



RAVMD configuration file

Release Date: February 18, 2003

Product version: 8.4.2

User Guide revision: 2.4

Address: 223, Mihai Bravu Blvd, 3rd district, Bucharest, Romania
Phone/Fax: +40-21-321.78.03, **Hotline:** +40-21-321.78.59

Copyright © since 2001 GeCAD Software® S.R.L.

All rights reserved. This material or parts of it cannot be reproduced, in any way, by any means.

The product and the documentation coming with the product are protected by GeCAD Software's copyright.

GeCAD Software reserves itself the right to revise and modify its products according to its own necessities.

This document describes the product at this writing and may not correctly describe the latest developments. For this reason, we recommend you to periodically check our website, <http://www.ravantivirus.com>, for the latest versions of product documentations.

GeCAD Software cannot be hold responsible for any special, collateral or accidental damages, related in any way to the use of this document.

GeCAD Software's entire liability, depending on the action, cannot go beyond the price paid for the product described in this material.

GeCAD Software does not guarantee either implicitly or explicitly the suitability of this material for your specific needs. This material is provided on an "as-is" basis.

GeCAD Software trademarks: GeCAD, GeCAD Fast Commander, GFC, RAV Reliable AntiVirus, A.V.A.C., RAlert, RAVUtil, RAVeSpy, R.A.C.E., RAX, WisDOM.

The following are registered trademarks of their respective owners: Times New Roman, Courier, Arial, IBM, OS/2, Intel, Microsoft, MS-DOS, Windows, Windows95, Windows98, WindowsNT, OEMM, F-PROT, TBSCAN, VirSCAN, TBAV, DSAV, DrWEB, AVP, MSAV, MS Office, MS Word, MS Access, MS Excel, MS Visual Basic, NetWare.

Terms and conditions of the License Agreement

RAV Reliable AntiVirus is a registered trademark of GeCAD Software S.R.L. (hereafter referred as "GeCAD Software"). All the products from the **RAV AntiVirus** family are offered to our clients under the terms and conditions of the License Agreement accompanying all the products of GeCAD Software.

Before installing or using The Software, please read carefully this License Agreement, because it represents a legal agreement between you and GeCAD Software for the software product you are installing, which includes the software itself and the related documentation. By installing or otherwise using the software, you accept all the terms and conditions of this agreement. If you do not accept the terms of this agreement, you do not have the rights to install or otherwise use The Software.

On the distribution CD-ROM, you may find other programs, in addition to the one you have bought. These programs are offered for evaluation only and are the object of separate terms of license. These terms are included in the **Evaluation license** section of the **License Agreement**.

CONTENTS

RAVMD configuration file	5
<i>NAME</i>	5
<i>SYNOPSIS</i>	5
Structure	5
Macros	6
What is new in ravmd 8.4.2	7
What was new in ravmd 8.4.1	7
What was new in ravmd 8.4.0	8
Definitions	9
<i>SECTION DESCRIPTIONS</i>	10
Regular Expression Declaration section	11
<i>Syntax</i>	11
Action Definitions section	12
<i>Syntax</i>	12
FAQ 1: Scanning existing mails	14
FAQ 2: Separate domain file	14
FAQ 3: Changing the contents of the warn.txt file	14
FAQ 4: Modified message is not appearing in the warn.txt file	14
Warning Mails Message Declarations section	15
<i>Syntax</i>	15
FAQ 5: Warning messages not sent	17
FAQ 6: Omitting the SUBJECT from warning mail	18
FAQ 7: Changing the warning message in case of attachment match	18
FAQ 8: Changing the warning message in case of subject match	18
FAQ 9: Changing the warning message in case of message's body content match	19
FAQ 10: Offering information about viruses in the warning mails	19
FAQ 11: Stopping update notifications	19
Antispam Definitions section	20
<i>How does it work</i>	20
<i>Group Declarations</i>	23
What is a group	23
The [global] group	23
Defining other groups	23
<i>THE_include DIRECTIVE</i>	23
How do I configure groups	24
How do I configure separate antispam options for my groups	24
FAQ 12: Creating different rules for a domain	24
FAQ 13: Example on how to set the domains and the IP addresses	25
FAQ 14: Global group configuration	25
FAQ 15: Excluding one particular account	25
<i>The Advanced Content Filtering feature</i>	26
<i>What is the Advanced Content Filtering feature of ravmd</i>	26
<i>How does it work</i>	26
FAQ 16: Rejecting double extension files	28
FAQ 17: Denying certain attachment extensions	28
FAQ 18: Example of subject filtering	29
FAQ 19: Example of body filtering	29

FAQ 20: Digging even deeper in string-based body filtering	30
<i>Explaining the parameters</i>	32
Domain parameters	32
Group members	32
Engine actions	34
Engine parameters	35
Warning messages	37
Specifying the sender of the warning mails	43
Antispam parameters	45
Real-time Blackhole List parameters	47
White/Black List parameters	49
Miscellaneous parameters	50
RAV logging system	53
Group-specific parameters	55
Embedded messages	61
<i>BUGS</i>	62
<i>SEE ALSO</i>	62

NAME

ravmd.conf - the configuration file for **ravmd** daemon

SYNOPSIS

This is the configuration file for **ravmd** daemon, **ravmd.conf**, and it contains info on the run-time configuration for **RAV AntiVirus** mail-scanning daemon. After installation, **ravmd.conf** resides in your `${ETCDIR}` directory.

Other documents containing valuable information are available:

- The *Install*, *UnInstall* and *ReadMe* files included in the **tar** or **tar.gz** files containing the setup programs;
- The configuration manuals for **ravmd** filter clients included in the *User's Guide for Mail Servers* and in the **tar** or **tar.gz** files containing the setup programs.

Structure

This configuration file consists of:

- Description of four declaration sections (the *Regular Expression Declaration* section, the *Action Definitions* section, the *Warning Mails Message Declarations* section and the *Antispam definitions* section),
- Group declarations, and
- Explanation of possible parameters.

The info included in this part of the documentation is for reference purposes. Other valuable sources of information are also available. Information pertaining to the installing and uninstalling processes, for instance, is included in the *Install* and *Uninstall* files provided in the **tar.gz** files containing the installation kit for the product you are using. Additional information about **RAV AntiVirus for Mail Servers** (hardware and software requirements, list of installed files, instructions for configuration and updating) is provided in the *ReadMe* file, included in the installation kit.

When presenting the possible parameters in **ravmd**, besides the default values and examples for these parameters *Frequently Asked Questions* are also provided, to help you better understand the functions of **ravmd** and know all the options available in it.

For the users of previous version of **RAV AntiVirus for Mail Servers**, please read the [What is new in ravmd 8.4.2](#) section below.

Macros

Strings preceded by the \$ character (i.e. `${BINDIR}`) are used as *macros* denoting paths that are specific to the operating system you are using.

These macros have the following meaning:

`${BINDIR}` - location of RAV executables;

`${LIBDIR}` - location of RAV libraries;

`${ETCDIR}` - location of RAV configuration files;

`${DATADIR}` - location of RAV logs, temporary files, quarantine folder, activation key, etc.;

`${RAVEDIR}` - location of RAV Engine.

The corresponding paths should replace these macros as follows:

For Linux and Solaris:

`${BINDIR}` = `/opt/rav/bin`

`${LIBDIR}` = `/opt/rav/lib`

`${ETCDIR}` = `/etc/opt/rav`

`${DATADIR}` = `/var/opt/rav`

`${RAVEDIR}` = `/var/opt/rav/rave` for products running on i386 platforms, `/var/opt/rav/s390rave` for products running Linux on s390 platforms, `/var/opt/rav/spc_rave` for products running on sparc platforms and `/var/opt/rav/ppc_rave` for products running Linux on PowerPC platforms.

For FreeBSD, NetBSD and BSDi:

`${BINDIR}` = `/usr/local/bin`

`${LIBDIR}` = `/usr/local/lib/rav`

`${ETCDIR}` = `/usr/local/etc/rav`

`${DATADIR}` = `/var/rav`

`${RAVEDIR}` = `/var/rav/rave` for products running on i386 platforms.

For OpenBSD:

`${BINDIR}` = `/usr/local/bin`

`${LIBDIR}` = `/usr/local/lib/rav`

`${ETCDIR}` = `/etc/rav`

`${DATADIR}` = `/var/rav`

`${RAVEDIR}` = `/var/rav/rave` for products running on i386 platforms.

For MacOS X:

`${BINDIR} = /usr/local/rav/bin`

`${LIBDIR} = /usr/local /rav/lib`

`${ETCDIR} = /usr/local/rav/etc`

`${DATADIR} = /usr/local/rav`

`${RAVEDIR} = /usr/local/rav/ppc_rave.`

What is new in ravmd 8.4.2

Version 8.4.2 of **ravmd** has been released mainly because some users of **RAV AntiVirus for Mail Servers** have asked, for legal reasons, the warning mails issued by our product to be sent as **Bcc** copies.

Here is a list of the new features included in version 8.4.2 of **ravmd**:

- **supervisor_addr**: New parameter in **ravmd** version 8.4.2 used for specifying the mail addresses of the administrators to be notified:
 - in case a **ravmd** malfunction is recorded; or
 - with 14 days before your license will expire; or
 - on each **ravmd** reload in the last 14 days before your license expires, until you extend your subscription period.
- **disclose_supervisor**: New parameter in **ravmd** used for specifying if the address of the supervisor will or will not be disclosed in the warning mail's header and body.
- **disclose_sender**, **disclose_receivers**, **disclose_admin**: New parameters in **ravmd** used for specifying if the addresses of the corresponding sender/receivers/administrators will or will not be disclosed in the warning mail's header and body.
- Before the 8.4.2 version of **ravmd**, the warning mails were all sent having their receivers listed in the **To:** header. Now all the warning mails are sent by default as **Bcc** messages.
- String comparisons executed by WBL library for domain names and e-mail addresses are not case sensitive.
- Eliminating bug in parsing the mail header for obtaining the IP addresses used by WBL and RBL.

What was new in ravmd 8.4.1

Version 8.4.1 of **ravmd** includes the following new features:

- **warn_domains**: new flag for **warn_sender**, **warn_receivers** and **warn_admin**; when using **warn_domains**, only the users from the domains specified by the 'domain' parameter are notified;
- **wbl_discard**: new action for mails matching a rule in **WBL**. When using this

parameter, the corresponding mails are discarded (rejected with no bounce back to the sender, saving therefore valuable bandwidth for users);

- **forward**: new action for bulk mails. The bulk mails are forwarded to the mail addresses specified by the antispam 'forward_to' parameter;
- **forward_to**: new parameter for the antispam section; contains a list of mail addresses where the spam mail is forwarded;
- **new log levels**:
 1. + **2048** -> log actions performed for bulk mails
 2. + **4096** -> log all the IPs from 'Received:' mail header field
 3. + **8192** -> log the rule matched by content filter.
- The default location of RAV directories may be now changed.
- The **_include** directive now supports paths relative to the current file.
- The directory structure for **RAV AntiVirus** products running on BSDs has been changed in order to comply with FreeBSD's hierarchy recommendations.

For more information, read the *Release Notes* for **RAV AntiVirus for Mail Servers** version 8.4.2, available [here](#).

What was new in ravmd 8.4.0

The main enhancement brought by **ravmd** version 8.4.0 is the integration of the antispam module. The antispam module is actually a combination between the new bulk mail module and features already implemented in **ravmd**, i.e. **Real-time Blackhole List (RBL)** and the **White/Black List (WBL)**. Correctly configured, this combination should be a winner in the fight against unsolicited mails.

Note: You can find more details on the new features of **ravmd** version 8.4.0 in the *Release Notes for ravmd 8.4.0* and in the *User Guide for RAV AntiVirus for Mail Servers* (version 1.41 or later of this document). You can find these documents on <http://www.ravantivirus.com/support/documentation.php>.

The new antispam module included in **ravmd** version 8.4.0 triggers important changes in this document.

- A new section called *Antispam definitions* was added after the *Warning Mails Message declaration* section, explaining how the new antispam module works.
- The parameters specific to the antispam module are detailed in the *Antispam parameters* sub-section under the *Explaining the parameters* section.
- New actions are available for bulk mails: save, embed, add_header, add_subject, deliver, reject, discard.
- New group-specific parameter: **antispam_configuration**.

Among other changes to be noted in **ravmd** version 8.4.0:

- the folder structure for **ravmd** has changed. For instance, **ravmd.conf** and **actions** files are to be found in your `#{ETCDIR}` folder, instead of the older `/usr/local/rav8/etc` location (under Linux). The file **antispam** containing the settings for the antispam module is to be found in the same folder.
- the format (font, paragraph) of this document has changed since its previous version.

Definitions

Some concepts frequently used in the programming world might have different understandings for the purpose of this manual. Here you can find these terms and the meanings they are given in this document:

- A **variable** is a place where we can store data;
- A **string** is a sequence of characters ending with a new line or delimited by quotes;
- A **boolean** is one of the keywords: **yes/no**;
- An **enumeration** is a sequence of words separated by commas;

Note: The spaces are ignored by **ravmd**, except for **WBL**, where spaces are accepted instead of commas.

- A **regexp** is a POSIX regular expression.
- A **group** is a category of users (senders or receivers) that have different mail addresses and/or domains, but share the same action parameters for **ravmd**.
- A **macro** is a stored template of instructions to be replaced with actual values by **ravmd**;
- A **commented** line is a line beginning with a hash (#) character. All commented lines are ignored. All the lines containing only white spaces are also ignored;
- A **default** is a predefined value for one parameter. If that **parameter** is missing from **ravmd.conf** then it is considered to have that **default** value.

The values following the '=' sign in *parameters* may be: a **string**, a **boolean**, a **regexp** or an **enumeration**.

Note: The section and parameter names are **not** case sensitive.

- An *object*, for the purpose of this *User Guide*, represents any type of file (archive, executable, .dll, etc.), a folder, a disk sector, the boot sector of a hard disk, the memory or any other hardware component that can be scanned by **ravmd**.
- An object is classified as *infected* when **ravmd** has detected a virus in that object (i.e. the virus signature is included in the signature database of **ravmd**).
- An object is classified as *suspicious* when it contains potentially dangerous code, but this code does not match to any virus in the current signature database of **ravmd**. This might be the case of a new virus or a new variant of a virus already included in the signature database of **ravmd**.
- An object is classified as *clean* if **ravmd** identifies no virus or suspicious code when scanning that object.
- An object is classified as *cleaned* when **ravmd** has identified a virus/suspicious code while scanning that object, but the program was able to disinfect it.
- An object is classified as *uncleaned* when **ravmd** has identified a virus/suspicious code while scanning that object, but the program was not able to disinfect it.

SECTION DESCRIPTIONS

Four different sections are included in `ravmd.conf`: the *Regular Expression Declarations* section, the *Action Definitions* section, the *Warning Mails Message Declarations* section and the *Antispam Configuration* section. These sections are presented below in the following format:

- Short description;
- Keyword representing the beginning of the section;
- Syntax;
- Example.

Some *Frequently Asked Questions* (FAQs) are also provided in connection with some relevant aspects of these sections. The *Frequently Asked Questions* are meant to help you with the practical aspects of using **RAV AntiVirus for Mail Servers**.

The *Frequently Asked Questions* were selected from the questions asked by users of **RAV AntiVirus for Mail Servers** on the mailing lists available for our products.

Regular Expression Declaration section

In the *Regular Expression Declaration* section you can define all the regular expressions you will use for the content filtering feature. This section can appear anywhere in the configuration file, as long as it is placed before the group definitions.

The **Regular Expression Declaration** section begins with the following keyword:

```
_define_regular_expressions
```

Syntax

```
variable = string
```

or

```
variable = regexp
```

Example 1:

```
for_subject_filter = I love you
```

This regular expression defines a filter for all the mails including the string "I love you" in the **Subject** field.

Example 2:

```
file_regexp = .*\.exe
```

This regular expression defines a filter for all mail messages having **.exe** attachments.

Action Definitions section

In the *Action Definitions* section you can define the actions you want to be used by **ravmd**, depending on the file status (for more information, please refer to the [Definitions](#) section of this document). The *Action Definitions* section can appear anywhere in the configuration file as long as it is placed before group definitions.

This section starts with the keyword: `_define_actions`.

Syntax

variable = enumeration

Depending on the scanning status of the mail message (infected, suspicious, subject/attachment/content filter match), the variable will be associated with some different actions (*clean, move, copy, delete, rename, ignore, reject, discard*). The enumeration contains one or multiple actions separated by a comma. The actions from this enumeration are executed by **ravmd** in the order you specify.

Depending on the scanning status, the following valid actions are supported:

- for infected files: **clean, move, copy, delete, rename, ignore, reject, discard**.
- for suspicious files: **move, copy, delete, rename, ignore, reject, discard**.
- for mails matching the subject filter: **copy, ignore, reject, discard**.
- for files matching the attachment filter: **move, copy, delete, rename, ignore, reject, discard**.
- for mails matching the content filter: **move, copy, delete, ignore, reject, discard**.
- for mails tagged as spam: **save, embed, add_header, add_subject, deliver, reject, discard**.

Note 1: If the last action you define for infected mails is not one of the following: **Discard, Reject** or **Ignore**, the **Reject** action is automatically applied.

Note 2: If the last action you define for bulk mails is not one of the following: **Deliver, Reject** or **Discard**, the **Deliver** action is automatically applied.

For more information, please refer to the `actions` file included in your setup program.

The following table includes a description of all the actions available in **ravmd** (first column), a description of these actions (second column) and a listing of circumstances when each specific action is available (third column), depending on the scanning status:

Action	Description	Available for objects with status:
<i>Clean</i>	Ask ravmd to clean the infected file.	Infect
<i>Move</i>	Ask ravmd to move the file to quarantine (equivalent to Copy + Delete actions).	Infect, suspicious, attach match, content match
<i>Copy</i>	ravmd will copy the infected object to quarantine.	Infect, suspicious, subject match, attach match, content match
<i>Delete</i>	ravmd will delete the infected object and replace it with a new file automatically generated. The file's name is warn.txt and this file is customisable (for more details please refer to FAQ 3). Note that ravmd doesn't change the mail file size because some protocols (like IMAP) may request the mail size first and then the mail body. So, the warn.txt file will be filled with spaces to fit the original file length.	Infect, suspicious, attach match, content match
<i>Rename</i>	The file will be renamed using the <code>rename_ext</code> extension specified in the configuration file.	Infect, suspicious, attach match
<i>Ignore</i>	The file is ignored, no action is taken and the e-mail is delivered.	Infect, suspicious, subject match, attach match, content match
<i>Reject</i>	The e-mail is rejected; it will not be delivered to any of its recipients, but will be bounced back to the sender.	Infect, suspicious, subject filter, attach match, content match, bulk mail
<i>Discard</i>	The e-mail is discarded; it will not be delivered to any of its recipients and will not be bounced back to the sender.	Infect, suspicious, subject filter, attach match, content match, bulk mail
<i>Deliver</i>	The original mail is delivered to its recipients even though it is tagged as Spam.	Bulk mail
<i>Save</i>	The bulk mail is saved in the Quarantine directory.	Bulk mail
<i>Embed</i>	Creates an embedded mail including a custom message and the bulk mail as attachment.	Bulk mail
<i>Forward</i>	New action available from version 8.4.1. The bulk mail tagged as Spam is forwarded to the mail addresses specified by the antispam <code>forward_to</code> parameter.	Bulk mail
<i>add_header</i>	Add a custom extra header to the bulk mail tagged as Spam.	Bulk mail
<i>add_subject</i>	Affix a custom string in the Subject field of a bulk mail tagged as Spam.	Bulk mail

Table 1: Actions available in ravmd.

The following *Frequently Asked Questions* will help you better understand the characteristics of the actions available in **ravmd**.

FAQ 1: Scanning existing mails

Question: I just installed **RAV AntiVirus for Mail Servers**. How can I scan mail messages existing prior to this installation?

Answer: If you wish to scan mail messages existing prior to the installation of **ravmd**, do the following:

```
`${BINDIR}`/ravav -AM --smart --report=/tmp/ravreport.txt [path to mail accounts]
```

This setting results in scanning a mailbox (with **Ignore** as default action) and delivering a report (ravreport.txt) in the tmp directory.

FAQ 2: Separate domain file

Question: Is it possible to put all domains to be scanned in a separate hash or normal file?

Answer: Create a domains file in `\${ETCDIR}` and operate the following changes:

In ravmd.conf, replace the "domains to scan (separate them with ',')" section with:

```
domain1.com, domain2.com, domain3.net, domain4.com, domain5.org
```

Then use:

```
domain = _include `${ETCDIR}`/domains.list
```

FAQ 3: Changing the contents of the warn.txt file

Question: Recipients who get a virus-infected email have their attachments replaced with a warning message set in the warn.txt file. How can I change the contents of this warn.txt file?

Answer: Please edit `\${ETCDIR}`/languages/english and customize the value of:

```
warn_txt_msg_english = "Your message".
```

Then restart **ravmd** with:

```
kill -HUP `cat `${DATADIR}`/run/ravmd.pid` .
```

Please note that warn.txt has exactly the size of the deleted attachment and in some cases, because of the small size of the attachment, you will not be able to view your complete customized message.

FAQ 4: Modified message is not appearing in the warn.txt file

Question: I have modified the warn.txt message in my configuration but the modified message is not appearing in the warn.txt file. Why is that happening?

Answer: Either you did not correctly change the warn.txt message or the infected attachment file is too small to allow the entire warn.txt message to be displayed (see the answer to [FAQ 3](#) for more details). The steps to be followed are:

- In `${ETCDIR}/languages/english` use: `warn_txt_msg_english = "Your message here"`
- In `${ETCDIR}/languages/english.equiv` use: `warn_txt_msg = warn_txt_msg_english`
- Then restart `ravmd` with: `kill -HUP `cat ${DATADIR}/run/ravmd.pid``

Warning Mails Message Declarations section

In the *Warning Mails Message Declarations* section you can define the subjects and the messages for the warning mails that will be sent to those interested. The *Warning Mails Message Declarations* section can be declared anywhere in the configuration file as long as it is placed before the group definitions.

This section starts with the keyword:

```
_define_strings
```

Syntax

variable = string

The string defining one warning mail should give the user some basic information such as: what file has caused the warning, who sent that file and whom was it intended for, what is the warning about, what action was taken by **ravmd** and so on. One example of good warning message would be the following:

"That file coming from that sender and addressed to this user is infected with this virus. The action taken by ravmd was this."

The warning mails can be sent to: sender, receivers and administrators, according to your configuration. Before version 8.4.2 of **ravmd**, the warning mails were sent having all their receivers listed in the **To:** header. Now all the warning mails are sent by default as **Bcc** messages. For more information, please refer to the [Explaining the parameters - Warning messages](#) section of this *User Guide*.

Of course, not all the warning mails are about virus-infected files. There are some other situations when you might want to be alerted by **ravmd**. For instance, you might want **ravmd** to alert you when a mail containing one specific type of attachment has arrived on your mail server or when one of your users is trying to transmit a confidential document. Of course, in these cases the warning mail should change accordingly to that specific situation. However, in all the cases some information is *always* included (the name of the file causing the alert, the sender, the action, etc.).

Warning mails are also *automatically* sent to the mail addresses specified in the `supervisor_addr` parameter from the `[global]` group and to `postmaster@hostname` and `root@hostname`, where `hostname` is the value for the `on_host` parameter in the `[global]` group or the host name, as returned by the `gethostbyname()` function:

- in case a **ravmd** malfunction is recorded; or
- with 14 days before your license will expire; or
- on each **ravmd** reload in the last 14 days before your license expires, until you extend your subscription period.

The `supervisor_addr` parameter was introduced starting with version 8.4.2 of **ravmd**. For all prior version, these messages were sent to all mail addresses specified in the `admin_addr` parameters from your groups.

The information included in a warning mail is always based on *macros*. One macro is (for the purpose of this document) a stored template of instructions containing one piece of info from the string representing the warning mail. `FILE_NAME`, for instance, is an example of macro containing the name of the file generating the warning mail. **ravmd** will replace this macro with the actual name of the trouble-making file. `VIRUS_NAME`, on the other hand, is a macro containing the name of the virus generating the warning mail. **ravmd** will replace this macro with the actual name of the virus.

The string that the warning message will be created from can contain the following macros:

<code>FILE_NAME</code>	The full path to the scanned file and its name.
<code>ATTACH_NAME</code>	The name of the attachment scanned by ravmd .
<code>VIRUS_NAME</code>	The name of the virus discovered by ravmd .
<code>FROM_USER</code>	The mail address of the sender.
<code>ON_HOST</code>	The host ravmd is running on.
<code>TO_USER(S)</code>	The mail addresses of the receivers.
<code>SUBJECT</code>	The mail's subject.
<code>QUARANTINE_NAME</code>	The name of the file saved to the Quarantine folder using the Move or Copy action.
<code>SAVED_FILE_NAME</code>	The name of the mail file saved to quarantine when <code>save_infected=yes</code> and/or <code>save_suspicious=yes</code> .
<code>HEADER_RECEIVED</code>	Macro replaced with the Received: lines in the received mail's header (if this information exists). This helps you finding the infected machine more easily.
<code>HEADER</code>	This macro is replaced with the entire received mail's header.

Table 2: Macros available in ravmd.

These macros are combined in one warning string that might look like this:

"The file `ATTACH_NAME` attached to mail (with subject:`SUBJECT`) sent by `FROM_USER` to `TO_USER(S)` is infected with virus: `VIRUS_NAME`".

Note: The macros `FROM_USER` and `VIRUS_NAME` can be used in the warning mail's subject too.

In this case, one attachment file is virus-infected. Info is provided about the attachment's name, the subject of the mail containing the infected attachment (this is an optional info), the sender's name, the addressees' names and the virus name.

To eliminate any mystification, here are some lines about the differences between the most confusing two pairs of macros:

1. QUARANTINE_NAME vs. SAVED_FILE_NAME

QUARANTINE_NAME represents the name of the infected object saved to Quarantine as a result of using the *Move* or *Copy* action. The file is saved by **ravmd** in the Quarantine directory with the extension `.qto`. The files are encrypted, but starting with version 8.3.3 **ravav** can decrypt them.

SAVED_FILE_NAME represents the name of the infected object saved in the Quarantine folder as a result of using the parameters `save_infected=yes` and `save_suspicious=yes`. The file is saved by **ravmd** in the Quarantine directory in the `UNIXTIME-RAV{PID_OF_RAV_FILTER}` format, where `RAV_FILTER` is `ravexim`, `ravpostfix`, etc. For sendmail, the file is saved in the `UNIXTIME-df{MESSAGE-ID}` format, and for sendmail-milter the format is `UNIXTIME-RAV{MESSAGE-ID}`.

2. HEADER_RECEIVED vs. HEADER

The `HEADER` macro is replaced with the entire mail header, while `HEADER_RECEIVED` is replaced only by the lines beginning with "**Received:**". Therefore, the lines *Message-Id*, *X-Sender* and *X-Mailer*, for instance, are only included in the `HEADER` macro is used, but not if the `HEADER_RECEIVED` macro is used.

The following *Frequently Asked Questions* will hopefully help you better understanding this very interesting feature of **ravmd**.

FAQ 5: Warning messages not sent

Question: Why am I no receiving the warning messages when viruses are found?

Answer: Most probably you did not correctly configure the **ravms** mail account. Here are the excerpts from the documentation that should help you:

"Default values:

`ravms_name = ravms`

`on_host = the official name returned by the gethostbyname() function`

`smtp_server = same as host (IP address)`

`smtp_port = 25`

`ravms_full_name = displayed.name (RFC822)`

Default values are provided and they will probably work. Define these fields only if warning mails are sent when a virus is found or if you want to use a different account instead of **ravms**. Specify `smtp_server` IP address only if that machine is behind a firewall and **ravmd** can't get its host name".

FAQ 6: Omitting the SUBJECT from warning mail

Question: Is it possible to define that SUBJECT should be omitted from the warning for specific viruses? Some of the viruses disclose potentially sensitive information from the victim's hard drive, and puts it in the subject of the email.

Answer: In `${ETCDIR}/languages/your_language` edit the following line:

```
infected_m_english = "The file ATTACH_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S)is infected with virus: VIRUS_NAME."
```

and remove "with subject: SUBJECT".

The subject will not be displayed anymore in your warning mails. Please note that this solution will be applied to all mails, not just for specific viruses.

FAQ 7: Changing the warning message in case of attachment match

Question: When a mail with .exe, .com or other attachments of this type is sent, a warning message is also sent, which is OK, but it says the file is infected with the virus: UNAUTHORIZED_MAIL_ATTACHMENT, which seems to scare the average user. Can the message be modified?

Answer: Yes, the message can be modified:

- In the `${ETCDIR}/languages/your_language` file define the variable:
`infected_m_attach_english = "The file ATTACH_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S) has an unauthorized file extension."`
- In section `_attachment_filter_warning_messages` of `${ETCDIR}/languages/your_language.equiv` file use: `infected_msg=infected_m_attach_english`
- Finally, restart ravmd with: `kill -HUP `cat ${DATADIR}/run/ravmd.pid``

The same procedure can be applied for UNAUTHORIZED_MAIL_SUBJECT and UNAUTHORIZED_MAIL_CONTENT warning messages, making the proper adjustments, as described in [FAQ 8](#) and [FAQ 9](#) below.

FAQ 8: Changing the warning message in case of subject match

Question: How can I change the warning message sent by **ravmd** in case of subject match?

Answer: Here is how you can do it:

- In the `${ETCDIR}/languages/your_language` file define the variable:
`infected_m_subject_english = "The mail with subject:SUBJECT sent by FROM_USER to TO_USER(S) has an unauthorized subject."`
- In the `_subject_filter_warning_messages` section of `${ETCDIR}/languages/english.equiv` use: `infected_msg=infected_m_subject_english`
- Restart ravmd with: `kill -HUP `cat ${DATADIR}/run/ravmd.pid`` and you're done.

FAQ 9: Changing the warning message in case of message's body content match

Question: How can I change the warning message sent by **ravmd** in case of content match?

Answer: In `${ETCDIR}/languages/your_language` define the variable:

`infected_m_content_english = "The mail with subject:SUBJECT sent by FROM_USER to TO_USER(S) has an unauthorized content."`

- In the `content_filter_warning_messages` section of `${ETCDIR}/languages/english.equiv` use:
`infected_msg=infected_m_content_english`
- Restart **ravmd** with: `kill -HUP `cat ${DATADIR}/run/ravmd.pid``

FAQ 10: Offering information about viruses in the warning mails

Question: How can I add a link to a web page where users can easily find the information regarding the virus generating a warning mail?

Answer: Add the following link in your `${ETCDIR}/languages/<your_language_file>` file in the message for infected files:

[http://www.ravantivirus.com/virus/by-keyword.php?k=\\$VIRUS_NAME&sourceid=WM](http://www.ravantivirus.com/virus/by-keyword.php?k=$VIRUS_NAME&sourceid=WM)

Now, for every infected file in the warning mail you will receive the direct link for the viruses that can be found in our **Virus Encyclopedia**. If the virus generating the warning mail cannot be found, the link will return the main page of the Virus Encyclopedia.

FAQ 11: Stopping update notifications

Question: How do I stop update notifications?

Answer: In order to stop the update notifications, in `${BINDIR}/ravmdupdate.sh` change:

`#Specify when should the administrator be notified by the update process`

`#VERBOSE="silent"`

`VERBOSE="noisy"`

`#VERBOSE="errors"`

with:

`#Specify when should the administrator be notified by the update process`

`#VERBOSE="silent"`

`#VERBOSE="noisy"`

`VERBOSE="errors"`

or with:

`#Specify when should the administrator be notified by the update process`

`VERBOSE="silent"`

`#VERBOSE="noisy"`

`#VERBOSE="errors"`

Antispam Definitions section

The **Antispam** module is available in **RAV AntiVirus for Mail Servers** starting with version 8.4.0. This feature is designed to protect the users of unsolicited mail messages.

The parameters pertaining to the antispam module of **RAV AntiVirus for Mail Servers** are described in the [Antispam parameters](#) sub-section of this document.

The Antispam functionality is based on the bulk mail module available in RAV Engine version 8.9 or later. This module is working in close co-operation with older features like **Real-time Blackhole List (RBL)** or **White-Black List (WBL)**, available in **ravmd** since version 8.3.3.

How does it work

When a mail message reaches a RAV-protected mail server, it is first confronted with the **White/Black List (WBL)**. This is a static configurable list that any system administrator can use for specifying the mail addresses from which he wants to automatically *accept* messages (**Static White List**) or the mail addresses from which he wants to automatically *reject* or *discard* messages (**Static Black List**).

If the mail just received by the RAV-protected mail server comes from an e-mail address found in the **Static White List**, then the search in the **RBL** is not executed anymore and **ravmd** jumps directly to the scanning process for the respective mail. Also, the antispam module is not used for the mail coming from addresses found in the **Static White List**. If the mail just received by the RAV-protected mail server comes from an e-mail address in the **Static Black List**, the mail is automatically *rejected* or *discarded* (according to the specified settings).

If the sender's e-mail address is not found in the **White/Black List (WBL)**, the mail is confronted against the **Real-time Blackhole List (RBL)**. This is a dynamic list (**rbl_site**) with sites containing listings of known spammers. If any of the IP addresses from the mail's header is listed on one of the websites defined in the **rbl_site**, the mail is automatically rejected. If no IP address from the mail's header is found in the **rbl_site** list or the **use_rbl** parameter is set to **No**, **ravmd** jumps to the scanning process for the respective mail. The system administrator can of course choose not to use these **WBL** and **RBL** features. In this case, **ravmd** scans directly the received messages, using the options defined for the corresponding groups.

Assuming the mail is passing OK through the virus-scanning process (i.e. the message is clean or **ravmd** has cleaned it), the antispam search is executed. **ravmd** is looking for patterns known to be specific to spammers on the **Header** and **Body** levels of the mail message. **ravmd** applies the same antispam criteria (about 500, at this writing) for each mail message confronted with the antispam module. Depending on its specifics, the message receives a number of points and is tagged as **Spam** or **No spam** message. Depending on the number of points they receive, the mail messages *tagged as spam* are classified in one of the following *accuracy levels*: **Low**, **Medium**, **High** and **Very High**.

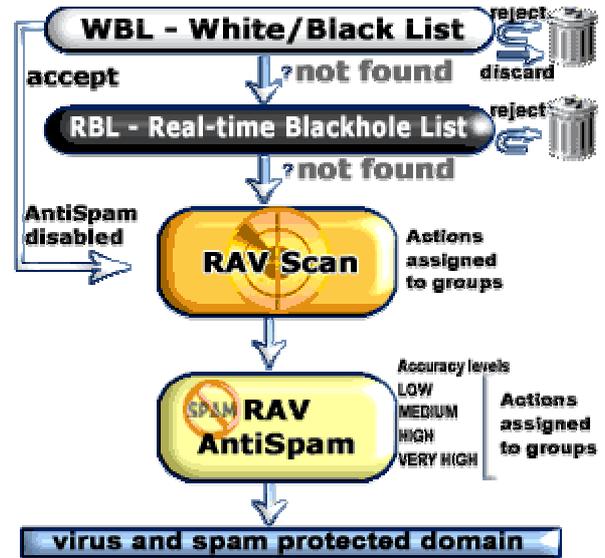
A „low“ accuracy level means that **ravmd** reports the suspected mail message as spam even if only few spam patterns exist. This means that you might get some „false alarms“, but no spam messages will pass by **ravmd**.

A „medium“ accuracy level means that **ravmd** reports the suspected mail message as spam if several spam patterns exist. This means that the number of „false alarms“ is lower than in case of low accuracy spam, but several spam messages might pass by **ravmd**.

A „high“ accuracy level means that **ravmd** reports the suspected mail message as spam if more spam patterns exist. This means that the number of „false alarms“ is low, but some spam messages might pass by **ravmd**.

A “very high” accuracy level means that **ravmd** reports the suspected mail message as spam only if lots of spam patterns exist. Therefore, it is unlikely for **ravmd** to report false alarms, but more spam messages will probably pass by **ravmd**.

Below you can find a scheme detailing the flow described above.



For each of the four accuracy levels specified above you can configure different strings and separate actions to be used/taken by **ravmd**. As mentioned in the [corresponding](#) table, the actions specific to the Antispam module of **ravmd** are: **save**, **embed**, **add_header**, **add_subject**, **reject**, **discard**, **deliver**, **forward**. All these actions are documented below:

- **save** (the mail is saved in the **Quarantine** folder, for further analysis, before any other action is executed on the respective mail);

Note: The **Quarantine** folder for bulk mails is different than the **Quarantine** folder for suspicious files.

- **embed** (the mail tagged as spam is sent as embedded mail);

Note: When using this feature of **ravmd**, the messages tagged as spam are reinjected in the MTA queue using the following syntax:

```
cat <modified_mail> | sendmail -i -f<sender> <receivers>
```

(you must have 'sendmail' executable in your search path for commands - environment variable PATH).

If the sendmail binary is not corresponding to the binary of your MTA, then you should make the link manually in /usr/sbin (for example) by using (example for Qmail MTA):

```
ln -sf /var/qmail/bin/sendmail /usr/sbin/sendmail
```

This command assumes that you have the Qmail's binaries in /var/qmail/bin.

- **add_header** (a header is added to the mail tagged as spam);
- **add_subject** (a user-defined variable is affixed in the **Subject** field of the mail tagged as spam);
- **reject** (the mail tagged as spam is rejected);
- **discard** (the mail messages tagged as spam are discarded, i.e. rejected with no bounce back to the sender);

- **deliver** (the mail messages tagged as spam are delivered to their recipients);
- **forward** (the mail messages tagged as spam are forwarded to the addresses specified by the `forward_to` parameter).

A default configuration is included in the `${ETCDIR}/antispam` file. Here you should define your rules and the actions to be taken (for each accuracy level), and then you should include in each group's configuration file the following parameter:

```
antispam_configuration = rule_name_1, rule_name_2, rule_name_3, rule_name_4
```

where `rule_name_x` is the rule name you defined in the **antispam** file.

Here is the checklist to be followed when using the antispam module of **ravmd**:

- define the name of the section, flanked by two "@" symbols (i.e. `@bulk_detection_low@`) in `ravmd.conf` (after the **Regular Expression Declarations**, **Action Definitions** and **Warning messages** sections and before the **Group configuration** section);
- Define the accuracy level. This should be one of the following four keywords: `accuracy_low`, `accuracy_medium`, `accuracy_high`, `accuracy_very_high`;
- define the strings that will be used by the `extra_header`, `extra_subject` and `embedded_msg` parameters;
- define the actions to be executed for each of the four bulk detection levels;
- define the path to the **Quarantine** folder where bulk mails are saved when using the **Save** action.

Important: If no action is defined for one specific accuracy level, **ravmd** will automatically assume the actions defined for the level having the immediate lower priority.

Here is one example for Linux:

```
@bulk_high_precision@
accuracy_high
extra_header = bulk_header_high_english
extra_subject = bulk_subject_high_english
embedded_msg = bulk_embedded_high_english
actions = bulk_actions_high
quarantine = ${DATADIR}/bulk
```

Then, in the groups you want to use the actions you just defined to mails tagged as spam with level accuracy "high", include the following line:

```
antispam_configuration = bulk_high_precision
```

For more details, please refer to the corresponding [Antispam parameters](#) sub-section of this document. The `bulk_header_high_english`, `bulk_subject_high_english` and `bulk_embedded_high_english` (and the other similar parameters) are to be defined in the selected language files (i.e. `${ETCDIR}/languages/english`) and the `bulk_actions_high` parameter - in the `_define_actions` section of `ravmd.conf` or in the `actions` file.

Group Declarations

What is a group

A group is a category of users (senders or receivers) having different mail addresses and/or domains, but share the same configuration parameters for **ravmd**. Using the group feature of **ravmd**, you can use the same characteristics (for the scanning engine or the update process, for instance) and the same actions (filters, warning mails and so on) for users having different mail addresses and mail domains. This group structure is very useful in case you use **ravmd** in a network with hundreds of mailboxes.

In **ravmd** you have one default group, called `[global]`, containing at the beginning all the users and mail domains protected by **ravmd**. When you define a new group, its members are taken away from the `[global]` group and included in the new group, for which you have to define new group-specific options.

Warning: Make sure when defining groups not to include the same user in different groups.

The `[global]` group

This is the default group, which contains all the users and mail domains that are not defined in the other groups. The `[global]` group **MUST NOT** contain any member declaration parameters.

Defining other groups

A new group definition begins with the group name written between square brackets "[]". The group definition must be followed by the member declarations (it is mandatory that the members are declared before any other parameters) and the group options.

THE `_include` DIRECTIVE

In order to keep the configuration file more readable you can use the `_include` directive to insert other files in the main one. This can be very useful if you have a large number of groups. Defining them on a single configuration file will make the future maintenance difficult. Instead of using a single large configuration file, you can split it in more small files.

If you want to add a new group called `[mygroup]`, all you have to do is to append the following line to the main configuration file:

```
_include ${ETCDIR}/groups/mygroup_file
```

Then define the group in the `mygroup` configuration file:

```
[mygroup]
sender = user_1@domain.com
```

If you want to add another group called `[yourgroup]`, append to the main configuration file:

```
_include ${ETCDIR}/groups/yourgroup_file
```

Then define that group in the `yourgroup` configuration file:

```
[yourgroup]
sender = user_2@domain.com
```

Note: You can use the `_include` directive to include as many groups you wish in the main configuration file. However, you should keep in mind that `ravmd` will apply for each mail only the rules defined for the first matched group using the `_include` directive. For instance, when analysing a mail with multiple recipients, one in `mygroup` and one in `yourgroup`, `ravmd` will apply the actions defined for the first matched group (`mygroup` in this case).

How do I configure groups

The configuration file must include the `[global]` group, containing the default options for all mail scanning processes. Besides the `[global]` group, you can customize `ravmd` by creating additional groups, with different configurations.

If no value is specified for one parameter in the configuration file for one group, `ravmd` will use the *default value* for that parameter. If no *default value* is defined, `ravmd` will use the value specified in the `[global]` group for that parameter, with the following exceptions: `filter_subject`, `filter_attachment`, `filter_content`, `warn_sender`, `warn_receivers`, `warn_admin`, `do_not_warn`, `do_not_show`, `admin_addr`, `do_not_scan`, `cf_do_not_scan_extensions`, `advertising_msg`, `wbl_reject`, `wbl_accept`, `wbl_discard`, `use_rbl`, `embed_clean_mail`, `embed_cleaned_mail`, `embed_unclean_mail`, `use_embedded_msg`, `use_embedded_warning`, `antispam_configuration`.

How do I configure separate antispam options for my groups

If you intend to use the antispam feature of `ravmd`, you should specify what *antispam configuration* you want to use for the targeted groups. You do that using the `antispam_configuration` parameter, described in the [Group-specific parameters](#) section of this *User Guide*.

The following *Frequently Asked Questions* are providing the answers for some of the confusions of the existing users of **RAV AntiVirus for Mail Servers** in aspects pertaining to group configuration.

FAQ 12: Creating different rules for a domain

Question: How do I create different rules for a domain?

Answer: You can define different rules for a domain by creating a group in which you specify `from_host` and/or `to_host`. Then specify the actions that `ravmd` can perform for the newly created domain. Here are the steps you should follow for creating different rules for a domain under Linux, Solaris and MacOS X. For BSDs, the process is identical, except for the paths to the `my_domain` file:

- In the actions file use:
`act_for_my_domain = clean, delete, ..`
- At the end of `ravmd.conf` define the group:
`[my_domain]`
`_include ${ETCDIR}/groups/my_domain`
- Create the `${ETCDIR}/groups/my_domain` file and edit it:

```
#from_host=a.com
to_host=a.com
infected_actions=act_for_my_group
```

FAQ 13: Example on how to set the domains and the IP addresses

Question: I need an example on how to set the domains and the IP addresses.

Answer: In the [global] section of ravmd.conf use:

```
domain = localhost, your.host.com
```

In order to receive the warning mails, in the `${ETCDIR}/groups/global` file use:

```
ravms_name = ravms
on_host = your.host.com
smtp_server = ip.address.of.host
ravms_full_name = displayed.name (RFC822)
```

FAQ 14: Global group configuration

Question: How is the [global] group configured?

Answer: Please define in the `${ETCDIR}/groups/global` file the following options:

```
ravms_name=ravms
on_host=your.host.com
smtp_server=127.0.0.1 (IP address)
smtp_port=25
ravms_full_name = displayed.name (RFC822)
```

Then start **ravmd** with `/etc/init.d/ravmail start` for Linux, Solaris and MacOS X and with `/usr/local/etc/rc.d/ravmail.sh` for BSDs.

Note: In `/etc/hosts` you should have defined at least: `127.0.0.1 localhost.localdomain localhost`

FAQ 15: Excluding one particular account

Question: I need to exclude one account in particular as I receive a daily email of over 60 MB in size and I don't want RAV to try to process that file. What do I get it done?

Answer: You can create a group with that sender and choose not to scan that mail. To do that, in `ravmd.conf` add at the end of the file:

```
[group1]
_include ${ETCDIR}/groups/group1
```

Then create the file `${ETCDIR}/groups/group1` and edit it:

```
sender=sender@host.com
do_not_scan=yes
```

Then restart `ravmd` with: `kill -HUP `cat ${DATADIR}/run/ravmd.pid``

The Advanced Content Filtering feature

A common mail message is considered to have the following components: **Subject**, **Body** and **Attachments**. However, when talking about **ravmd**'s content filtering module, things are a little bit different. Specifically, the **Body** level consists of the mail's body and the attachments' contents, while the **Attachment** level is only represented by attachments' names.

What is the Advanced Content Filtering feature of ravmd

The **Advanced Content Filtering** is a very powerful feature of **ravmd** that can help you configure the product to answer your specific needs. Using this feature, you can define **filters** on the following levels:

- Subject
- Body
- Attachment name and
- Attachment contents.

You can afterwards define **specific actions** for the mail messages matching the rules you specify. For instance, you can instruct **ravmd** to deny incoming mail messages containing one specific string in the **Subject** field (i.e. "I love you"), deny outgoing mail messages containing user-specified strings (i.e. "confidential", "balance sheet") and deny incoming/outgoing mail containing attachment with user-specified file types/names.

How does it work

The *Content Filtering* module of **ravmd** is using:

- **POSIX regular expressions** for searches executed on the *Subject* and *Attachment* (attachment file names) levels;
- **User-defined strings** for searches executed on the *Body* level - message's body and the attachments contents.

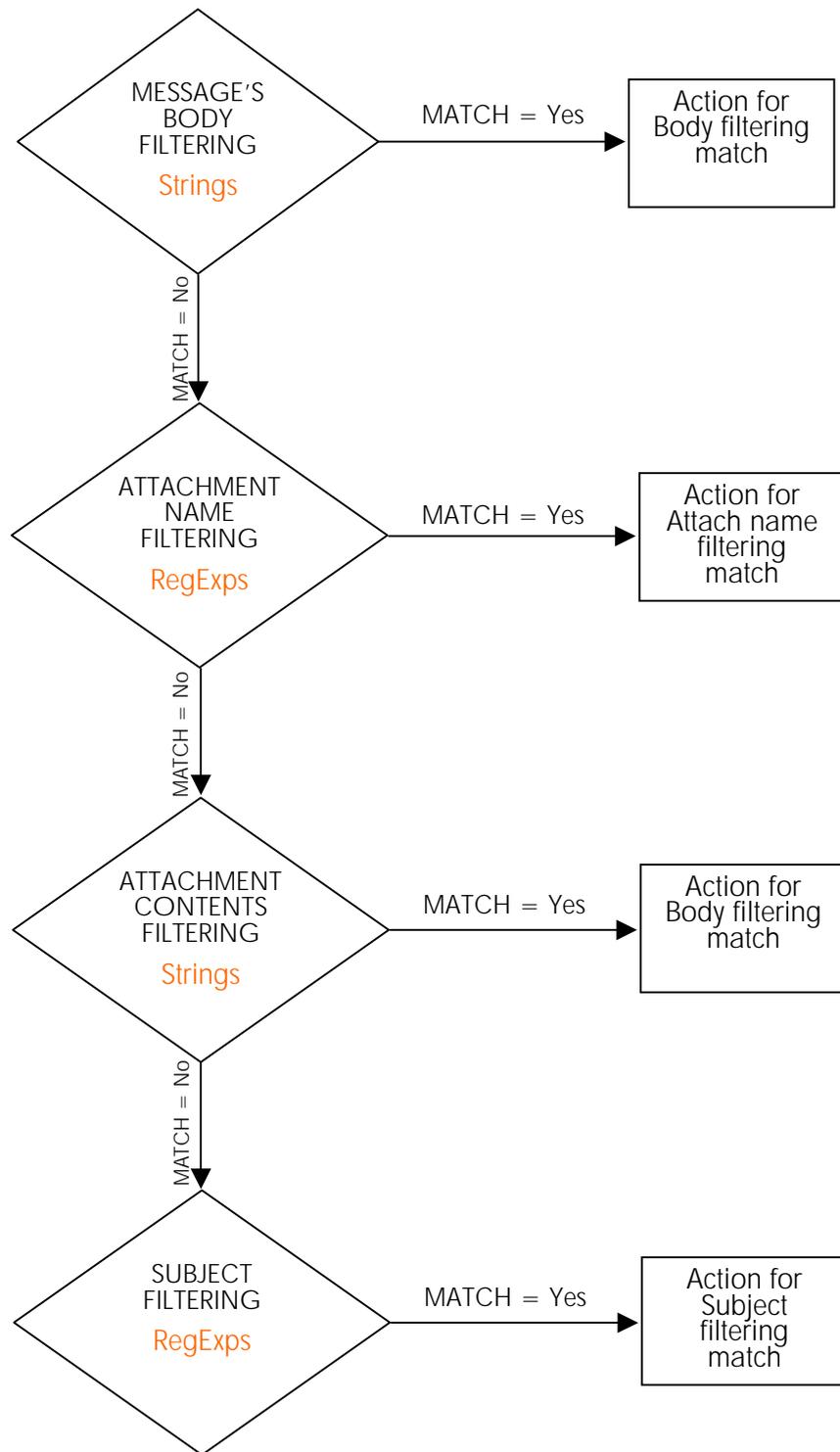
Note: The user-defined string-based body filtering is only used starting with version 8.4.0 of **ravmd**; previous versions of **ravmd** have used POSIX regular expressions for body filtering.

The rules you define are processed in the following order:

- Message body,
- Attachment file names,
- Attachment contents,
- Subject.

If you define more rules for one single component, the rules will be processed in the order you are defining them.

If more than one component are matched by a body filtering rule, for instance, the action defined by the system administrator is executed for all the matched components.



The scheme above is illustrating the different content filtering modules of **ravmd** and the way they basically work (using POSIX regular expressions or user-defined strings).

Here are some *Frequently Asked Questions* users of **RAV AntiVirus for Mail Servers** have asked about the *Advanced Content Filtering* feature of **ravmd**. You can even find an example of configuring a content filter for mail messages containing one specific string.

FAQ 16: Rejecting double extension files

Question: Has anyone setup the configuration file to reject any attachment with a double extension?

Answer: Please use in your `regexp` file from your `${ETCDIR}` directory:

```
var_regexp = .*\..*\.*
```

Then define the action to be taken by **ravmd** in your `actions` file from your `${ETCDIR}` directory:

```
var_action = reject
```

In all the groups for which you want to reject the messages containing double extension attachments use:

```
filter_attachment var_regexp var_action
```

Please note that this restrictive action will also filter the `.tar.gz` files, for instance.

FAQ 17: Denying certain attachment extensions

Question: How can I deny incoming messages containing attachment with extensions known to be dangerous?

Answer: The attachment name filtering feature of **ravmd** can help you if you want to deny incoming messages containing attachment with specific extensions. The attachment name filtering module of **ravmd** is using POSIX regular expressions.

Here is an example for how to deny `.exe` attachment files:

Please use in the `regexp` file from your `${ETCDIR}` directory:

```
file_regexp = .*\.exe
```

In the `actions` file from your `${ETCDIR}` define:

```
file_action = delete, reject
```

In all the groups for which you want to reject the messages containing `.exe` attachments use:

```
filter_attachment file_regexp file_action
```

To add other types of attachments separate them with a `|` symbol in the `regexp` definition.

Example:

```
file_regexp = .*\.((vbs)|(vbe)|(js)|(exe)|(com)|(pif)| (lnk)|(scr)|(bat)|(shs)|(sh))
```

As is the case for each change in the configuration files (`ravmd.conf`, `global`, `english`, `english.equiv`), you must restart **ravmd**:

```
kill -HUP `cat ${DATADIR}/run/ravmd.pid` .
```

FAQ 18: Example of subject filtering

Question: Can I have an example of subject filtering for mail messages containing one specific expression in their **Subject** field?

Answer: Yes, you can. Here you can find how to create a content filtering rule for mail subjects containing the "xxx" expression. Please note that this feature is not available in **ravsendmail**.

The following options are available for the mail messages matching the subject filtering rule: **copy, ignore, reject, discard**.

For this scenario, the mail will be allowed to pass **RAV AntiVirus for Mail Servers**. The sender and the receiver will not receive warnings, but a warning will be sent to the administrator.

- In the `regexp` file from your `${ETCDIR}`, define the `xxx` expression:
`subjxxx_regexp = xxx`
- Define the action for `xxx` in the actions file:
`define xxxsubj_action = ignore`

This will allow the `xxx` mail to pass the content filtering module of **ravmd**.

- To add other words, separate them with a `|` symbol in the `regexp` definition.

Example:

```
subjxxx_regexp = (word1)|(word2)|(word3)
```

- Edit the file `${ETCDIR}/groups/global` file.
- Specify the administrator's mail address for alerting him that a mail matching the subject filtering rule is entering the company:
`admin_addr=administrator@yourcompany.com`
- Activate the subject filtering rule by adding the following line:
`filter_subject subjxxx_regexp xxxsubj_act`

As is the case for each change in the configuration files (`ravmd.conf`, `global`, `english`, `english.equiv`), you must restart **ravmd**:

```
kill -HUP `cat ${DATADIR}/run/ravmd.pid`.
```

FAQ 19: Example of body filtering

Question: Can I have an example of body filtering for mail messages containing one specific expression in their bodies?

Answer: Here is one example from which you can find how to create a body filtering rule for mail messages containing the following expressions in their bodies: "confidential", "salaries" and "balance sheet".

For this scenario, the mail messages matching the body filtering rule will not be allowed to pass **ravmd**. The sender and the receiver will not receive warnings, but a warning will be sent to the administrator.

- In the `regexp` file from your `${ETCDIR}`, define the string `confidential`:
`bodyconfidential_string = confidential`
- To add other words, separate them with a `|` symbol on the string definition. *Example*:
`bodyconfidential_string = confidential|salaries|balance sheet`
- Define the action for `confidential` in the `actions` file from your `${ETCDIR}`:
`bodyconfidential_action = reject`

This will not allow the mail messages containing to pass the content filtering module of `ravmd`.

- Edit the file `${ETCDIR}/groups/global` file.
- Specify the administrator's mail address for alerting him that a mail matching a body filtering rule is entering the company:
`admin_addr=administrator@yourcompany.com`
- Activate the content filter by adding the following line:
`filter_subject bodyconfidential_string bodyconfidential_action`

As is the case for each change in the configuration files (`ravmd.conf`, `global`, `english`, `english.equiv`), always remember to restart `ravmd`: `kill -HUP `cat ${DATADIR}/run/ravmd.pid``.

FAQ 20: Digging even deeper in string-based body filtering

Question: I still have some problems with body filtering. Can you give me more details?

Answer: Below you can find some examples that should help you.

Example 1

To filter **complete words**, please follow the steps described below:

- Insert the following line in your `${ETCDIR}/regexp` file:

```
custom_body_string = | word1 | word2 |
```

Please note that there is a `<SPACE>` character before *and* after each of these words:

```
| <SPACE>word1<SPACE> | <SPACE>word2<SPACE> |
```

Note: `<SPACE>` is the ASCII character with decimal code `#32`.

- Insert the following line in your `${ETCDIR}/actions` file:

```
custom_body_action = discard
```

- Insert the following line in your `${ETCDIR}/groups/global` file:

```
filter_content custom_body_string custom_body_action
```

This will filter all occurrences of the strings ' word1 ' or ' word2 ', as in the following example:

Content filtering example using word1 and word2 strings

Example 2

To filter words at the **beginning/end** of a text string, please follow the steps described below:

- Insert the following line in your `${ETCDIR}/regexp` file:

```
custom_body_string = |word1 | word2|
```

Please note again the `<SPACE>` character:

```
|word1<SPACE> |<SPACE>word2|
```

Note: `<SPACE>` is the ASCII character with decimal code #32.

- Insert the following line in your `${ETCDIR}/actions` file:

```
custom_body_action = discard
```

- Insert the following line in your `${ETCDIR}/groups/global` file:

```
filter_content custom_body_string custom_body_action
```

This will filter all the occurrences of string 'word1 ' or 'word2 ', as in the following example:

Content filtering example using_word1 and word2_strings

The filter from the first example cannot filter the body string described above.

Example 3

To filter **any occurrence** of any word in a string, follow the steps described below:

- Insert the following line in your `${ETCDIR}/regexp` file:

```
custom_body_string = |word1|word2|
```

Please note again the `<SPACE>` character:

```
|word1<SPACE> |<SPACE>word2|
```

Note: `<SPACE>` is the ASCII character with decimal code #32.

- Insert the following line in your `${ETCDIR}/actions` file:

```
custom_body_action = discard
```

- Insert the following line in your `${ETCDIR}/groups/global` file:

```
filter_content custom_body_string custom_body_action
```

This will filter all occurrences of 'word1' and 'word2', as in the following examples:

Content filtering example_using_word1_and_word2_strings

or:

ContentFilteringExampleUsingWord1AndWord2Strings

Explaining the parameters

The parameters included in this configuration file for **ravmd** (`ravmd.conf`) have different functions. Some are used for specifying the domain parameters, other for specifying the group members, the actions for the scanning engine, the warning messages and the sender of warning messages and so on. All the parameters have been grouped below depending on their function.

Some of these parameters are group-specific, meaning that they are different for each defined group. Other parameters are common for different groups and/or are inherited from the `[global]` group. This classification is also important for understanding the way **ravmd** is working.

Domain parameters

domain = enumeration

Description: This parameter specifies the domains scanned by **RAV AntiVirus**.

Note: During the evaluation period of 30 days you can set only two domains. Mails to/from other domains are delivered normally, without being scanned.

This parameter is a global one. It is sufficient to define it in the `[global]` group.

Important: You **must** specify at least one domain name, or else **ravmd** will not start. Starting with the 8.3.3 version of **ravmd**, the following default groups are provided: `machine.name` and `domain.name`.

Example:

```
domain = mail.domain.com, domain.com
```

```
domain = second.domain.net
```

Important: In the acceptance of **ravmd** and according to the **License Agreement**, a 'domain' is defined as "the string which follows the '@' symbol in a mail address".

Group members

sender = enumeration

Description: Parameter used for specifying the mail addresses of the senders who will be members in the actual group.

Example:

```
sender = user1@domain1.com, user2@domain1.net
```

```
sender = user3@domain2.org
```

receiver = enumeration

Description: Parameter used for specifying the mail addresses of the receivers who will be members in the actual group.

Example:

```
receiver = user4@domain4.com, user5@domain4.org
```

```
receiver = user5@domain2.org
```

from_host = enumeration

Description: Parameter used for specifying the hosts that are members in the actual group.

Example:

from_host = mail.domain1.com, domain1.net

from_host = domain2.org

to_host = enumeration

Description: Parameter used for specifying the receiving host names members in the actual group.

Example:

to_host = domain3.com, domain3.net

to_host = domain3.com

Engine actions

The following parameters are used to define the actions for the scanning engine: **infected_actions**, **suspicious_actions**. The **infected_actions** parameter helps you define the actions to be performed when an infected object is found, and the **suspicious_actions** parameter - the actions to be performed when a suspicious object is found.

infected_actions = variable

Description: Parameter used for specifying a variable name defined in the **actions** file, which contains the actions to be performed when an infected object is found. If this parameter is not defined then the *reject* action is used for the infected mails.

Example:

```
infected_actions = act_for_infected_files
```

Where:

```
act_for_infected_files = clean, move, delete, reject, discard
```

In this example, when an infected file is detected, **ravmd** tries first to *clean* that file. If the cleaning action is completed successfully, **ravmd** moves to the next file. If the cleaning action fails, **ravmd** tries to *move* (*copy* to quarantine and *delete* that file from mail) the file. If the moving action is completed successfully, **ravmd** moves to the next file. If the moving action fails, **ravmd** tries to *delete* the file, and so on. The *ignore*, *reject* and *discard* actions **always** return success.

Note: **ravmd** appends a *reject* action to the actions enumeration by default.

suspicious_actions = variable

Description: Parameter used for specifying a variable name defined in the **actions** file, which contains the actions to be performed when a suspicious object is found. If this parameter is not defined then the *reject* action is used for the suspicious mails.

Example:

```
suspicious_actions = act_for_suspicious_files
```

Where:

```
act_for_suspicious_files = move, rename, delete, ignore
```

As in the previous example, when **ravmd** detects a suspicious file, it tries successively to *move*, *rename* and *delete* that file. If the move action fails, **ravmd** tries to rename the file (see below the description for **rename_ext**). If the *rename* action fails, **ravmd** tries to *delete* the file. If one of the previous actions is successfully completed, **ravmd** moves to the next file. Eventually, if all of the actions (*move*, *rename*, *delete*) fail, **ravmd** will ignore that file.

Engine parameters

use_heuristics = boolean

Description: This parameter is used to control the heuristic methods for detecting new viruses.

Default value: **Yes**.

Example:

```
use_heuristics = no
```

use_cf_inside_embedded_objects = boolean

Description: Use this parameter to specify if **ravmd** will use the content filtering feature for embedded objects (i.e. files included in archives, scripts inside HTML files, OLE objects).

Default value: **Yes**.

Example:

```
use_cf_inside_embedded_objects = no
```

Note: Setting the `use_cf_inside_embedded_objects` to **No** will determine **ravmd** to exclude files inside attachments from the content filtering search.

cf_do_not_scan_extensions = enumeration

Description: Use this parameter to instruct **ravmd** not to apply the content filter feature to the specified attachment file types.

Default value: **-not defined-**.

Accepted values: Valid file extensions (preceded by dot and separated by blank spaces and/or commas) plus **All** (if the `cf_do_not_scan_extensions` parameter is set to **All**, all the attachment files will be excluded from content filtering).

Example 1:

```
cf_do_not_scan_extensions = .jpg .jpeg .gif .mpeg
```

Example 2:

```
cf_do_not_scan_extensions = All
```

scan_packed_executables = boolean

Description: This parameter is used to control the scanning process for packed executables (i.e. `vpack`, `ucexe`, `pepack`, etc.)

Default value: **Yes**.

Example:

```
scan_packed_executables = no
```

scan_archives = boolean

Description: This parameter is used to control scanning in archive files, like **ZIP**, **ARJ**, **RAR**, **LHA**, **LHZ**, **ACE**, **CAB**, etc. If the Boolean value is set to **Yes**, **ravmd** will scan inside archives.

If the Boolean value is set to **No**, **ravmd** will not scan inside archives.

Default value: **Yes**.

Example:

```
scan_archives = yes
```

rename_ext = string

Description: Parameter used for specifying the *extension* used to *rename* the infected/suspicious files. This function is necessary for preventing inexperienced users from accessing by accident infected/suspicious files moved to Quarantine.

Default extension is: `_??`

Example:

```
rename_ext = _??
```

smart_scan = boolean

Description: Parameter used for specifying the scanning modes for **ravmd**. The available options are:

- scan all files,
- let **ravmd** decide what files to scan.

Note: If `smart_scan` is not defined, then **ravmd** will scan ALL files. By default, the smart scanning is enabled.

Example:

```
smart_scan = yes
```

Warning messages

The following parameters help you define the messages used to create warning mails in different circumstances (virus found, subject filtering match, attachment name filtering match, content filtering match, etc). If some warning messages are not specified for the [global] group, then **ravmd** uses a default string (<<not defined>>). For all the other groups, the warning messages are inherited from the [global] group.

[_virus_warning_messages](#)

Description: Section used for specifying the contents of the warning mails sent by **ravmd** when a *virus* is found in one mail message.

[_subject_filter_warning_messages](#)

Description: Section used for specifying the contents of the warning mail sent by **ravmd** when the content filtering is enabled and the search in the mail's **Subject** field has yielded a *match* with one of the user-defined rules.

[_attachment_filter_warning_messages](#)

Description: Section used for specifying the contents of the warning mail sent by **ravmd** when the content filtering is enabled and the search in the mail's **Attachment's name** field has yielded a *match* with one of the user-defined rules.

[_content_filter_warning_messages](#)

Description: Section used for specifying the *content* of the warning mail sent by **ravmd** when the content filtering is enabled and the search in the mail's **Body** or **Attachment** has yielded a *match* with one of the user-defined rules.

[warning_mail_subj = variable](#)

Description: Parameter used for specifying the *subject* of the warning mails.

Example:

```
warning_mail_subj = wm_sbj
```

Note: In the **Warning Mails Message Declarations** section you must specify: `wm_sbj = "RAV AntiVirus scan results."`

[infected_msg = variable](#)

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when an *infected* file is detected.

Example:

```
infected_msg = wm_inf_msg
```

Where:

```
wm_inf_msg = "The file ATTACH_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S) is infected with virus: VIRUS_NAME."
```

suspicious_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a *suspicious* file is detected.

Example:

suspicious_msg = wm_sus_msg

Where:

wm_sus_msg = "The file ATTACH_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S) contains suspicious code."

ignored_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when an *infected/suspicious* mail file is ignored.

Example:

ignored_msg = wm_ign_msg

Where:

wm_ign_msg = "All the defined actions have failed. Do not use this file."

rejected_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when an *infected/suspicious* mail file is rejected.

Example:

rejected_msg = wm_rej_msg

Where:

wm_rej_msg = "The mail was rejected because it contains dangerous code."

discarded_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when an *infected/suspicious* mail file is discarded. **Discard** is a new action available starting with **ravmd** version 8.3.3.

The value for **variable** must be declared in the language file and the `language.equiv` file must be included in the group file. So, in the `/${ETCDIR}/languages/english.equiv` file, define:

```
_virus_warning_messages
discarded_msg=discarded_m_english
.....
_subject_filter_warning_messages
discarded_msg=discarded_m_english
.....
```

_content_filter_warning_messages
discarded_msg=discarded_m_english
_attachment_filter_warning_messages
discarded_msg=discarded_m_english

Example:

discarded_msg = discarded_m_english

Where:

discarded_m_english = "This mail was discarded. Please contact your system administrator."

cleaned_msg = variable

Description: Parameter used for specifying the *body* of the info mail sent by **ravmd** when a file is *cleaned*.

Example:

cleaned_msg = clean_ok

Where:

clean_ok = "The file was successfully cleaned by RAV AntiVirus."

moved_msg = variable

Description: Parameter used for specifying the *body* of the info mail sent by **ravmd** when a file is *moved*.

Example:

moved_msg = move_ok

Where:

move_ok = "The file was successfully moved to quarantine with name: QUARANTINE_NAME."

copied_msg = variable

Description: Parameter used for specifying the *body* of the info mail sent by **ravmd** when a file is *copied*.

Example:

copied_msg = copy_ok

Where:

copy_ok = "The file was successfully copied to quarantine with name: QUARANTINE_NAME."

deleted_msg = variable

Description: Parameter used for specifying the *body* of the info mail sent by **ravmd** when a file is *deleted*.

Example:

deleted_msg = delete_ok

Where:

delete_ok = "The file was successfully deleted by RAV AntiVirus."

renamed_msg = variable

Description: Parameter used for specifying the *body* of the info mail sent by **ravmd** when a file is *renamed*.

Example:

renamed_msg = rename_ok

Where:

rename_ok = "The file was successfully renamed by RAV AntiVirus."

saved_inf_msg = variable

saved_sus_msg = variable

Description: Parameters used for specifying the string used in the warning mail when an infected/suspicious file is *saved* to quarantine.

Example:

saved_inf_msg = save_ok

saved_sus_msg = save_ok

Where:

save_ok = "The mail file `SAVED_FILE_NAME` was saved to quarantine."

cannot_clean_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a file *cannot be cleaned*.

Example:

cannot_clean_msg = not_cleaned

Where:

not_cleaned = "Cannot clean this file."

cannot_move_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a file *cannot be moved*.

Example:

cannot_move_msg = not_moved

Where:

not_moved = "Cannot move this file."

cannot_copy_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a file *cannot be copied*.

Example:

```
cannot_copy_msg = not_copied
```

Where:

```
not_copied = "Cannot copy this file."
```

cannot_delete_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a file *cannot be deleted*.

Example:

```
cannot_delete_msg = not_deleted
```

Where:

```
not_deleted = "Cannot delete this file."
```

cannot_rename_msg = variable

Description: Parameter used for specifying the *body* of the warning mail sent by **ravmd** when a file *cannot be renamed*.

Example:

```
cannot_rename_msg = not_renamed
```

Where:

```
not_renamed = "Cannot rename this file."
```

cannot_save_inf_msg = variable

cannot_save_sus_msg = variable

Description: Parameters used for specifying the *string* used in the warning mail when the infected/suspicious file cannot be saved to quarantine.

Example:

```
cannot_save_inf_msg = not_saved  
cannot_save_sus_msg = not_saved
```

Where:

```
not_saved = "The infected mail file cannot be saved to quarantine."
```

supervisor_addr = enumeration

Description: Parameter available from version 8.4.2 of **ravmd** used for specifying a list (separated by commas) of mail addresses of the administrators to be notified:

- in case a **ravmd** malfunction is recorded; or

- with 14 days before your license will expire; or
- on each **ravmd** reload in the last 14 days before your license expires, until you extend your subscription period.

This parameter should be declared only in the `[global]` group.

Besides the addresses specified in the `supervisor_addr` parameter from the `[global]` group, warning mails are also automatically sent (in all the above-mentioned cases) to `postmaster@hostname` and `root@hostname`, where `hostname` is the value for the `on_host` parameter in the `[global]` group or the host name, as returned by the `gethostbyname()` function:

Default: No value.

Example:

```
supervisor_addr = postmaster@domain1.com, postmaster@domain2.net
```

`disclose_supervisor=boolean`

Description: Parameter used for specifying if the addresses of the supervisor will or will not be disclosed in the warning mail's **To:** header.

Default: **No**.

This parameter should be declared only in the `[global]` group.

Specifying the sender of the warning mails

`ravms_name = string`

`on_host = string`

`smtp_server = string`

`smtp_port = number`

`ravms_full_name = string`

Description: Using these parameters you can define the mail address for the sender of the warning mails. *Default values* are provided and they will probably work.

Define these fields only if warning mails are sent when a virus is found or if you want to use a different account instead of `ravms`. Specify the `smtp_server` IP address only if that machine is behind a firewall and `ravmd` can't get its host name. If you are using Postfix as MTA then you can set `ravmd` to use a specified port when sending warning mails. Setting `smtp_port` on 10026 (in our configuration example) will make Postfix to send those mails without being scanned.

`ravms_full_name` is a parameter included in `ravmd` starting with version 8.3.3. This parameter is used for compiling the sender's mail address according to RFC822, the standard for the format of ARPA Internet text messages.

Default values:

`ravms_name = ravms`

`on_host = official host name`

`smtp_server = official host IP address`

`smtp_port = 25`

`ravms_full_name = RAV Antivirus`

Example:

```
ravms_name = ravms
```

```
on host = ravantivirus.com
```

```
smtp_server = 127.0.0.1
```

```
smtp_port = 25
```

```
ravms_full_name = RAV AntiVirus Scanner
```

The mail address displayed in the warning mail will be: "RAV Antivirus Scanner" <ravms@ravantivirus.com>.

`no_subject = string`

Description: Parameter used for specifying the string replacing the `SUBJECT` macro in the warning mail if `ravmd` does not find a valid subject in the infected email.

Default value: `--no subject found--`.

Example:

```
no_subject = "original mail didn't contain any subject field"
```

mailer_daemon = string

Description: Parameter used for specifying the name that will replace the FROM_USER macro when the mail sender is <>.

Default value: --unknown--.

Example:

```
mailer_daemon= "MAILER-DAEMON"
```

Antispam parameters

The parameters pertaining to the antispam module of **ravmd** are described below.

The default configuration is included in the `antispam` file that you can find in the `${ETCDIR}` directory.

accuracy_low

accuracy_medium

accuracy_high

accuracy_very_high

Description: Keywords designating the accuracy level.

quarantine = string

Description: String specified in the `_define_strings` section defining the location where the messages meeting certain spam patterns will be saved for the groups where the **save** action is configured.

Default value: `${DATADIR}/bulk`.

extra_header = string

Description: String specified in the `_define_strings` section, defining the extra-header that will be added to the message tagged as spam.

The *default values* depend on the corresponding bulk detection accuracy level.

Example:

```
extra_header = bulk_header_high_english
```

extra_subject = string

Description: String specified in the `_define_strings` section, defining the extra-subject that will be added in the **Subject** field of the messages meeting certain spam patterns.

The *default values* depend on the corresponding bulk detection accuracy level.

Example:

```
extra_subject= bulk_subject_high_english
```

embedded_msg = string

Description: String specified in the `_define_strings` section, defining the message that will be used for the embedded mail body.

The *default values* depend on the corresponding bulk detection accuracy level.

Example:

```
embedded_msg= bulk_embedded_high_english
```

forward_to = enumeration

Description: List of mail addresses where the spam mail is to be forwarded. This parameter is available only from version 8.4.1 of **ravmd**.

Example:

```
forward_to = admin@domain.com
```

actions = variable

Description: String defined in **actions** file specifying the actions to be taken by **ravmd** for the corresponding bulk detection level.

The *default values* depend on the corresponding bulk detection accuracy level.

```
bulk_actions_low = add_subject, add_header, deliver
```

```
bulk_actions_medium = add_subject, add_header, deliver
```

```
bulk_actions_high = embed, add_subject, add_header, deliver
```

```
bulk_actions_very_high = save, discard
```

Example:

```
actions = bulk_actions_high
```

Note: When using this feature of **ravmd**, the messages tagged as spam are reinjected in the MTA queue using the following syntax:

```
cat <modified_mail> | sendmail -i -f<sender> <receivers>
```

(you must have 'sendmail' executable in your search path for commands - environment variable PATH).

If the sendmail binary is not corresponding to the binary of your MTA, then you should make the link manually in `/usr/sbin` (for example) by using (example for Qmail MTA):

```
ln -sf /var/qmail/bin/sendmail /usr/sbin/sendmail
```

This command assumes that you have the Qmail's binaries in `/var/qmail/bin`.

Real-time Blackhole List parameters

Real-time Blackhole List (RBL) is a feature available in **RAV AntiVirus for Mail Servers** starting with version 8.3.3. This functionality is implemented in `librbl.so` and it consists in defining a dynamic list (`rbl_site`) with sites containing listings of known spammers.

Note: The `rbl_site` list has to be configured and updated by your system administrators.

When this feature is enabled, **RAV AntiVirus for Mail Servers** checks if any of the IP addresses from the mail's header is listed on one of the websites defined in the `rbl_site`. If it does, the mail is automatically rejected.

No warning mail is sent; no file is saved to **RAV Quarantine** folder. The `librbl.so` library is loaded only if at least one site is included in `rbl_site`.

The `rbl` options are used for the `[global]` group. You cannot set different values for different groups. If you do not want the mails for one of your groups to be checked against the **Real-time Blackhole List**, you should use the following parameter in the configuration for that group:

```
use_rbl = no
```

Below you can find the parameters associated with this feature.

`use_rbl = boolean`

Description: Use this parameter to specify if the Real-time Blackhole List (RBL) feature is used or not for one specific group.

Accepted values: **Yes** or **No**.

Default value: **No**.

`rbl_site = enumeration`

Description: List containing the sites where **ravmd** will be looking for IP addresses for known spammers to be checked against the IP addresses from the mail's header.

`rbl_cache_file = string`

Description: Variable containing the path where **ravmd** cache is saved when the program is stopped. When **ravmd** is restarted, the cache is re-loaded from this path.

Default value: `${DATADIR}/rbl.cache`.

`rbl_cache_size= number`

Description: Library caching the latest IP addresses **rbl** has been looked for. The parameter following the '=' sign specifies how many IP addresses are included in this cache.

Default value: **10007**.

rbl_timeout = number

Description: Number specifying how many seconds **ravmd** will be waiting for one site to answer to its DNS request.

Accepted values: **minim=2, maxim=30.**

Default value: **5.**

rbl_retry = number

Description: Number specifying how many times the DNS request is sent to the same server in case of timeout.

Accepted values: **minim=1, maxim=5.**

Default value: **4.**

White/Black List parameters

`wbl_accept` IP, IP/MASK, IP/netmask, mail@address, mail.domain

`wbl_reject` IP, IP/MASK, IP/netmask, mail@address, mail.domain

`wbl_discard` IP, IP/MASK, IP/netmask, mail@address, mail.domain

Description: These parameters are used for specifying the parameters for the **Static White/Black List**, a feature included in **ravmd** starting with version 8.3.3. `wbl_discard` is available only from version 8.4.1. When using this parameter, the corresponding mails are discarded (rejected with no bounce back to the sender, saving therefore valuable bandwidth for users).

Using the **White/Black List** feature you can define IP addresses and mail addresses/domains to be included in the **Static White List** (mail messages to be received) or in the **Static Black List** (mail messages to be rejected). The keyword `wbl_accept` anticipates the IP addresses and mail addresses/domains added to the **Static White List**. The keyword `wbl_reject` anticipates the IP addresses and mail addresses/domains added to the **Static Black List**.

The rules are applied in the order you define them. The first rule to match will give the result of the `wbl` search: accept or reject. The `wbl` options are not inherited from the [global] group. The `wbl` search is done before the `rbl` search. If there is a **reject** rule match for the current mail, the message is automatically rejected. If there is a `wbl_accept` rule match for the current mail, no `rbl` search is executed. If no rule is found in the `wbl` for the current mail, the `rbl` search is executed (unless you specified `use_rbl=no` for the corresponding group).

No warning mail is sent; no file is saved to **RAV Quarantine** folder. The `libwbl.so` library is loaded only if at least one `wbl` rule is defined for one group.

Example:

1. If you want to accept mail messages only from the IP address 193.230.245.100 and to reject all mail messages from the entire class 193.230.245.0/24 (193.230.245.0/255.255.255.0), use:

```
wbl_accept 193.230.245.100
```

```
wbl_reject 193.230.245.100/24
```

2. If you want to accept mail messages only from the `user@domain.com` mail address and to reject all mail messages from the other mail addresses from the `domain.com`, use:

```
wbl_accept user@domain.com
```

```
wbl_reject domain
```

Miscellaneous parameters

These parameters are inherited by additional groups from the [global] group.

warn_header_msg = variable

Description: Parameter used for specifying the text used as a header in the warning mail.

Example:

```
warn_header_msg = notification_header
```

Where:

```
notification_header = "This message is automatically generated by RAV AntiVirus."
```

warn_footer_msg = variable

Description: Parameter used for specifying the text used as footer in the warning mail.

Example:

```
warn_footer_msg = notification_footer
```

Where:

```
notification_footer = "The mail scanned was received from: HEADER_RECEIVED."
```

warn_txt_msg = variable

Description: Parameter used for specifying the text appended after the default one in the warn.txt file.

Example:

```
warn_txt_msg = append_to_warn_txt
```

Where:

```
append_to_warn_txt = "Please contact your system administrator for more information."
```

For more information about the warn.txt, please refer to [FAQ 4](#).

charset = string

Description: Parameter (available beginning with version 8.3.3 of **ravmd**) used for specifying the value of the **charset** field used in the MIME header of the warning mails. If the warning mails contain strings with a character encoding system different from ASCII you should specify the respective encoding using the **charset** parameter.

Default value: **US-ASCII**.

Example:

```
charset = iso-2022-jp
```

custom_msg = number

The warning mails are created using the strings defined by the user in the `_define_strings` section and RAV-related information (always added during the evaluation period). A warning mail looks like this:

RAV AntiVirus for OSTYPE version: x.x.x (snapshot-yyyyymmdd)
 Copyright (c) 1996-2001 GeCAD The Software Company. All rights reserved.
 X more days to evaluate. (or: Registered version for N domain(s).)
 Running on host: HOSTNAME

The file ATTACHED_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S) is infected with virus: VIRUS_NAME. The file was successfully deleted by RAV AntiVirus.

Scan engine 8.7 () for i386.
 Last update: Thu, 27 Jun 2002 15:44:53 +0300
 Scanning for 68249 malwares (viruses, trojans and worms).

To get a free 30-days evaluation version of RAV AntiVirus v8 (fully functional) please visit:
<http://www.ravantivirus.com>

The macros are replaced with their corresponding values. In the registered version of **RAV AntiVirus for Mail Servers**, all RAV-related information can be omitted, except for the first header line.

In the registered version the warning mails can be customized.

Default value: **255** (use all RAV information).

Accepted values:

- = **0** – No information.
- + **1** – Add "Registered version ..."
- + **2** – Add "Running on host ..."
- + **4** – Add "Scan engine ..."
- + **8** – Add "Last update ..."
- + **16** – Add "Scanning for ..."
- + **32** – Add "To get a free 30-days ..."
- + **64** – Add "Copyright ..."
- + **128** – Add "RAV AntiVirus for ..."

Example:

custom_msg = 136

RAV AntiVirus for OSTYPE version: x.x.x (snapshot-yyyyymmdd)

The file ATTACHED_NAME attached to mail (with subject: SUBJECT) sent by FROM_USER to TO_USER(S) is infected with virus: VIRUS_NAME.

The file was successfully deleted by RAV AntiVirus.

Last update: Thu, 27 Jun 2002 15:44:53 +0300

max_processes = number

Description: Parameter used for specifying the maximum number of **ravmd** scanning processes running at the same time.

Accepted values: **1** to **128**.

Default value: **24**.

Example:

max_processes = 80

timeout_per_file = number

timeout_per_mega = number

Description: These parameters are used to specify the maximum time in seconds that a

scanning process can spend on a file. The total timeout is computed using the following formula:

$$\text{timeout_per_file} + \text{timeout_per_mega} * \text{filesize}/1\text{Mb}$$

Accepted values: 10-600 (for `timeout_per_file`) and 5-600 (for `timeout_per_mega`).

Default values: 300 for `timeout_per_file` 60 for `timeout_per_mega`.

Example:

```
timeout_per_file = 120
```

```
timeout_per_mega = 25
```

save_infected = boolean

save_suspicious = boolean

Description: Parameter used for specifying if infected/suspicious mail files are saved to the local disk before executing any action. You can set these options to **Yes** or **No**.

Default value: **Yes**.

Note: The infected/suspicious messages will be placed in the quarantine regardless of the defined `infected_actions` and `suspicious_actions`.

Example:

```
save_infected = no
```

```
save_suspicious = yes
```

quarantine = string

Description: String used to specify the directory where the infected/suspicious mails are saved.

Default value: `${DATADIR}/quarantine`.

Example:

```
quarantine = /tmp/rav/quarantine
```

RAV logging system

When **ravmd** is launched, it logs some information in the system mail info file (using **syslog**) then it switches to the internal log. By default the **log** files are created in: `${DATADIR}/log`. It is possible to use a different log file for every group, with different options for it. For this you have to declare one of the log options in that group. If a group doesn't contain any log options then the log data for the [global] group will be used. Here are the parameters used for **ravmd**'s logging system:

log_file_name = string

Description: Parameter used for specifying the path to and the name of the log file used by **ravmd**. If the path to the log file is not valid, **ravmd** will exit and inform the user that the log file could not be created.

Default value: `${DATADIR}/log/group_name`.

Example:

```
log_file_name = ${DATADIR}/log/global
```

```
log_file_name = ${DATADIR}/log/my_group
```

log_max_length = {number}(Kb|Mb)

Description: Parameter used for specifying the maximum log file size.

Default value: **500Kb**.

Accepted values: **10-1000Kb** and **1-10Mb**. Starting with version 8.4.0 beta 2 of **ravmd**, the **0** value is also accepted, assigning an unlimited length to the log file used by **ravmd**.

log_rotate_after = {number}(m|h|d)

Description: Parameter used for specifying the period of time elapsing before creating a new log file.

Default value: **6h** (hours). *Accepted values:* **10-60m** (minutes, with 23m rounded to 20m, 25m rounded to 30m), **1-24h** (hours) or and **1-30d** (days). Starting with version 8.4.0 beta 2 of **ravmd**, the **0** value is also accepted, determining **ravmd** not to rotate the log file anymore.

log_delete_after = {number}(h|d|m)

Description: Parameter used to specify the period elapsing before deleting log files older than the specified period.

Default value: **7d** (days).

Accepted values: **1-24h** (hours), **1-30d** (days) and **1-12m** (months). Starting with version 8.4.0 beta 2 of **ravmd**, the **0** value is also accepted, determining **ravmd** not to delete the log file anymore.

log_use_zip = boolean

Description: Parameter used for specifying if the log files should be archived (using the **zlib** library) or not.

Default value: **Yes**.

log_level = number

Description: Number controlling the logging level used by **ravmd**.

Default value: **2047** (use all **ravmd** information).

Accepted values:

0 – No log information.

+ **1** – Add error messages (i.e. “can't fork”, “error reading from socket”, etc.).

+ **2** – Add the name of the mail file.

+ **4** – Add mime part scanned.

+ **8** – Add final scan result.

+ **16** – Add actions taken during scanning.

+ **32** – Add the mail addresses of the sender and the first receiver.

+ **64** – Add the group name matched.

+ **128** – Add information generated by the external triggered update.

+ **256** – Add LICENSE LIMIT warnings.

+ **512** – Add **WBL** logs.

+ **1024** – Add **RBL** logs.

+ **2048** - Add actions performed for bulk mails (this log level is available only from version 8.4.1).

+ **4096** - Add all IPs from the **Received:** mail header field (this log level is available only from version 8.4.1).

+ **8192** - Add the rule matched by the content filtering (this log level is available only from version 8.4.1).

Example:

To set **ravmd** to display only **RBL** logs, set the value for **log_level** to 1024. To set **ravmd** to display error messages *and* **RBL** logs, set the value for **log_level** to 1025 (1 + 1024).

Group-specific parameters

The following parameters must have **different** values for every defined group. If these parameters are not defined for each group, their values are **NOT** copied from the [global] group. The *default values* are specified for each of these parameters in the following section.

filter_subject variable_1 variable_2

Description: This parameter is used for *subject filtering*. **variable_1** is a regular expression defined in the `regexp` file from your `#{ETCDIR}` directory. **variable_2** is a variable defined in the `actions` file from your `#{ETCDIR}` directory. If this parameter is not defined then the **subject filtering** is disabled.

Example:

```
filter_subject subj_regexp subj_actions
```

Where:

```
subj_regexp = I love you
```

```
subj_actions = reject
```

Using this rule, mails having the "I love you" string in the **Subject** field will be rejected. Please note that you can use here a regular expression, not only a simple string.

filter_attachment variable_1 variable_2

Description: This parameter is used to filter the mail's attachments names. **variable_1** is a regular expression defined in the `regexp` file from your `#{ETCDIR}` directory. **variable_2** is a variable defined in the `actions` file from your `#{ETCDIR}` directory. If this parameter is not defined then the **attachment names filtering** is disabled.

Example:

```
filter_attachment file_regexp file_actions
```

Where:

```
file_regexp = .*((vbs)|(exe)|(com))
```

```
file_actions = delete, reject
```

This filtering rule deletes all the attached files with extension ".vbs", ".exe" or ".com" from all mail messages. If a file cannot be deleted then the entire mail message is rejected.

filter_content variable_1 variable_2

Description: This parameter is used to filter the mail's *body* and the *attachment's contents*. **variable_1** is a string defined in the `regexp` file from your `#{ETCDIR}` directory. **variable_2** is a variable defined in the `actions` file from your `#{ETCDIR}` directory. If this parameter is not defined, then the **body filtering** (inside the *mail body* and the *attachment's contents*) is disabled.

Example:

```
filter_content body_string_1 body_actions_1
filter_content body_string_2 body_actions_2
filter_content body_string_3 body_actions_3
filter_content body_string_4 body_actions_4
filter_content body_string_5 body_actions_5
filter_content body_string_6 body_actions_6
```

Where:

```
body_string_1 = confidential
body_string_2 = salaries
body_string_3 = balance sheet
body_string_4 = tax
body_string_5 = income
body_string_6 = revenue
```

and:

```
body_actions_1 = delete, reject
body_actions_2 = delete, reject
body_actions_3 = delete, reject
body_actions_4 = copy, ignore
body_actions_5 = copy, ignore
body_actions_6 = copy, ignore
```

In this case the content filtering module of **ravmd** is looking for user-specified strings. A priority is assigned to each rule. The rule with the highest priority is the first one you specify (**body_string_1**). This rule has a priority of 1. The next rules receive lower priority levels (2...n), depending on the order you are specifying them (**body_string_2** has a priority of 2, **body_string_3** has a priority of 3 and so on). In our example, **ravmd** will start searching for all the strings defined by the user (**confidential**, **salaries**, **balance sheet**, **tax**, **income** and **revenue**) in the same time. If the first match is found for one string having the highest priority level (**confidential**, in this example), the search stops and the actions from **body_actions_1** are executed. If the first match is found for one string having a lower priority level (**tax** for instance), the search will continue for the rest of the mail, looking for eventual matches for **confidential**, **salaries** and **balance sheet**, strings with higher priority. If a match is found, **ravmd** will execute **body_actions_1** if the match is for **confidential**. If the match is not for **confidential** but for **salaries** for instance, **ravmd** will keep looking for **confidential**. If a match with **confidential** is found, **ravmd** will execute **body_actions_1**. If no match with **confidential** is found, **ravmd** will execute **body_actions_2** (actions corresponding to **salaries**). **ravmd** will ignore the first match (for **tax**) in all these cases.

If after finding a match for **tax** no match with higher priority strings (**confidential**, **salaries** or **balance sheet**) is found, **ravmd** will execute **body_actions_4** (the actions corresponding to **tax**).

Because **body_actions_1**, **body_actions_2** and **body_actions_3** are identical (**delete, reject**) and **body_actions_4**, **body_actions_5** and **body_actions_6** are also identical (**copy, ignore**), you can significantly simplify your work using:

```
body_string_1 = confidential|salaries|balance sheet
body_string_2 = tax|income|revenue
```

and:

```
body_actions_1 = delete, reject
body_actions_2 = delete, reject
```

The behaviour of **ravmd** is in this case the same as explained above.

Note: If you want to define content filtering rules for mail bodies containing strings that include the „|“ character, use „|“ (i.e. `body_string = confidential||salaries` will return matches for mail bodies containing the „confidential|salaries“ expression).

warn_sender = enumeration

Description: Use this parameter to specify when to send warnings to the mail sender. The valid keywords are explained in the table below.

warn_receiver = enumeration

Description: Use this parameter to specify when to send warnings to the mail receivers. The valid keywords are explained in the table **below**.

warn_admin = enumeration

Description: Use this parameter to specify when to send warnings to the mail receivers. The valid keywords are explained in the table **below**.

You have to specify *who* is warned by **ravmd** and *when*. If one of these parameters is not defined then the respective user category will **not** receive warnings. The valid keywords are specified in the table **below**.

Keyword	Meaning
found_virus	Send alert to the defined recipients when a virus is found.
found_subject	Send alert to the defined recipients when the subject matches a content filtering rule.
found_attach	Send alert to the defined recipients when an attached file name matches a content filtering rule.
found_content	Send alert to the defined recipients when the mail body contains a string matched by a content filtering rule.
always	Send alert to the defined recipients in all the above-mentioned situations.
never	Never send alert.
match_all_flags	Send alert to the defined recipients only when <i>ALL</i> the previously defined rules (<code>found_virus</code> , <code>found_subject</code> , <code>found_attach</code> or <code>found_content</code>) are matched.
warn_domains	New flag available from version 8.4.1. When using <code>warn_domains</code> , only the users from RAV-protected domains (specified in the <code>domain</code> parameter) are notified. You should use this flag for the <code>warn_sender</code> parameter, in order to avoid warning mails from being sent to fictive addresses used by some viruses.

Table 3: Keywords for warning messages.

Please note that these values are not inherited from the [global] group. This way you can specify different notification policies for different groups. For more info, please read the examples below.

Example 1:

`warn_sender = found_virus, found_subject, found_attach, found_content`

```
warn_receivers = found_virus  
warn_admin = always
```

In this example, the sender is warned whenever a virus is found *or* the subject matches a content filtering rule *or* an attached file name matches a content filtering rule *or* the mail body contains a string matched by a content filtering rule. The receivers are warned *only* when a virus is found. The administrator is *always* warned.

Example 2:

```
warn_sender = found_virus, found_subject, match_all_flags
```

In this example, the sender is warned whenever a virus is found *AND* the subject matches a content filtering rule.

Example 3:

```
warn_receiver = found_virus, found_subject, found_content, match_all_flags
```

In this example, the receivers are warned whenever a virus is found *AND* the subject matches a content filtering rule *AND* the mail body contains a content filtering rule match.

Example 4:

```
warn_sender = found_virus, found_subject, found_content, match_all_flags, warn_domains
```

In this example, only the senders from RAV-protected domains are warned whenever a virus is found *AND* the subject matches a content filtering rule *AND* the mail body contains a content filtering rule match.

do_not_scan = boolean

Description: Parameter used for specifying if the mail files for the current group are scanned or not. This way it is possible to exclude some mail addresses and/or domains from the scanning process.

Default value: **No**.

Example:

```
do_not_scan = yes
```

do_not_warn = enumeration

Description: Parameter used for specifying the mail addresses that will not be notified.

do_not_show = enumeration

Description: Parameter used for specifying the mail addresses that will be hidden in all warning mails.

Note: Why is this parameter required? There may be cases when one user should not be notified or his mail address should not be displayed in the warning mails. This parameter will help you solve the problem. Note that only the receiver's mail address is compared against the specified address.

`do_not_warn` and `do_not_show` parameters have no *default values* (no comparison will be made).

Example:

```
do_not_warn = user3@domain2.org
```

do_not_show = user1@domain1.com, user2@domain1.com

admin_addr = enumeration

Description: Parameter used for specifying the mail addresses of the administrators to be notified when infected or suspicious files are detected. The warning mail contains messages created using the strings specified for each situation. This parameter has no *default value*.

Example:

admin_addr = postmaster@domain1.com, postmaster@domain2.net, user1@domain1.com

admin_addr = ravmails@stats.ravantivirus.com

Note: Forwarding warning mails to ravmails@stats.ravantivirus.com in case of virus infections is highly recommended. This will help RAV Research Team to determine the level of spreading for new viruses or pinpoint potential detection problems. The Technical Support team at GeCAD Software may also diagnose potential problems for the user, such as old updates of the virus signatures database. In any case, the user will be informed about the best solution for solving his problem. GeCAD Software treats each mail in strict confidence.

disclose_sender = boolean

disclose_receivers = boolean

disclose_admin = boolean

Description: Parameters used for specifying if the addresses of the corresponding sender/receivers/administrators will or will not be disclosed in the warning mail's **To:** header and body.

Default: **No** (for all three parameters). If no value is specified for any of these parameters, they are inherited from the [global] group.

antispam_configuration = enumeration

Description: Using the `antispam_configuration` parameter you can set the desired actions for the four different accuracy levels available (**low**, **medium**, **high** and **very high**). In the **enumeration** part you should specify `name_conf_antispam1`, `name_conf_antispam2`, `name_conf_antispam3` and `name_conf_antispam4`, corresponding to the four different accuracy levels.

antispam_configuration = name_conf_antispam1, name_conf_antispam2, name_conf_antispam3, name_conf_antispam4

advertising_msg = variable

Description: Using the `advertising_msg` parameter you can append a personal message to each mail message scanned by **ravmd**. The length of the mail message will grow accordingly when using this feature.

The `advertising_msg` parameter must be declared in the group file or even in the [global] group. The value for **variable** must be declared in the language file and the `language.equiv` file must be included in the group file.

Note 1: RAV AntiVirus for Mail Servers cannot append the text to any mail due to the MIME format of the mail and the `advertising_msg` feature is not supported by **ravcgate**.

Note 2: The `advertising_msg` parameter is not supported by **ravcgate**.

Example:

- In the group file (that might be even the [global] one) define:
_include \${ETCDIR}/languages/english.equiv
advertising_msg = my_advertising
- In \${ETCDIR}/languages/english define:
my_advertising = "COMPANY_NAME maintains mail messages virus free"

update_executable= string

Description: Parameter used for specifying the name of the executable file used by **ravmd** to start the update process. You must specify the full path to the executable file. The update process is started only if **ravmd** is receiving an update mail sent by RAV Team (triggered update - feature available from **ravmd** version 8.4.0). If you want to receive update mails, please send a request to: updates-l-subscribe@lists.ravantivirus.com

Default value: \${BINDIR}/ravmdupdate.sh.

Example:

The following line disables the **update executable** feature:

```
update_executable = null
```

The following line executes the specified script file every time an update mail is processed by **ravmd**:

```
update_executable = /home/ravms/ravmdupdate.sh
```

Embedded messages

`embed_clean_mail = boolean`

Default value: No.

`embed_cleaned_mail = boolean`

Default value: No.

`embed_unclean_mail = boolean`

Default value: No.

`use_embedded_msg = boolean`

Default value: No.

`use_embedded_warning = boolean`

Default value: No.

Description: These parameters are used for specifying what messages you want to be send as embedded mails. Embedded mails are a new feature available in **ravmd** version 8.3.3. After being scanned, the original message can be attached to a new mail message, created by **ravmd**, and sent to its recipients. When the new mail is accepted by the MTA, the original mail is discarded (if the MTA is supporting the **discard** feature) or rejected (if the MTA does not support the **discard** feature, i.e. Courier or Dmail). When the new mail is not accepted by the MTA, the original message is sent instead.

You can specify the mail messages to be encapsulated:

- **embed_clean_mail**: encapsulate all clean mails (mail messages that were not infected/did not contain suspicious code),
- **embed_cleaned_mail**: encapsulate all cleaned mails (mail messages that were infected/did contain suspicious code, but they were cleaned by **RAV AntiVirus for Mail Servers**), or
- **embed_unclean_mail**: encapsulate all uncleaned mails (mail messages that were infected/did contain suspicious code and RAV AntiVirus was not able to disinfect them).

For each status described above the administrator can send a customized message using the `embedded_clean_msg`, `embedded_cleaned_msg` and `embedded_unclean_msg` parameters (described below).

- **use_embedded_msg**: when using this parameter, `embedded_clean_msg`, `embedded_cleaned_msg` and/or `embedded_unclean_msg` are added to the encapsulated mail.
- **use_embedded_warning**: when using this parameter, the warning mail is added to the encapsulated mail only if the original mail is infected.

The rules for embedded mails are not inherited from the `[global]` group – you have to define the parameters for the suitable groups.

`embedded_clean_msg=variable`

`embedded_cleaned_msg=variable`

`embedded_unclean_msg=variable`

Definition: These three parameters are used for specifying the customized message send when embedding clean/cleaned/uncleaned messages.

The newly created message will therefore include, besides the original message:

- the customized message (when defined by the administrator); and
- the customized warning messages (when defined by the administrator).

embed_queue = string

Definition: The path on the local disk where **ravmd** is temporarily saving the embedded mail messages, before sending them.

Default value: `${DATADIR}/tmp`.

Note: When using this feature of **ravmd**, the messages tagged as spam are reinjected in the MTA queue using the following syntax:

```
cat <modified_mail> | sendmail -i -f<sender> <receivers>
```

(you must have 'sendmail' executable in your search path for commands - environment variable PATH).

If the sendmail binary is not corresponding to the binary of your MTA, then you should make the link manually in `/usr/sbin` (for example) by using (example for Qmail MTA):

```
ln -sf /var/qmail/bin/sendmail /usr/sbin/sendmail
```

This command assumes that you have the Qmail's binaries in `/var/qmail/bin`.

BUGS

Please mail bug reports and suggestions to: ravteam@ravantivirus.com

SEE ALSO

ravmd(8), **ravav(8)**