address) or the address tests as Not Spam. After the first address has been tested, the next available address is extracted by `GetNextAddress()` and the looping will continue until `GetNextAddress()` returns an empty string or an address tests as Not Spam.

Note that `GetNextAddress()` requires `GetFirstAddress()` to define the header field to read.

Function `MessageSize()`

Parameters - None

Return Type - Integer

Operation

Returns the size of the current message in bytes.

Function `IsValidDate(<DateString>)`

Parameters -

Value	Type	Description
<code><DateString></code>	string	String to check for a valid date. Typically this will be the Date: mail header field.

Return Type - Logical Integer

Operation

Checks the specified string for characters which represent a valid date. Returns True if a valid date is found otherwise returns False.

Example

```
IsValidDate("Fri, 6 Oct 2000 09:05 +0100 (BST)")  
will return True.
```

Function WildcardMatch(<String>,<MatchPattern>)

Parameters -

Value	Type	Description
<String>	string	The string to search for a pattern match.
<MatchPattern>	string	The wildcard pattern string to search <string1> for.

Return Type - Logical Integer

Operation

Checks <string 1> for a match to <string2> where <string2> can contain wildcard specifiers. Returns True if a match is found otherwise returns False.

Example

`WildcardMatch("ABCDEF", "*BCD*")` will return True.

`WildcardMatch("ABCDEF", "*BDE*")` will return False.

Function WildcardMatchHeader(<MatchPattern>)

Parameters -

Value	Type	Description
<MatchPattern>	string	The wildcard pattern to look for in the mail header

Return Type - Logical Integer

Operation

Checks the entire mail header for a match against the specified wildcard pattern. Returns True if a match is found otherwise returns False.

Example

```
if WildcardMatchHeader("@spammer.kom*") then
    IsSpam()
endif
```

Identifies the mail as being Spam if "@spammer.kom" is found anywhere in the header.

Function WildcardMatchBody(<MatchPattern>)

Parameters -

Value	Type	Description
<MatchPattern>	string	The wildcard pattern to look for in the mail body.

Return Type - Logical Integer

Operation

Checks the entire mail body for a match against the specified wildcard pattern. Returns True if a match is found otherwise returns False.

Example

```
if WildcardMatchBody("*make millions*") then
    IsSpam()
endif
```

Identifies the mail as being Spam if the phrase "make millions" is found anywhere in the mail body.

Function MatchesListItem(<PatternList>,<String>)

Parameters -

Value	Type	Description
<PatternList>	string	The name of the patterns list to use as a string.
<String>	string	The string to be checked for a pattern match.

Return Type - Logical Integer

Operation

Check for a match between <string2> and any of the items in the specified patterns list. Returns True if a match is found otherwise returns False.

Example

```
MatchesListItem("My Patterns","String to be Checked")
```

This will return True if there is a pattern in the list "My patterns" which matches to the string "String to be checked". Note that wildcards can be used in the Patterns lists and matching is not case sensitive.

Function HeaderMatchesListItem(<PatternList>)

Parameters -

Value	Type	Description
<PatternList>	string	The name of the patterns list whose items are to be checked against the message header.

Return Type - Logical Integer

Operation

Check for a match between each of the items in the specified Patterns list and the content of the message header. Returns True if any list item matches otherwise returns False.

Example

```
if HeaderMatchesListItem("Usual spam message phrases")
then
    IsSpam( )
endif
```

Message is marked as Spam if a pattern match is found between any of the items in the Patterns list "Usual spam message phrases" and the message header content.

Function BodyMatchesListItem(<PatternList>)

Parameters -

Value	Type	Description
<PatternList>	string	The name of the patterns list whose items are to be checked against the message body.

Return Type - Logical Integer

Operation

Check for a match between each of the items in the specified Patterns list and the content of the message body. Returns True if any list item matches otherwise returns False.

Example

```
if BodyMatchesListItem("Usual spam message phrases") then
    IsSpam( )
endif
```

Message is marked as Spam if a pattern match is found between any of the items in the Patterns list "Usual spam message phrases" and the message body content.

Function FindWordInString(<WordList>,<String>)

Parameters -

Value	Type	Description
<WordList>	string	The words list to use as a string.
<String>	string	The string to check for a word match.

Return Type - Logical Integer

Operation

Checks the words in <string2> for a match against all the words contained in the specified words list. Returns True if a match is found otherwise returns False.

Note that wildcards are not permitted in the Words lists so words must exactly match but matching is not case sensitive.

Example

```
FindWordInString("My Word List","I am a subject")
```

This will return True if any of the words in "I am a subject" exist in the words list "My Word List".

Function FindWordInHeader(<WordList>)

Parameters -

Value	Type	Description
<WordList>	string	The words list to use as a string.

Return Type - Logical Integer

Operation

Checks the words in the message header for a match against all the words contained in the specified words list. Returns True if a match is found otherwise returns False.

Note that wildcards are not permitted in the Words lists so words must exactly match but matching is not case sensitive.

Function FindWordInBody(<WordList>)

Parameters -

Value	Type	Description
<WordList>	string	The words list to use as a string.

Return Type - Logical Integer

Operation

Checks the words in the entire message for a match against all the words contained in the specified words list. Returns True if a match is found otherwise returns False.

Note that wildcards are not permitted in the Words lists so words must exactly match but matching is not case sensitive.

Windows Registry

Windows Registry

The SpamWeasel stores its configuration in the Windows Registry. All settings are stored under the section:-

HKEY_LOCAL_MACHINE

Software

Mailgate Ltd

SpamWeasel

Within the SpamWeasel key are found the main configuration settings. There is also a subkey named Priority. This key has an entry for each rule and the value is the rule priority level as defined in the **Rules Tab** (page 14). See the following pages for details.

Registry - SpamWeasel

Most of the SpamWeasel configuration settings are stored in this Registry key.

Some of the settings are in the form <SettingName><n> where n is the value of the priority for an action setting. n is in the range 1 to 5 where 1 = highest priority and 5 = lowest.

Value Name (Data Type)	Description
Archive<n> (DWORD)	Indicates whether the action should archive a copy of a spam message. 1=Archive Copy. Default=0.
ArchivePath<n> (String)	Provides the drive and path to use when archiving a spam message
CollectionReport (DWORD)	Indicates if a <i>per collection</i> report should be generated. 1=Generate report. Default=1.
DisableRules (Multi-String)	An array of the names of all rules that are disabled.
LastId (DWORD)	Contains the last internal message id used.
ListenPort (DWORD)	The port number to listen to for email client POP3 requests. Default =110.
Logging (DWORD)	The logging detail level to use. 0,1 or 2.
MaxSpamSize (DWORD)	The maximum size of messages in KB to be checked for Spam. Any message over this size will be passed through.
SeparatorChar (DWORD)	Optional setting to define the character used to separate the account name and POP server name when configuring your Email client. The default is character is @. Decimal value for the ASCII character to use. Example:- To use a ? character, set SeparatorChar to decimal 63. Your email account name would then be jdoe?pop.myisp.net.
SubjectMacro<n> (String)	The macro to use to modify the subject line for a spam message being passed on to the email client software.
Tidy (DWORD)	Indicates if old archive files should be automatically deleted. 1=Delete. Default=0.
TidyDays (DWORD)	The number of days worth of archives to keep when tidying. Default=7.
Version (String)	Gives the version number for the currently installed SpamWeasel.

Registry - Priority

The Priority Registry Key has a value entry for each rule listed in the **Rules Tab** (page 14).

The rule name is used for the value name and each is of type DWORD. The DWORD value gives the rule priority in the range 1 to 5 where 1 = highest priority and 5 = lowest.