USER GUIDE

RAV AntiVirus for Mail Servers (Unices and MacOS X)

Release Date: January 27, 2003

Product version: 8.4.0

User Guide revision: 2.0

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

USER GUIDE

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.

USER GUIDE

CONTENTS

Part I: Introduction

Congratulations! You have just acquired a **RAV AntiVirus** product. **RAV AntiVirus** is one of the best antivirus programs, ranked among the first ten in the world.

RAV AntiVirus products and the documentation associated to these products are the exclusive property of GeCAD Software. The products are licensed to the users under the terms of the License Agreement accompanying each product, so please carefully this License Agreement.

We suggest to take a moment to fill the *Registration Form* attached to the *License Certificate* or register to the manufacturer's site. The personal data you fill in this form is strictly confidential and will only be recorded in our database of registered customers to keep you informed on new developments and updates and to activate your technical support account. Any suggestion you make will be taken into consideration for future versions.

Note: Because of the attention we pay to delivering highly efficient antivirus software, the rapid evolution of viruses and implementation of features requested by our customers, the present documentation may not be up to date. The best source of information on **RAV AntiVirus** products is our website, which you can find at http://www.ravantivirus.com/.

Structure of this document

This document is structured in four major components and two appendixes.

The first part, *Introduction*, includes information on the *Scope* of this document, its *Intended audience* and the *Related documentation*. You are also briefed on the manufacturer of **RAV AntiVirus**, GeCAD Software, and informed about modalities you can use to obtain the *Technical support* you might need for the product you have acquired.

The second part, *RAV AntiVirus Product Family*, is an overview of the products included in **RAV AntiVirus** product family (**RAV AntiVirus Desktop**, **RAV AntiVirus for Mail Servers**, **RAV AntiVirus MailFilter**, **RAV AntiVirus for File Servers** and **RAV AntiVirus for Instant Messengers**). All these products are based on *RAV Engine*, a revolutionary antivirus engine. The cutting-edge technologies included in RAV Engine and the most important updates included in RAV Engine version 8.9 are also included in this *RAV AntiVirus Product Family* part.

The third part of the document, *RAV AntiVirus for Mail Servers*, includes some sections (*Short description, What can it do, How does it work, Who should use it*) briefly presenting **RAV AntiVirus for Mail Servers**, its functionality and the potential users. Sections describing the *Currently supported operating systems and MTAs* and the *Hardware and software requirements* are also included, as well as a section presenting in more detail the *features* of **RAV AntiVirus for Mail Servers**. An *Awards* section containing international acknowledgements gained by this product and a *Registration procedure* section, explaining the main principles fundamental for the license policy of our company, including a detailed description of the main differences between the three different phases in which you can use a **RAV AntiVirus** product (evaluation, registration and activation), are also included in this part of the document. The *Updates* section lets you know the available update servers for **RAV AntiVirus** products.

The fourth part of the document, *Configuration files and man pages*, includes the configuration file for **ravmd**, as well as the **man** pages for **ravmd**, **ravav**, **ravupdate**, and the external filters for the MTAs currently supported by **RAV AntiVirus for Mail Servers.**

Two appendixes (*Bug Report Form* and *Index*) are included at the end of the document.

Scope

This document describes the features and functionality of **RAV AntiVirus for Unices and MacOS X Mail Servers.** Additional valuable documentation is also separately available - see the Related documentation section below.

To make sure you will be efficiently using **RAV AntiVirus for Mail Servers** from the very beginning, we strongly recommend you to read carefully this *User Guide*, even if you have been using a previous version of this product.

Intended audience

This *User Guide* is intended for administrators responsible with the installation of **RAV AntiVirus for Unices and MacOS X Mail Servers.** These persons should have a strong and comprehensive knowledge and an extensive working experience in the operating systems the product is designed for.

Related documentation

Here is a list of documents that should be used in connection with this *User Guide for RAV AntiVirus for Unices and MacOS X Mail Servers*:

- RAV AntiVirus for Mail Servers – *Release Notes* (available at http://www.ravantivirus.com/rav/mailservers/documentation/notes/releasenotes840.txt);
- **RAV AntiVirus for Mail Servers** – *Product Sheet* (available at http://www.ravantivirus.com/rav/mailservers/documentation/pdf/ravmailservers.pdf);
- **RAV AntiVirus for DMail on Windows** - *User Guide* (available at http://www.ravantivirus.com/rav/mailservers/documentation/pdf/ravdmailwin32-usersguide.pdf);
- **RAV AntiVirus for CommuniGate Pro on Windows** - *User Guide* (http://www.ravantivirus.com/rav/mailservers/documentation/pdf/ravcgatewin32-usersguide.pdf).

These documents and other up-to-date documentation concerning **RAV AntiVirus** products, as well as white papers on security policies and the latest information about viruses are available on our web site (http://www.ravantivirus.com/support/documentation.php).

USER GUIDE

About GeCAD Software

GeCAD Software is a leading technology company specialized in providing top anti-virus solutions for all categories of users. After releasing its first antivirus program, back in 1994, GeCAD Software has grown to be represented, by Distributors, Resellers and OEM Partners, on all the continents around the world. Our strong commitment towards quality has secured us a privileged position in a fast-evolving market, the key advantage being a state of art product based on cutting-edge technologies.

Founded in 1992, GeCAD Software is headquartered in Bucharest, Romania, and its activity is focused on producing, developing and internationally distributing high-quality antivirus products.

Technical support

We value very much your opinion and try to fulfil our customers' requests as soon as possible. Only you, the user of RAV AntiVirus, can help us make the product better and more suitable to your needs. Therefore, your suggestions are more than welcomed.

For any details regarding the installation and the functionality of this product, please contact the local dealer you have bought the product from. If you consider you do not receive an adequate technical support, please contact us.

For any suggestions or problems regarding the copyright, the guarantee and other aspects related to RAV AntiVirus products or data recovery from a destructive viral attack, please contact us at the following address:

GeCAD Software S.R.L.

Address:	223, Mihai Bravu Blvd, 3rd district, Bucharest, ROMANIA
Phone/Fax:	+40-21-321 78 03
Hotline:	+40-21-321 78 59
E-mail:	
Sales:	international.sales@ravantivirus.com
Technical support:	support@ravantivirus.com
Website:	http://www.ravantivirus.com/support

To keep in contact with other users of RAV AntiVirus products, join the *discussion lists* available for our products or subscribe to the free weekly *newsletter* edited by GeCAD Software. You should also consider using the *Virus Encyclopedia* and our *Knowledge Base* and subscribing to *RAV Outbreak Security Service* and *RAV Newsletter*, services offered to you free of charge and described below.

RAV Discussion Lists

Interesting ideas and insights, installation and configuration scenarios, troubleshooting solutions and other information are also available *via* specialized *discussion lists* for RAV Antivirus products:

- *rav-cgate* – RAV AntiVirus for CommuniGate Pro;

- *rav-courier* – RAV AntiVirus for Courier;
- *rav-desktop-unices* – RAV AntiVirus Desktop for Unices;
- *rav-desktop-windows* – RAV AntiVirus Desktop for Windows;
- *rav-dmail* – RAV AntiVirus for DMail;
- *rav-enterprise* – RAV AntiVirus Enterprise;
- *rav-exchange* – RAV AntiVirus for MS Exchange Server;
- *rav-exim* – RAV AntiVirus for Exim;
- *rav-mailfilter* – RAV AntiVirus MailFilter for POP3, IMAP, SMTP;
- *rav-novell* – RAV AntiVirus for Novell Networks;
- *rav-postfix* – RAV AntiVirus for Postfix;
- *rav-qmail* – RAV AntiVirus for Qmail;
- *rav-sendmail* – RAV AntiVirus for Sendmail;
- *rav-fileservers* – RAV AntiVirus for File Servers (Win32);
- *rav-samba* – RAV AntiVirus for File Server (Samba).

You can subscribe to these discussion lists by visiting our website http://www.ravantivirus.com/ (*RAV Discussion Lists* section) or by sending an empty e-mail message to: listname-subscribe@lists.ravantivirus.com (replace "`listname`" with the list you want to subscribe to).

RAV Newsletter

A free *newsletter*, containing virus alerts, advisories and other useful advices for avoiding virus disasters, as well as information regarding updates, tips and tricks and insights on RAV AntiVirus products, is also available from GeCAD Software. You can subscribe to this newsletter using this link: http://www.ravantivirus.com/pages/newsletter.php.

Knowledge Base

The Knowledge Base is a new service offered to you by the producer of RAV AntiVirus. You can access it at the following address: http://www.ravantivirus.com/kb. Here you can find technical information regarding the configuration and usage of all the products included in RAV AntiVirus family (RAV AntiVirus Desktop, RAV AntiVirus for Mail Servers, RAV AntiVirus for Windows File Servers, RAV AntiVirus MailFilter and RAV AntiVirus for Instant Messengers).

Virus Encyclopedia

RAV Virus Encyclopedia is a professional knowledge resource, specially designed to allow you being always up-to-date with the latest virus information and threats. RAV Virus Encyclopedia includes information on the most important and interesting viruses, including details on the *Description*, *Payload*, *Likelihood*, *Technical Description*, *Removal Instructions* and other important info. RAV Virus Encyclopedia is available at the following address: http://www.ravantivirus.com/pages/virus.php

USER GUIDE

RAV Outbreak Security Service

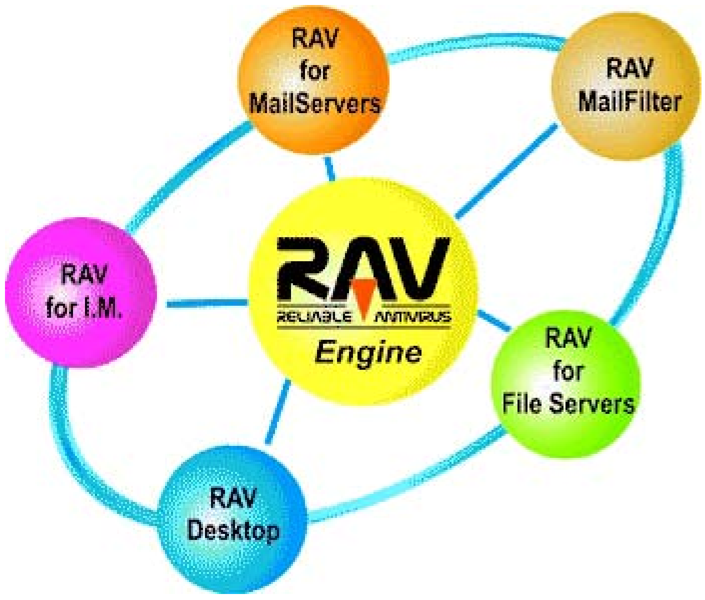
RAV Outbreak Security Service is a free subscription-based service offered by GeCAD Software as first-hand information alerting users in case of **virus outbreaks.** To subscribe to **RAV Outbreak Security Service**, visit http://www.ravantivirus.com/pages/outbreak.php, just enter your e-mail address in the corresponding text field and press on the **Subscribe** button. You will receive real-time customized notifications on the most recent and dangerous threats to your system's security.

Part II: RAV AntiVirus Product Family

The products

RAV AntiVirus product family is currently having the following members:

- **RAV AntiVirus Desktop** (for Windows and Unices);
- **RAV AntiVirus for Mail Servers**;
- **RAV AntiVirus MailFilter**;
- **RAV AntiVirus for File Servers** (Windows and Samba);
- **RAV AntiVirus for Instant Messengers**.



RAV AntiVirus for Mail Servers and all the other products included in **RAV AntiVirus** family are based on *RAV Engine*, now at version 8.9.

The Engine

RAV Engine combines the operational strength, the extensibility, the scalability, the scanning speed and the robustness needed in the fight against viruses and other malicious software (Trojans, worms, hoaxes, etc.). At this writing, RAV Engine includes in its database 76,999 distinct malware signatures and the RAV Antivirus Research Team daily adds new signatures to the RAV Engine's database. RAV Engine includes modules for scanning **inside archives** that detect infected files in most common types of archives and can scan archives inside archives no matter how deep they go. RAV Engine also scans inside **packed executables** (`lzexe`, `pklite`, `cryptcom`, `wwpack`, `aspack`, `pepack`, `vgcrypt`, `upx`). Working with Virtual File Systems, RAV Engine can scan the processes in memory and IFS chains, thus detecting and cleaning resident viruses like CIH.

RAV Engine also implements two extra modules (**bulk mail** and **content filtering**) that should help the user customize the product to work the way he/she wants.

Cutting-edge technologies included in RAV Engine

RAV Engine, now at version 8.9, has some unique features, ranking it among the best antivirus engines in the world. Here is a short description of the cutting-edge technologies included in the latest version of RAV Engine.

The TPI (Total Platform Independent) technology

The same engine is used for detecting and cleaning malwares for each operating system and platform **RAV AntiVirus** is installed on. Thus, the intelligibility, the unity and the easy update of our programs are logically ensured for all the products from RAV **AntiVirus** family (**RAV AntiVirus Desktop**, **RAV AntiVirus for Mail Servers**, **RAV AntiVirus for File Servers**, **RAV AntiVirus MailFilter** and **RAV AntiVirus for Instant Messengers**).

The IC (Integrity Checker) technology

When the files are scanned for the first time, the detection engine creates a database with all the information it has gathered during the scanning process. When doing a second scan, only the new or changed files are scanned, therefore increasing the detection speed with over 50%.

The MLES (Multi Layer Embedded Scanning) technology

RAV engine is promptly responding to any threat, scanning embedded objects on multiple layers, without affecting the detection speed or slowing down the machine it is installed on.

The HMETH (heuristic method) technology

Using this technology for all electronic threats, RAV Engine can study the behaviour of eventual malwares (malicious software, like worms, Trojans and hoaxes) and propose different methods for handling it. Therefore, a virus can be detected and cleaned even if its signature does not exist in the database.

The BMS (Bulk Mail Stop) technology

Bulk mails are causing daily headaches to system administrators and other people. RAV Engine can help you get rid of these unsolicited messages, using a heuristic detection of bulk messages based on mail headers and bodies.

The user can choose between four different levels of accuracy (Low, Medium, High and Very High) and can customize different actions (delete, block, reject) for each level.

The CF (Content filtering) technology

RAV Engine scans mail messages on three different levels: **Subject**, **Attachments names** and **Body**, looking for regular expressions or user-defined strings. RAV Engine can execute the actions you specify for mail messages matching these criteria.

The CUP (Cumulative Update Plug-ins) technology

The cumulative update is another advantage of RAV Engine, being used to add to the main signatures database only the latest available signatures. This procedure results in extremely small files used for the update (10-15Kb), very little download time (5-10 seconds for a 28,8Kbs connection speed) and better management of the virus signatures.

RAV Engine 8.9 vs. 8.7

RAV Engine version 8.9 was released on August 28th, 2002. In comparison with the precedent version of RAV Engine (8.7), the current version is enjoying the following improvements:

- A heuristic SPAM (Bulk mail) detection module was included. To make updating for this module more efficient, a new VDM module was included in RAVE's distribution (`filters.vdm`).
- RAR3 archives are now supported.
- Improved scanning inside CHM/HXS embedded files.
- Improved scanning inside SFX archives. Support for these files was significantly improved, and the number of supported types has increased.
- Improved heuristics/generic detection for Visual Basic-compiled programs.
- Improved heuristics/generic detection for .NET compiled programs.
- Added heuristics/generic detection for Visual C compiled programs.
- Added detection for several heavy-polymorphic Win32 viruses.
- Added support for `a.out` executable files, and improved the support for damaged ELF files.
- Improved the LE/LX executable parser to handle damaged/handcrafted files.
- Scanning inside files packed by several installers was included.
- Scanning of OMF object files is now supported.
- Damaged (without headers) MIME files are now supported.
- PEPatch, PaquetBuilder, PrivateEXE v2.2, EPPE, PeBundle, Shrinker, SPEC AcidCrypt support was included/improved.
- Improved the loading time and the scanning speed under Win95/Win98 VxD platform.
- Improved the **Scanner** process under Windows 9x and Windows NT.

Part III: RAV AntiVirus for Mail Servers

Short description

RAV AntiVirus for Mail Servers is a highly customisable award-winning antivirus program for mail servers using different operating systems on different platforms. Besides its main antivirus functionality, starting with its 8.4.0 version, **RAV AntiVirus for Mail Servers** also acts as **antispam** program, due to a combination between the new bulk mail module (available in RAV Engine starting with version 8.9) and features already implemented in older versions of **ravmd**, i.e. **Real-time Blackhole List** (RBL) and the **White/Black List** (WBL).

What can the product do

RAV AntiVirus for Mail Servers scans and cleans mail messages and all types of attachments including archives, exe files, embedded files, etc. It helps you avoid Internet malwares (viruses, worms, Trojans, hoaxes, etc.), bulk mail and information leaks. **RAV AntiVirus for Mail Servers** is scanning, detecting and removing any electronic threats from the messages flowing to/from your mail server, therefore protecting important data for your company and preventing your computers from being infected by viruses, worms, Trojans and other malwares.

When finding infected/suspicious objects, customisable warning mails can be sent to the sender, receiver(s) and system administrator(s), according to the specified settings.

The content filtering module of **RAV AntiVirus for Mail Servers** allows you to define different rules for filtering incoming/outgoing messages at different levels (subject, body, attachment file name and attachment contents) and specify actions to be taken for messages matching the content filtering rules.

How does it work

The product contains a daemon based on *RAV Engine* (**ravmd**, RAV **mail daemon**) and a filter module interfacing the daemon with your mail server (**ravcgate**, **ravcourier**, **ravdmail**, **ravexim**, **RAVMilter**, **ravpostfix**, **ravqmail** or **ravsendmail**).

When a mail message is reaching a mail server protected by **RAV AntiVirus for Mail Servers**, it is intercepted by the filter and sent to the daemon for scanning (the scan process is executed using RAV Engine).

Assuming the message is clean, it is sent back to the filter, which will decode and redirect it to the mail server. Then the message can be sent - scanned and cleaned – to the initial destination.

If the message is not clean, customisable actions are taken, according to the options set by the system administrator.

Who should use it

RAV AntiVirus for Mail Servers is targeting three categories of users:

- **large companies**, with heavy traffic on their mail servers and demanding security policies;
- **Internet Solution Providers** (ISPs – over 60% of them are using Linux-based mail servers). **RAV AntiVirus for Mail Servers** is the perfect solution for ISPs are dealing with heavy traffic and large amount of clients. **RAV AntiVirus for Mail Servers** can improve the services ISPs are providing to their customers by scanning the mail flow and adding protection against viruses for the hosted domains; and
- **small companies** willing to protect their internet traffic at low cost.

No matter if you're a big or small company or an ISP, the protection offered by **RAV AntiVirus for Mail Servers** is always working on multiple levels:

- You are protected of viruses and other malwares that might try to infect your machines coming from the Internet, as mail attachments or mail messages.
- The outgoing flow is scanned for viruses and other malwares.

You can also customize the outgoing mail flow for not allowing sensible information to leave your company.

Awards

Starting with March 1st, 2002, **RAV AntiVirus for Mail Servers** (Linux) is awarded the West Coast Labs'[1] Checkmark Certificate *Level 1* and *Level 2*, the two standards to be achieved by antivirus products. **RAV AntiVirus for Mail Servers** is the first (and so far the only) anti-virus software winning both certificates for Linux-based mail servers. The Checkmark Certificate system establishes standards for computer security products, giving end users a clear idea on reliable anti-virus products.

For a product to be certified to Anti-Virus Checkmark, *Level One*, the product must be able to detect all "in the wild" viruses. According to Checkmark, "In the Wild" viruses are currently defined as those appearing on The Wildlist Organization's "In the Wild" list and reported as such by more than one person.

For a product to be certified to Anti-Virus Checkmark, *Level Two*, the product must be able to comply with Checkmark Level One and, in addition, disinfect all "in the wild" viruses capable of being disinfected. The product must also detect all viruses on the wild list more than one month old.

After rigorous tests, conducted in the West Coast Labs, **RAV AntiVirus for Mail Servers (Linux)** proved to successfully detect and disinfect 100% of all in-the-wild types of viruses[2]. Anti-Virus Checkmark Level 2 being granted to **RAV AntiVirus for Mail Servers (Linux)** indemnify

[1] West Coast Labs is an independent organization testing information security products. It is owned by West Coast Publishing Limited that also owns *SC Magazine*, the largest circulation information security magazine in the world.

[2] For an overview of different types of viruses and other malwares and methods for protecting against them please consult the white papers available on our website, http://www.ravantivirus.com/.

the users to have complete confidence in our product's abilities to prevent infection and disinfect all the viruses from the "In the Wild" list.

Currently supported operating systems, platforms and MTAs

RAV AntiVirus for Mail Servers is currently available for the following platforms/operating systems:

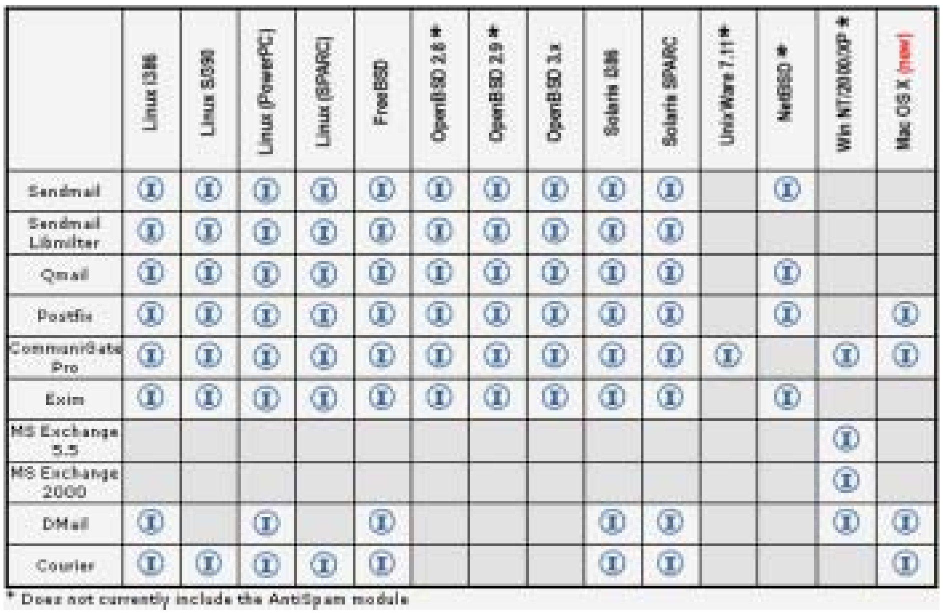
- Linux (Slackware, Mandrake, SuSe, RedHat, e-Smith, Debian, etc.) on i386 platforms;
- Linux (SuSe, RedHat and Mandrake) on s390 platforms;
- Linux (Yellow Dog) on ppc platforms;
- Linux (Suse, Mandrake, Red Hat) on Sparc platforms;
- FreeBSD on i386 platforms;
- Open BSD (2.8, 2.9 and 3.x) on i386 platforms;
- Solaris on i386 platforms;
- Solaris on Sparc platforms;
- Unixware on i386 platforms;
- Mac OS X on ppc platforms;
- Windows NT and Windows 2000 on i386 platforms.

RAV AntiVirus for Mail Servers is currently supporting the following MTAs:

- CommuniGate Pro
- Courrier
- DMail
- Exim
- MS Exchange (5.5 and 2000)
- Postfix
- Qmail
- Sendmail
- Sendmail-Milter.

Here a cross-table presenting the supported operating systems, platforms and MTAs supported by **RAV AntiVirus for Mail Servers** version 8.4.0 at this writing.

Important: 8.4.0 is NOT the last version of **RAV AntiVirus for Mail Servers**. For the latest product versions, always remember to visit http://www.ravantivirus.com/.



	Linux i386	Linux S/390	Linux (PowerPC)	Linux (SPARC)	FreeBSD	OpenBSD 2.8 *	OpenBSD 2.9 *	OpenBSD 3.x	Solaris i386	Solaris SPARC	UnixWare 7.1.1 *	NetBSD *	Win NT/2000/XP *	Mac OS X (new)
Sendmail	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ		Ⓘ		
Sendmail Libmilter	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ				
Qmail	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ		Ⓘ		
Postfix	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ		Ⓘ		Ⓘ
CommuniGate Pro	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ		Ⓘ	Ⓘ
Exim	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ		Ⓘ		
MS Exchange 5.5													Ⓘ	
MS Exchange 2000													Ⓘ	
DMail	Ⓘ		Ⓘ		Ⓘ				Ⓘ	Ⓘ			Ⓘ	Ⓘ
Courier	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ				Ⓘ	Ⓘ				Ⓘ

* Does not currently include the AntiSpam module

Further developments

GeCAD Software is currently working on developing antivirus solutions for other operating systems and platforms. For details on the developing process and other info about the characteristics of our products, please visit our website: http://www.ravantivirus.com.

Features of RAV AntiVirus for Mail Servers

Some of the distinctive features of **RAV AntiVirus for Mail Servers** are presented below:

- *Simple installation process*: The installation process is very simple and can be executed using an interactive install script (`install.sh`). If you want to manually install the product, you can find install instructions on the manufacturer's website;
- *Easy to configure and use:* **RAV AntiVirus for Mail Servers** is extremely easy to configure: options are available to order the actions to be taken by **RAV AntiVirus** when dealing with an infected file (**Clean, Move/Copy to Quarantine, Delete, Rename, Ignore, Reject**) or with a file containing suspicious code (**Move/Copy to Quarantine, Delete, Rename, Ignore, Reject**).
- *Complete antivirus protection:* **RAV AntiVirus for Mail Servers** scans all incoming and outgoing mail flow for the protected domains, removing malwares from all levels (subject, body, attachments). **RAV AntiVirus for Mail Servers** is also scanning archives inside archives, packed executables and multiple MIME-type encodings.
- *Enhanced e-mail traffic scanning:* **RAV AntiVirus for Mail Servers** is using the *Integrity Checker* technology: when the files are scanned for the first time, the detection engine creates a database with all the information it has gathered during the scanning process. When doing a second scan, only the new or changed files are scanned, therefore increasing the detection speed with over 50%.
- *Multi platform virus removal*: RAV Engine detects and removes all known Windows, Linux, Unix and DOS viruses, regardless of the operating system they're stored on or designed for.
- *Heuristic methods*: **RAV AntiVirus for Mail Servers** is using heuristic methods, to extend the protection offered to its users and act against new viruses and new versions of existing viruses.
- *Integration:* **RAV AntiVirus for Mail Servers** is an **integrated** suite, containing all the components (**antivirus**, **antispam**, **content filtering**, **group management**) in one single installation.
- *Enhanced virus scanning:* Mail attachments with multiple recipients are scanned only once, and not for all the recipients, therefore enhancing the scanning speed. Different technologies are also applied, in order to improve the speed and accuracy of the scanning process.
- *Content filtering:* All incoming/outgoing mail messages are scanned by **Subject, Attachment** and **Body**. This way, you can deny incoming mail messages containing suspicious objects and outgoing mail messages containing confidential information, reducing all types of threats to the very minimum. The available options for mail files matching the content filter are: **Move/Copy to Quarantine, Delete, Ignore, Reject**. For more details, please refer to the RAVMD configuration file section of this *User Guide*.
- *Group configuration:* The system's administrator can define different groups of users and specify different settings for these groups. **RAV AntiVirus for Mail Servers** will scan these differently configured groups according to their specified scanning settings.
- *Antispam:* Using its new bulk-mail module, **RAV AntiVirus for Mail Servers** can act as a customizable antispam program, blocking incoming spam and mail messages from certain addresses, according to the settings made by the server's administrator.

Features included in older versions (**Realtime Blackhole List** and **White/Black List**) help you ease your efforts. For more details, please refer to the Anti-spam Definitions section section of this *User Guide*.

- *Information leaks pre-empted:* **RAV AntiVirus for Mail Servers** restricts the mail flow according to defined parameters/patterns. You can configure specific groups (for example one group named Accounting, including all the computers from your company's accounting department) and set **RAV AntiVirus for Mail Servers** to deny all outgoing mail messages containing certain attachment types or confidential informantion, according to the patterns set by your mail server's administrator.
- *Warning mails:* You can configure **RAV AntiVirus for Mail Servers** to send instantly warning mails when a virus allert occurs to the sender of the message, to the receiver and/or to a third party (i.e. RAV Research Team or the server's administrator).
- *Intelligent Update:* The update process can be performed on demand or on a scheduled basis, according to the administrator's settings. The latest versions of **RAV AntiVirus for Mail Servers** (including engine updates and documentation) can be found on the following ftp site: ftp://ftp.ravantivirus.com/pub/rav/. Mirror sites for **RAV AntiVirus** products and updates are available all over the world. A complete and updated list of these mirror sites can be found at the following address: http://www.ravantivirus.com/kb/viewarticle.php?ano=1283. Starting with version 8.3.3, the updates for **RAV AntiVirus for Mail Servers** are executed via **ftp** and **http** (with and without proxy) using **ravupdate**. This utility allows "on the fly" update from mirrors, guarantees the integrity of the downloaded files and is self-updateable. For more details, please refer to the Update section of this *User Guide*.

Other features of RAV AntiVirus for Mail Servers

- Modular structure, allowing easy customization for every work environment;
- Extended protection (for new mail boxes);
- Scanning archives inside archives;
- Scanning for packed executables;
- Multiple MIME-type encodings;
- Technical support available by mail and phone for registered users.

Hardware and software requirements

Software requirements:

RAV AntiVirus for Mail Servers has been tested on the following flavors of Linux: Slackware 8.0, Mandrake 7.2, SuSE 6.4, RedHat 6.2, e-Smith (SME) and Debian Linux. The Linux kernel version (for the Linux flavors) should be 2.2 or later and the **zlib** version must be 1.1.3 or later. Fully functional installation kits for some MTAs are also provided for the following operating systems: Free BSD (version 4.1 or later), Open BSD (version 2.8, 2.9 and 3.x), Net BSD (version 1.5 or later), Solaris 8 (i386 and SPARC), Unixware 7.1.1 and Win32.

For Win32 OSs, the requirements are as following: Windows 2000 with Service Pack 2 or Windows NT 4 with Service Pack 4.

For **RAV AntiVirus for Mail Servers** running on MacOS X, the software requirements are: Mac OS X version 10.1.1 or higher.

RAV AntiVirus for Mail Servers has been comprehensively tested on the following MTAs:

- CommuniGate Pro version 3.4b1 or later.
- Courier version 0.38.0 or later.
- DMail version 3.0 or later.
- Exim version 3.33 or later.
- Postfix version 20000531 or later.
- Qmail version 1.03 or later.
- Sendmail version 8.11 or later (for libmilter version only).
- Sendmail (any version).

Hardware requirements:

RAV AntiVirus for Mail Servers will work on a computer meeting the minimal requirements for the installed Mail Server program.

For Intel-based systems, this means:

Processor:	Compatible Intel CPU, 150 Mhz or better
Memory:	32 MB of RAM or more.

For computers with SPARC processors, the hardware requirements are:

Processor:	SPARC V8 or later
Memory:	32 MB of RAM or more.

For **RAV AntiVirus for Mail Servers** products running on MacOS X, the hardware requirements are:

Processor:	powerpc (G3 or higher recommended)
Memory:	32 MB of RAM or more.

For **RAV AntiVirus for Mail Servers** products running on s390 platforms, the hardware requirements are:

Processor:	s390
Memory:	32 MB of RAM or more.

Note: In order to send warning messages, according to the settings made in your scanning configuration files, you must have a MTA running on the local machine and listening on port 25.

Registration procedure

Due to an extremely flexible licensing system, you can acquire different packages, depending on your needs of protection. **RAV AntiVirus for Mail Servers** is licensed *per domain*. The more domains you protect with **RAV AntiVirus for Mail Servers**, the less you pay per domain.

All the products included in **RAV AntiVirus** family are available under a license scheme with three different stages: *evaluation*, *registration* and *activation*. Each different stage has its own characteristics, described below.

Evaluation

The products included in **RAV AntiVirus** family are fully functional and their users benefit of complete update services and technical support for an *evaluation period* of 30 days. During this evaluation period, our potential customers should be able to evaluate **all** the functionalities and services provided by our products, in order to make a knowingly decision. For details regarding your rights and obligations pertaining to the usage of GeCAD Software's products during the evaluation phase, please read the *Evaluation license* section from the License Agreement.

During the evaluation period, each mail message scanned with a **RAV AntiVirus** product contains info about the number of days remaining for evaluation. After this evaluation period, the product expires. If you do not introduce a valid *Registration Code* during the evaluation period, you will not be able to use the product anymore. Introducing (anytime) a valid *Registration Code* extends *ad infinitum* the lifetime of the product.

RAV AntiVirus for Mail Servers can be used *free of charge* for *two domains* for an **evaluation period** of 30 days. During this evaluation period the product is fully functional and you can update free of charge its signatures database.

Registration

If you do not register your product within 30 days, you will no longer be able to use it for scanning your mail traffic (non even for the two domains you were protected in the evaluation period). In order to register your **RAV AntiVirus for Mail Servers**, you have to purchase it from an authorized RAV reseller/distributor. When you purchase **RAV AntiVirus for Mail Servers** from an authorized RAV reseller/distributor, you receive a *Registration Code*. If the purchasing involves physical delivery, the *License Certificate* will also be available. After registering your product, you have to *activate* it by installing an *Activation key* (please refer to the next section to see how you can do it). Activating your registered product is very important, because it offers you *free* updates, as well as technical support for *one year*.

Activation

The activation process is done free of charge for **all RAV AntiVirus** products, *upon request*, by visiting https://register.ravantivirus.com and filling in your End User information (or, you can fax this info to the number specified on the License Certificate). Subsequently, you will receive by e-mail an *Activation Key* and instructions for installation it. Installing this Activation Key enables you to use RAV AntiVirus for Mail Servers for the number of domains you have purchased the product for and offers you the advantage of the following *integrated services*:

- Free engine upgrades for one year;
- Free daily updates for one year;
- Full technical support for one year;
- Free virus alerts and security advisories.

You can extend the rights resulted from activating your product by purchasing annual update extensions available at special discounts.

Updates

Given the present rate of appearance of new viruses, an antivirus program becomes obsolete in a few weeks (sometimes even days). In order for any antivirus product (RAV AntiVirus for Mail Servers included) to insure an efficient protection, you must be periodically *update* its virus signatures database.

Our developing team is updating the signatures database of RAV Engine on a daily basis. The list with the latest virus additions to the signatures database of RAV Engine can be found at the following address: http://www.ravantivirus.com/pages/dldupdate.php?type=Daily.

As for the latest versions of RAV AntiVirus products, you can download them from the following ftp site: ftp://ftp.ravantivirus.com/pub/rav/.

Mirror sites for RAV AntiVirus products and updates are available all over the world. A complete and updated list of these mirror sites can be found in the following Knowledge Base article: http://www.ravantivirus.com/kb/viewarticle.php?ano=1283.

USER GUIDE

Part IV: Configuration files and man pages

In the following section you will find:

- The configuration file for **RAV AntiVirus for Mail Servers** (**ravmd.conf**);
- The **man** page for RAV mail scanning daemon (**ravmd**);
- The **man** page for **RAV AntiVirus** command line version (**ravav**);
- The **man** page for **ravupdate**;
- The **man** pages for the external filters for the MTAs supported by **RAV AntiVirus for Mail Servers**:
  - ✓ **ravcgate** – RAV external filter for CommuniGate Pro;
  - ✓ **ravcourier** – RAV external filter for Courier;
  - ✓ **ravdmail** – RAV external filter for DMail;
  - ✓ **ravexim** – RAV external filter for Exim;
  - ✓ **ravpostfix** – RAV external filter for Postfix;
  - ✓ **ravqmail** – RAV external filter for Qmail;
  - ✓ **ravsendmail** – RAV external filter for Sendmail;
  - ✓ **ravmilter** – RAV external filter for Sendmail-Milter.

These files are not meant to represent an exclusive documentation for **RAV AntiVirus for Mail Servers**, as additional documentation might also be found:

- In the *tar.gz* files containing the installation kit for the product you are using. Here you can find the *ReadMe, Install* and *Uninstall* files providing valuable information for the distinct operating system and MTA you are using.
- On the manufacturer's website (http://www.ravantivirus.com/). Here you can find additional "How To" files and "Frequently Asked Questions", the latest product updates, as well as documentation and information about our new products.

RAVMD configuration file

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 `/etc/opt/rav/` directory.

This document is part of the *User Guide for Mail Servers* and should be used only in connection with this *User Guide*.

Other documents containing valuable information are also 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 Guide* and in the `tar` or `tar.gz` files containing the setup programs;
- RAV Engine version 8.9 - *Release Notes* (available at: http://www.ravantivirus.com/rav/mailservers/documentation/notes/releasenotes840.txt

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 *Anti-spam 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 un-installing 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 8.4.0* section below.

What is new in ravmd 8.4.0

The main enhancement brought by **ravmd** version 8.4.0 is the integration of the new **anti-spam** functionality, based on a combination between the new bulk mail module (available starting with **ravmd** 8.4.0) 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/.

The new anti-spam module included in **ravmd** version 8.4.0 triggers important changes in this document.

- A new section called *Anti-spam definitions was* added after the *Warning Mails Message declaration* section, explaining how the new anti-spam module works.
- The parameters specific to the anti-spam module are detailed in the *Anti-spam 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: **anti-spam_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** are to be found in the `/etc/opt/rav/` folder, instead of the older `/usr/local/rav8/etc` location. The new anti-spam file is also to be found in the `/etc/opt/rav/` 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;

- 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.

USER GUIDE

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 *Anti-spam 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.

You can subscribe to these discussion lists by visiting our website http://www.ravantivirus.com/ (*RAV Discussion Lists* section) or by sending an empty mail message to: listname-subscribe@lists.ravantivirus.com. For more *Frequently Asked Questions*, please visit our website at http://www.ravantivirus.com/ or consult our *Knowledge Base* available at the following address: http://www.ravantivirus.com/kb.

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.

This section begins with the 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.

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**. 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 mail files matching the subject filter*: copy, ignore, reject, discard.
- *for files matching the attachment filter*: move, copy, delete, rename, ignore, reject, discard.
- *for mail files 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:
Clean	Ask **ravmd** to clean the infected file.	Infect
Move	Ask **ravmd** to move the file to quarantine (equivalent to `Copy` + `Delete` actions).	Infect, suspicious, attach match, content match
Copy	**ravmd** will copy the infected object to quarantine.	Infect, suspicious, subject match, attach match, content match
Delete	**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
Rename	The file will be renamed using the `rename_ext` extension specified in the configuration file.	Infect, suspicious, attach match
Ignore	The file is ignored, no action is taken and the e-mail is delivered.	Infect, suspicious, subject match, attach match, content match
Reject	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
Discard	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
Deliver	The original mail is delivered to its recipients even though it is tagged as Spam.	Bulk mail
Save	The bulk mail is saved in the Quarantine directory.	Bulk mail
Embed	Creates an embedded mail including a custom message and the bulk mail as attachment.	Bulk mail
Forward	New action available from version 8.4.1. The bulk mail tagged as Spam is forwarded to the mail addresses specified by the antispam `forward_to` parameter.	Bulk mail
add_header	Add a custom extra header to the bulk mail tagged as Spam.	Bulk mail
add_subject	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 eventually 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, enter the following command:

```
/etc/opt/rav/bin/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 scannable domains in a separate hash or normal file?

Answer: Create the `/etc/opt/rav/domains.list` file and/or 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 /etc/opt/rav/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 `warn.txt` file. How can I change the contents of the `warn.txt` file?

Answer: Please edit the `/etc/opt/rav/languages/English` file and customize the value of:

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

Then restart ravmd with:

```
kill -HUP `cat /var/opt/rav/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 to small to allow the entire `warn.txt` message to be displayed (see the answer to FAQ 3 above). The steps to be followed are:

- In `/etc/opt/rav/languages/english` use:

```
warn_txt_msg_english = "Your message here"

-in /etc/opt/rav/languages/english.equiv

warn_txt_msg = warn_txt_msg_english
```

- Restart **ravmd** with:

```
kill -HUP `cat /var/opt/rav/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*."

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.).

This information always included in a warning mail is based on *macros*. One macro is (for the purpose of this document) a stored template of instructions containing one piece on info from the string representing the warning mail. `FILE_NAME`, for instance, is an example of macro containing the name of the file causing the alert you're receiving. **ravmd** will replace this macro with the actual name of the trouble-making file.

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

`FILE_NAME`	The full path to the scanned file and its name.
`ATTACH_NAME`	The name of the scanned attachment.
`VIRUS_NAME`	The name of the virus discovered by **ravmd**.
`FROM_USER`	The mail address of the sender.
`ON_HOST`	The host **ravmd** is running on.
`TO_USER(S)`	The mail addresses of the receivers.

`SUBJECT` The mail's subject.

`QUARANTINE_NAME` The name of the file saved to RAV Quarantine using the *Move* or *Copy* action.

`SAVED_FILE_NAME` The name of the mail file saved to quarantine when `save_infected=yes` and/or `save_suspicious=yes`.

`HEADER_RECEIVED` `This macro is replaced with all the Received: lines` from the received mail's header (if this information exists). This helps you finding the infected machine more easily.

`HEADER` This macro is replaced with the entire received mail's header.

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 (optional info), the sender's name, the addressees' names and the virus name.

To eliminate any confusion, here are some explanations:

1. `QUARANTINE_NAME` *vs.* `SAVED_FILE_NAME`

`QUARANTINE_NAME` represents the name of the file 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 file saved to Quarantine when using the `save_infected=yes` and `save_suspicious=yes` parameters. 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 included if 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 don't I get warning messages when viruses are found?

Answer: You probably did not configure correctly **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 sensitive info from the victim's hard drive, and puts it in the subject of the email.

Answer: In `/etc/opt/rav/languages/english`, edit the 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 mail message

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 /etc/opt/rav/languages/your_language` define a variable, for example:

```
infected_attach = "The file ATTACH_NAME attached to mail (with subject:SUBJECT) sent by FROM_USER to TO_USER(S) has an unauthorized file extension"
```

- Then in `/etc/opt/rav/languages/your_language.equiv` use (in the `_attachment_filter_warning_messages` section):

`infected_msg = infected_attach` instead of

```
infected_msg = infected_m_english
```

- Restart **ravmd** with:

```
kill -HUP `cat /var/opt/rav/run/ravmd.pid`
```

FAQ 8: Stopping update notifications

Question: How do I stop update notifications?

Answer: To stop the update notifications, in `/etc/opt/rav/bin/ravupdate.sh` change this:

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

#VERBOSE="silent"

VERBOSE="noisy"

#VERBOSE="errors"
```

to:

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

#VERBOSE="silent"

#VERBOSE="noisy"

VERBOSE="errors"
```

Anti-spam Definitions section

The **Anti-spam** 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 anti-spam module of **RAV AntiVirus for Mail Servers** are described in the *Anti-spam parameters* sub-section of this document.

A default configuration is included in the `/etc/opt/rav/antispam` file.

The Anti-spam functionality is based on the new bulk mail module, 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* messages (**Static Black List**).

If the mail just received by the RAV-protected mail server comes from an 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 anti-spam 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 address in the **Static Black List**, the mail is automatically rejected.

If the 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 customize **ravmd** not to use **WBL** or **RBL** feature. In this case, **ravmd** scans directly the received mail, using the options defined for the corresponding group.

Assuming the mail is passing OK through the virus-scanning process, the anti-spam search is executed. **ravmd** is looking for patterns known to be specific to spammers on the **Header** and **Body** levels of the mail message. Depending on its specifics, the mail message is classified in one of the following *accuracy levels*: **Low**, **Medium**, **High** and **Very High**.

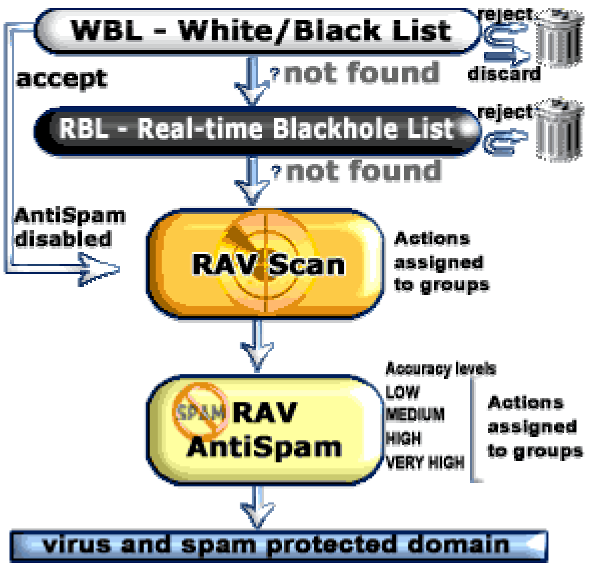
A „low" accuracy level means that **ravmd** reports the suspected mail message as spam even if only few spam patterns exist. This means that you might get some „false alarms", but no spam messages will pass by **ravmd**.

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**. The actions specific for the Anti-spam module of **ravmd** are:

- **save** (the mail is saved in the **Quarantine** folder, for further analysis, before any other action is executed on the respective mail);
- **embed** (the original mail 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 original mail);
- **add_subject** (a user-defined variable is affixed in the **Subject** field);
- **discard/reject/deliver** (the mail is discarded/rejected/delivered).

Here is the checklist to be followed when using the Anti-spam module of **ravmd**:

- define in ravmd.conf (after the Regular Expression Declarations, Action Definitions and Warning messages sections and before the Group configuration section) the name of the section, flanked by two "@" symbols (i.e. `@bulk_detection_low@`);
- define 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 for the Quarantine folder.

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:

```
@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 = /var/opt/rav/bulk
```

For more details, please refer to the corresponding parameters in the Explaining the parameters 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 (`/etc/opt/rav/languages/eng`, for instance) 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) that have different mail addresses and/or domains, but share the same action 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 **RAV AntiVirus for Mail Servers**. 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.

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.

Defining other groups

A new group definition begins with the group name written between "`[]`". 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 smaller files.

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

```
_include /etc/opt/rav/groups/mygroup_file
```

and define that group in the `mygroup_file`:

```
[mygroup]

sender = user_1@domain.com
```

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.

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 paramater. 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`, `use_rbl`, `embed_clean_mail`, `embed_cleaned_mail`, `embed_unclean_mail`, `use_embedded_msg`, `use_embedded_warning`, `antispam_configuration`.

How do I configure separate anti-spam options for my groups

If you intend to use the anti-spam feature of ravmd, you should specify what *anti-spam configuration* you want to use for the targeted groups. You do that using the `antispam_configuration` parameter, described in the *Explaining parameters* section, under *Group-specific parameters* sub-section.

FAQ 9: 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:

- In the actions file use:

```
act_for_my_group = clean, delete, …
```

- At the end of ravmd.conf define the group:

```
[my_domain]

_include /etc/opt/rav/groups/my_domain
```

- Create the file `/etc/opt/rav/groups/my_domain` and edit it:

```
#from_host=a.com
```

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

FAQ 10: 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 `/etc/opt/rav/groups/global` use:

```
ravms_name = ravms

on_host = your.host.com

smtp_server = ip.address.of.host

ravms_full_name = displayed.name (RFC822)
```

FAQ 11: Global group configuration

Question: How can I configure the `[global]` group?

Answer: Please define in `/etc/opt/rav/groups/global` 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`.

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

FAQ 12: 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 **ravmd** 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. Please make the following changes:

In **ravmd.conf**, add at the end of the file:

```
[group1]

_include /etc/opt/rav/groups/group1
```

Then create the file `/etc/opt/rav/groups/group1` and edit it:

```
sender=sender@host.com

do_not_scan=yes
```

Then restart **ravmd** with:

```
kill -HUP `cat /var/opt/rav/run/ravmd.pid`
```

The Advanced Content Filtering feature

What is this

The *Advanced Content Filtering* is a powerful feature of **ravmd** helping you configure the product to answer your specific needs. Using this feature, you can define filters on different levels of mail messages (subject, body, attachment or name of attached file). You can afterwards define specific actions for the mail messages matching the criteria you specify. For instance, you can instruct **ravmd** to deny all incoming/outgoing mail messages containing user-defined strings in the mail's **Subject** field (i.e. "I love you") or in the mail's body (i.e. "confidential", "balance sheet") or deny all incoming/outgoing mails containing user-defined attachment 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 file names* levels;
- **User-defined strings** for searches executed on the *Body* level (starting with version 8.4.0 of **ravmd**; previous versions of **ravmd** have used **POSIX regular expressions** on the *Body* level).

The rules you define are processed in the following order:

- Subject,
- File names,
- Body.

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

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 13: Rejecting double extension files

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

Answer: Please use in **ravmd.conf**:

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

var_action = reject
```

In `[global]`:

```
filter_attachment var_regexp var_action
```

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

FAQ 14: Denying certain types of attachments

Question: How can I deny certain types of attachments?

Answer: The content filtering feature of RAV Antivirus for Mail Servers can help you if you want to deny a certain type of attachment from entering your company via email. The content filtering module of RAV AntiVirus for Mail Servers uses POSIX regular expressions to find a pattern in *subject* and *attached file names* and user-defined strings to find a pattern in *message body* (and attachment contents). The rules are processed in the following order: subject, attachments, body. If you define more rules for the same component, they will be processed in the same order they are specified.

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

In the `define_regular_expressions` section from **ravmd.conf**:

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

In the actions file define:

```
file_action = delete, reject
```

In the `/etc/opt/rav/groups/global` file define (wherever inside the file):

```
filter_attachment file_regexp file_action
```

To add other types of attachments separate them with a | symbol on 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 /var/opt/rav/run/ravmd.pid`
```

FAQ 15: Example of content filtering for mail subject

Question: Can I have an example of content filtering for mail messages containing one defined expression in the **Subject** field?

Answer: Yes, you can. Here you can find how to create a content filtering rule for mail messages containing the expression "xxx" in the **Subject** field. Please note that this feature is not available in **RAV AntiVirus for Mail Servers** running on Sendmail if the mail server is not Milter-enabled.

The following options are available for the mail messages matching the content filter: **move, copy, delete, 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 `_define_regular_expressions` section from ravmd.conf, define the expression `xxx`:

```
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 on the `regexp` definition.

Example:

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

- Edit the file `/etc/opt/rav/groups/global`.
- Specify the administrator's mail address for alerting him that a mail matching an expression from content filter has been entering in the company:

```
admin_addr=administrator@ravantivirus.com
```

- Activate the content filter 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 /var/opt/rav/run/ravmd.pid`
```

FAQ 16: Example of content filtering for mail body

Question: Can I have an example of content filtering for mail messages containing one defined expression in the mail bodies?

Answer: Yes, you can. Here you can find how to create a content filtering rule for mail messages containing the following expressions in their bodies: “confidential”, “salaries” and “balance sheet”.

For this scenario, the mail will not 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 `_define_regular_expressions` section from ravmd.conf, 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:

```
bodyconfidential_action = reject
```

This will not allow the mail messages containg the word “confidential” in their bodies to pass the content filtering module of ravmd.

- Edit the file `/etc/opt/rav/groups/global`.
- Specify the administrator's mail address for alerting him that a mail matching an expression from content filter has been entering in the company:

```
admin_addr=administrator@ravantivirus.com
```

- Activate the content filtering by adding the following line in the `global` file:

```
filter_subject bodyconfidential_string bodyconfidential_action
```

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

```
kill -HUP `cat /var/opt/rav/run/ravmd.pid`
```

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 one. 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 for Mail Servers.**

Note: During the evaluation period of 30 days you can set only two domains. Mails for/from other domains are not scanned. They are normally delivered.

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 acception 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: **RAV AntiVirus for Mail Servers** 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 bellow 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 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 the content filtering search).

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. **vvpack**, **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 filter match, attachment filter match, content filter match).

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: Parameter used for specifying the messages used to create the warning mails when a virus is found.

_subject_filter_warning_messages

Description: Parameter used for specifying the messages used to create the warning mails when the content filtering is enabled and the mail subject matches one of the defined rules.

_attachment_filter_warning_messages

Description: Parameter used for specifying the messages used to create the warning mails when the content filtering is enabled and an attached file name matches one of the defined rules.

_content_filter_warning_messages

Description: Parameter used for specifying the messages used to create the warning mails when the content filtering is enabled and the mail body or one of the attachments contains a string matching one of the defined rules.

warning_mail_subj = variable

Description: Parameter used for specifying the subject used in the warning mail.

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 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 contained 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 `/etc/opt/rav/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: Parameter used for specifying the string used in the warning mail when the 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 *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: Parameter 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."
```

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"
```

Anti-spam parameters

The parameters pertaining to the anti-spam module of RAV AntiVirus for Mail Servers are described below.

The default configuration is included in the `antispam` file that you can find in the `/etc/opt/rav` 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: `/var/opt/rav/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
```

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**. 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 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: `/var/opt/rav/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

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.

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. 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 the 8.3.3 version 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-yyyymmdd)
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 (yet 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-yyyymmdd)

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:

*timeout_per_file + timeout_per_mega * filesize/1Mb*

Accepted values: **10-600** (for **timeout_per_file**), respectively **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: `/var/spool/rav/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: `/var/spool/rav/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: `log_file_name = /var/spool/rav/log/group_name`.

Example:

```
log_file_name = /var/spool/rav/log/global

log_file_name = /var/spool/rav/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.

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 to filter the mail subject.

variable_1 is a regular expression defined in the Regular Expression Declaration section.

variable_2 is a variable defined in the **actions** file. If this parameter is not defined then the content filtering for subject 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. You can use here a regular expression, not just a simple string.

filter_attachment variable_1 variable_2

Description: This parameter is used to filter the names of the mail attachments.

variable_1 is a regular expression defined in the Regular Expression Declaration section.

variable_2 is a variable defined in the **actions** file. If this parameter is not defined then the content filtering for attachment names 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 any mail messages.

If a file cannot be deleted then the whole mail is rejected.

filter_content variable_1 variable_2

Description: This parameter is used to filter the mail body and the contents of the attachments.

variable_1 is a string defined in the Regular Expression Declaration section.

variable_2 is a variable defined in the actions file. If this parameter is not defined then the content filtering for mail body and the contents of the attachments 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 you 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 boddies 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 **RAV AntiVirus for Mail Servers** and *when*. If one of these parameters is not defined then the respective user category **will not** receive warnings. The valid keywords are:

Table 2: Keywords for warning messages.

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 *ALL* the previously defined rules (`found_virus`, `found_subject`, `found_attach` or `found_content`) are matched.

Please note that these values are not inherited from the `[global]` group. This way you can specify different warning policies for different groups.

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 whenever 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_receivers = 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 string matched by a content filtering rule.

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 address that will not be notified.

do_not_show = enumeration

Description: Parameter used for specifying the mail address 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. If this is the case, these options will help you solve the problem. Note that only the receiver's mail address is compared against the specified address.

These parameters have no *default values* (no comparisons 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 that will be notified when an infected or suspicious file has been detected. This 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.

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 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 /etc/opt/rav/languages/english.equiv

advertising_msg = my_advertising
```

- In `/etc/opt/rav/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: `/opt/rav/bin/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 addressees. 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` is 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 mails, before sending them.

Default value: `/var/spool/rav/embed_queue`.

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)

RAVMD file

NAME

ravmd - rav mail scanning daemon

SYNOPSIS

```
/etc/opt/rav/bin/ravmd  [-cdfhrtvugsLT]  [--config=config_file]
[--dump_conf=config_file]      [--foreground]      [--help]
[--rav8path=rav8basedir]  [--testconf=config_file]  [--version]
[--user=user_name]  [--group=group_name]  [--syslog]  [--license]
[--temp-path=directory]
```

DESCRIPTION

DEFINITIONS

"filter client"

A program that resides in the **/opt/rav/bin/** directory and whose name depends on your MTA. (i.e. **ravexim**, **ravsendmail**, **ravpostfix**, etc.). Its function is to hook the MTA's mail flux and pass each mail to **ravmd** for scanning. Depending on the response is receives from **ravmd**, the "filter client" will discard or transmit the respective mail.

ravmd is powered by the platform independent RAV Engine, so it can detect and clean **all** viruses detected by this (linux, win, macro, dos, trojan, hoax, etc.). The program can scan mail files in MIME format containing attachments encoded with: **base64, quoted-printable, uuencode, 7bit, 8bit.**

ravmd also supports:

- mail *content filtering* for the mail subject, attachment file names and message body; and
- an *antispam* functionality, based on the new bulk mail module, integrated in **ravmd** starting with its version 8.4.0, and older features like **Real-time Blackhole List** (RBL) or **White-Black List** (WBL), available in **ravmd** since version 8.3.3. For more details, please refer to the RAVMD configuration file section.

When **ravmd** starts, it loads its configuration from `/etc/opt/rav/ravmd.conf`. If there are some errors (i.e. missing or bad format of these files), **ravmd** exits with non-zero status. In all the other cases, **ravmd** starts in background and binds an UNIX socket (`/var/opt/rav/run/_ravmdcom`) and listens it for "**filter clients**" queries. When a "**filter client**" connects to this socket, **ravmd** forks and the child will process "**filter client**" commands.

If started with `--syslog`, the daemon uses the system mail info log file for logging. The following command should display that file:

```
cat /etc/syslog.conf |grep -e ^[^#].*mail\.[^acdenw] | \

awk '{print $2}'
```

If you would like to perform periodic updates to RAV Engine and virus signature files, you should use a scheduling daemon (cron, fcron, ucron...) to execute `ravupdate.sh`**, a** script located in `/etc/opt/rav8/bin/`. Please modify that script file to fit your configuration. We recommend configuring the scheduling process so that the update process is executed once or twice an hour.

Example for fcron:

```
su

fcrontab -e
```

Insert the following line to run `ravupdate.sh` every 30 minutes:

```
@ 30 /opt/rav/bin/ravupdate.sh
```

Example for cron:

```
su

crontab -e
```

Insert the following line to run `ravupdate.sh` every 30 minutes:

```
*/30 * * * * /opt/rav/bin/ravupdate.sh
```

OPTIONS

Arguments are mandatory for both long and sort options.

-c, --config=config_file

Use the config_file instead of ravmd.conf.

-d, --dumpconf=config_file

Print the configuration from `config_file` to stdout.

-f, --foreground

Run the daemon in foreground instead of background.

-h, --help

Display the help screen.

-r, --ravepath=engine_dir

Full path to RAV Engine directory. Default value for **engine_dir** is `/var/opt/rav`.

-t, --testconf=config_file

Test the specified `config_file` configuration file.

-v, --version

Display **ravmd** version.

-u, --user=user_name

-g, --group=group_name

Use **user_name** and **group_name** as real user and real group for **ravmd** processes. By default the current user *uid* and *gid* are used.

Important: For security reasons, when **ravmd** is executed as root, it is highly recommended to use these options in order to drop the superuser privileges to an unprivileged user.

-s, --syslog

Use **syslog** daemon instead of RAV logging system.

-L, --license

Show the current license.

-T, --temp-path=directory

The temporary directory used to unpack large mail attachments (i.e. archived files).

EXIT STATUS

On error it returns non-zero, else returns zero.

BUGS

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

SEE ALSO

ravmd.conf(5), ravav(8), sysklogd(8)

USER GUIDE

RAVAV configuration file

NAME

ravav - RAV AntiVirus command line version

SYNOPSIS

```
/opt/rav/bin/ravav [OPTION ...] TARGET [TARGET ...]
```

DESCRIPTION

ravav is a command line antivirus being able to detect and remove known and unknown computer viruses, Trojans and worms. It uses the same engine as all the RAV AntiVirus products, with daily updates available on our web site: http://www.ravantivirus.com/pages/dldupdate.php?type=Daily.

OPTIONS

The following parameters are currently recognized by ravav:

-h, --help

Print the help screen to the console.

-v, --version

Display ravav version.

-V, --virlist

Print viruses list.

-D, --decode_qto=file_name

Decode the specified file from RAV Quarantine.

--license=AuthorizationCode

Licenses **ravav** using the Authorization Code provided by your supplier, as a string.

-u, --update=engine|full

Start **RAV AntiVirus** updating process.

--host=host_name

Download files from host_name. Current **ftp** sites are:

ftp://ftp.ravantivirus.com/ (site located in Romania);

ftp://download.ravantivirus.com/ (site located in USA).

--ravepath=engine_dir

Full path to RAV Engine directory. Default value for **engine_dir** is `/var/opt/rav`.

--hostpath=dirname

RAV path on **ftp** host. Default value for **dirname** is `/pub/rav`.

--ftpuser=username

The user name used on the ftp connection. Default value for **ftpuser** is “ftp”

--ftppass=password

The password used on the ftp connection. Default value for **ftppass** is “rave@”.

--all

Scan all files (default).

--smart

Use smart scan mode.

--ask

Ask the user what settings to be used for scanning.

--clean

Clean viruses from infected objects.

--delete

Delete infected and suspicious files

--copy

Copy infected/suspicious objects to quarantine.

--move

Move infected/suspicious objects to quarantine.

--rename

Rename infected/suspicious using the extension defined with: -R option.

-A, --archive

Scan inside archives.

-M, --mail

Scan mail files.

-H, --heuristics=on|off

Control the heuristic scanning.

-I, --integrity_check=on|off

Enable/disable integrity checker.

-Q, --quarantine=dir_name

Specify the path to the Quarantine folder. Default value for **dir_name** is `/var/opt/rav/quarantine`.

-R, --rename_ext=extension

Specify the extension used for rename action. Default extension is “_??”.

-T, --temp-path=directory

Specify the temporary directory used by ravav to unpack large archived files.

-l, --listall

List all scanned files.

--report=filename

Report names of viruses found in the filename file.

--rptall

Include all scanned files in the filename file.

--append

append information to the filename file.

EXIT STATUS

ravav returns the status of the last executed action:

1	The file is clean.
2	Infected file.
3	Suspicious file.
4	The file was cleaned.
5	Clean failed.
6	The file was deleted.
7	Delete failed.
8	The file was successfully copied to quarantine.
9	Copy failed.
10	The file was successfully moved to quarantine.
11	Move failed.
12	The file was renamed.
13	Rename failed.
20	No TARGET is defined.
30	Engine error.
31	Syntax error.
32	Help message.
33	Viruses list.
34	The updating process was successfully completed.
35	The updating process failed.
36	Already updated.
37	The licensing process was successfully completed.
38	The licensing process failed.

BUGS

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

SEE ALSO

ravmd(8)

USER GUIDE

RAVUPDATE configuration file

NAME

ravupdate - Update utility for RAV AntiVirus for Mail Servers.

SYNOPSIS

```
ravupdate [-adfhStTvVY] [--help] [--ftp-passive=on|off] [--
usage] [--use-proxy=on|off] [--dump]
[--serverresponse] [--tries=number]
[--proxy=http://[user[:password]@]sever:port]
[--timeout=secs] [--verbose] [--version]
```

DESCRIPTION

With new viruses appearing daily, your antivirus software (no matter what company is producing it) will become obsolete in weeks or even days, being unable to protect you from new threats. Updating your antivirus software is therefore a pre-requisite in the fight against malwares.

ravupdate is a utility helping you securely update RAV AntiVirus for Mail Servers (ravmd). It has the following features:

- Works with the ftp and http (with and without proxy) protocols;
- Allows updating from mirrors;
- Guarantees the integrity of the downloaded files;
- Allows "on the fly" update;
- Auto-update;
- Default configuration.

Supported protocols

ravupdate currently works with the ftp and http (with and without proxy) protocols. If you have any suggestions concerning new protocols to be supported by ravmd, please send a mail to ravteam@ravantivirus.com .

FTP

The FTP protocol allows the update procedure to be executed via one ftp server or one HTTP proxy.

The following options are customizable:

- User name;
- Password
- Server;
- Transfer type (active/passive);
- Timeout;
- Number of tries (in case of failure – how many times **ravupdate** will retry to execute the update procedure before returning an error message).

HTTP

The FTP protocol allows the update procedure to be executed via HTTP (with or without proxy).

The following options are customizable:

- User name;
- Password
- Server;
- Transfer type (active/passive)
- Timeout

Number of tries (in case of failure – how many times ravupdate will retry to execute the update procedure before returning an error message).

Updating from mirrors

ravupdate is designed to work with different servers implementing a certain type of protocol. This functionality was made possible using a minimal set of characteristics from each protocol.

For example, for a FTP protocol the mirror must implement the following commands:

USER, TYPE, PASV, PORT, RETR, REST, ABOR, QUIT, CWD.

For HTTP, the web server (proxy) must support the 1.1 version of the HTTP protocol.

Integrity of the downloaded files

ravupdate guarantees the integrity of the updated files by downloading at first a file containing information about the package needing to be updated/downloaded. The file initially downloaded by **ravupdate** is called **infofile**.

The program uses the information contained in this infofile to find out what files will be updated. The initially downloaded file is also used to check the integrity of the other downloaded files. Only after all the files needed for the update procedure are correctly downloaded **ravupdate** moves to the next phase: installing the updated files.

"On the fly" update

If the product is working as the update procedure is undergone, **ravupdate** is trying its best to make sure that the impact of the update procedure on the programs using the updated files is minimal. For instance, in case the engine of **RAV AntiVirus for Mail Servers** is updated, **raupdate** sends a SIGHUP signal to **ravmd**, forcing it to reload the updated engine. Already launched programs will continue the scanning process using the old engine, but the new processes will make use of the updated engine. In case we are talking about updating the scanning daemon (that is changing however quite rarely), the daemon must be stopped and restarted using the updated program.

Auto-update

In case a bug is reported in **ravupdate** or a new version of this program is launched, the file containing info about the package to be updated also contains info about the version of **ravupdate**. The program will update itself, restart and start the update procedure for the selected package.

Default configuration

ravupdate is shipped with a default configuration that allows running the update procedure on most of the computers using **RAV AntiVirus for Mail Servers**. However, for an enhanced performance of **ravupdate** we advise you to customize the configuration of **ravupdate** according to your specifications.

USAGE

Definitions

Module

Data containing information about one or more file logically grouped to obtain a certain functionality in **RAV AntiVirus for Mail Servers**.

Package

Data (transmitted in one communication session) containing one or more modules.

Example:

RAV for Qmail is a package that might contain the following modules:

- **rav engine** – module containing the files from **rave** folder;
- **ravmd** – module containing **ravmd.conf** and the scripts allowing you to launch/stop/restart/reload **ravmd** while it is working.
- **ravqmail** – containing the external filter for qmail, linking **ravmd** to **qmail**.

Description

At the beginning, **ravupdate** downloads a file containing information about the package you want to be updated. Depending on the modules you selected to be updated and the information from this file, **RAV AntiVirus for Mail Servers** checks every file susceptible to be updated. A list containing the files needing to be updated is created and the downloading process for these files begins.

After checking the integrity of the downloaded files **ravupdate** begins the install process for each of the modules needing update, in a precise order. In the example given above, first the updated engine is installed and then the updated **ravmd** and the filer are installed.

The installation procedure contains three different stages. For some modules, depending on their peculiarities, one or two of these stages might be skipped. However, the order these stages are executed is always as follows:

- **Pre-install**. The purpose of this step is to insure the successful of the install procedure. During this pre-install phase usually the correct permissions are checked and it is established if some programs have to be shut down.
- **Install**. The downloaded files are copied to a temporary location allowing the product to operate correctly.
- **Post-install**. **RAV AntiVirus for Mail Servers** is restarted or, in case the restart is not necessary, the user is notified the update procedure ended successfully.

Example:

Here is a brief example of what each phase is doing in case of updating the **engine** module containing the scanning engine.

- During the **Pre-install** phase: **ravupdate** is checking if you have writing permissions for copying the downloaded files in the folder containing the old engine
- During the **Install** phase: **ravupdate** is copying the new files to **rave** folder;
- During the **Post-install** phase: **ravupdate** is sending a SIGHUP signal to **ravmd** (if active), forcing it to reload the new engine.

Security and ravupdate

As far as the process security is concerned, we have to keep in mind that, in most cases, the update is made for a product under execution. Therefore, it is very important to ensure the authentication and the integrity of the files being updated. **ravupdate** answers this demanding request using the following mechanism:

- The **infofile** is digitally signed using an asymmetric system. The public key is encoded in the update program. This update program is only using files signed with the corresponding private key.

- The **infofile** contains the following information about the files used in the update process:
    - The file's size;
    - The date of the last modification in the file;
    - The path of the file (on the server used for updating **RAV AntiVirus for Mail Servers**);
    - The path on the file (on the local disk, where the update is being executed);
    - A digital signature of the file.

The update process uses this info to check each file downloaded from the update mirror in terms of precision (*is this the file I have to download?*), integrity (*does the downloaded file have the size specified in the initially downloaded file*?) and identity (is the digital signature recognized?). Only if the checking is returning the expected values the update process itself is launched. If the checking procedure for the downloaded files fails, the updating process returns an error code. Usually, this means the file was not correctly downloaded from the mirror site or this site is just executing its resync procedure.

CONFIGURATION

ravupdate can be configured:

- using a configuration file;
- using command-line parameters.

Note: The command-line parameters are overwriting the parameters from the configuration file.

SECTION DESCRIPTION

The configuration file for **ravpudate** contains sections and the sections contain attributes, names and values.

The values can be: numbers, strings or Booleans.

The `[global]` section

The `[global]` section contains the attributes managing the general behavior of **ravupdate**:

CONFDIR

Description: String containing the install path for the configuration files.

Default value: `/etc/opt/rav.`

DATADIR

Description: String containing the install path for the data required by RAV AntiVirus programs.

Default value: `/var/opt/rav.`

INSTALL

Description: String containing the install path for RAV AntiVirus programs.

Default value: `/opt/rav`.

Note1: Please note that the update procedure will return an error message in case this folder does not exist or the user does not have writing privileges in this folder.

Note 2: The INSTALL parameter has no correspondent in the command-line.

PLATFORM

Description: String describing the platform on which you wish to execute the update procedure. The available options are: i686, ppc, sparc, etc.

The default value is usually correct.

Note 1: Selecting a different platform than the one on which the update procedure is executing might determine the update process to fail.

Note 2: The PLATFORM parameter has no correspondent in the command-line.

alarm

Description: Number specifying the time (in seconds) allowed to one process to end the update procedure.

Note 1: In case the program is not completed in the specified period of time, the program might return a corresponding error message.

Note 2: The **alarm** parameter has no correspondent in the command-line.

proxy

Description: String with the following format:
`http://proxy_server_name:port_number`

The `port_number` may be overlooked. **ravupdate** is using in this case the default value (80).

Note 1: In case you wish to use a proxy, the `use-proxy` parameter has to be configured to `Yes`.

Note 2: The **proxy** parameter from the *config* file can be overwritten from the command-line using the following parameters (in short or long form): `-p, --proxy=http://port:name`

proxy-auth

Description: String helping to authenticate the proxy user.

Default value: `None`. This is currently the only available value. Some other values will be implemented in future versions of **ravupdate**.

Note: The **proxy-auth** parameter can be overwritten from the command-line using: `-a, --proxy-auth=none|basic|digest`

serverresponse

Description: Boolean. When set to `Yes` the program will display the commands it sends and the answers it receives from the server.

Note 1: The **serverresponse** parameter is useful to establish the causes of an eventual failure of the update procedure.

Note 2: The **serverresponse** parameter can be activated from the command line using: `-S, --server-response`

temppath

Description: String containing a valid path for the folder used to save the temporary files downloaded from the mirror site.

Note 1: A file named `update.start` is created in the specified **temppath**. This file is used for synchronizing/blocking/using more than one update process. It is advisable that only update program has accessing right for this file.

Note 2: The **temppath** parameter has no correspondent in the command-line.

timeout

Description: Number specifying the time (in seconds) allowed for a writing/reading operation to end successfully.

Default value: 0 (no timeout).

Note 1: We do not advise you to set for this parameter values over 20.

Note 2: The **timeout** parameter can be overwritten from the command-line using: `-T, --timeout=nr`

tries

Description: Number specifying the number of tries allowed for the communication protocol before the update procedure fails.

Note 1: In case the value for tries is set to **0**, the **alarm** parameter is the one responsible for establishing the time allowed for one process to complete.

Note 2: The **tries** parameter can be overwritten from the command-line using:

```
-t, --tries=nr
```

use-proxy

Description: Boolean enabling/disabling the usage of the proxy server for one specific protocol.

Note: The **use-proxy** parameter can be enabled from the command-line using: `-Y, --use-proxy`

verbose

Description: Boolean allowing or not **ravupdate** to display more information about the process.

Note 1: The **verbose** parameter can be useful when debugging or studying the behavior of **ravupdate**.

Note 2: The **verbose** parameter can be enabled from the command-line using: `-v`

The `[ftp]` section

The `[ftp]` section contains the following parameters used for the configuring the **ftp** connections: **passive**, **password**, **port**, **username**.

passive

Description: Boolean establishing if the update program or the **ftp** server is initiating the connection.

Available values: **Off** or **On**.

Default value: **Off**.

A short discussion about the architecture of the **ftp** protocol is made here for better understanding the **passive** parameter.

The **ftp** protocol uses two connections for transferring files between the ftp server and the ftp client. The first connection is used for transmitting the commands and receiving the answers. The second connection is created for each transferred file. Closing this second connection usually has the significance of ending the file transfer.

For the purpose of the second connection, it is the ftp server or the ftp client that has to initiate the connection and to send information about itself, to allow the other to connect.

The **passive** parameter is managing this connection:

- When set to **On**, the update program is establishing the connection with the ftp server and is sending the information the ftp server needs for connecting.
- When set to **Off**, the ftp server is establishing the connection and send the client the available information in the first transmission.

Note 1: Some ftp servers are using passive connections, other ftp servers are using active connections. Please make sure you dispose of all the required information concerning the ftp server before configuring the **passive** parameter.

Note 2: The passive parameter can be overwritten from the command-line using: `-f, --ftp-passive=on|off`

password

Description: String used by the **ftp** server to identify the user executing the update process.

The *default value* for **password** depends on the version of the program you are using.

Note: The password parameter has no correspondent in the command-line.

port

Description: Number used to specify the default port used by the **ftp** protocol.

Default value: 21.

username

Description: String used by the **ftp** server to identify the user executing the update process.

Default value: **ftp**.

Note: The **username** parameter has no correspondent in the command-line.

The `[http]` section

The `[http]` section contains the following parameters used for the configuring the http connections: password, port, username.

password

Description: String used by the http server to identify the user executing the update process.

Default value: `ravupdate(snapshot-20020528)@ravantivirus.com`.

Note: The password parameter has no correspondent in the command-line.

port

Description: Number used to specify the default port used by the http protocol.

Default value: 80.

username

Description: String used by the http server to identify the user executing the update process.

Default value: anonymous.

Note: The **username** parameter has no correspondent in the command-line.

The `[host]` section

The `[host]` section contains information about the infofile.

host

Description: String representing the name or the IP address of the server used for updating RAV AntiVirus for Mail Servers.

Note: The host parameter has to be customized to the closest server.

infofile

Description: String containing the relative path to infofile.

protocol

Description: String containing the name of the protocol used for the updating process.

Default value: ftp.

Note: The protocol parameter has no correspondent in the command-line.

The `[modules]` section

engine

Description: Boolean specifying if the user wants to update only the engine of RAV AntiVirus for Mail Servers.

Available options for engine: On/Off.

Default value: On.

Important: The `[modules]` section will contain other parameters ('ravcgate', 'ravdmail', 'ravexim', 'ravmd', 'ravmilter', 'ravpostfix', 'ravqmail' and 'ravsendmail') for specifying if the modules for RAV clients will also be updated. These options in are not yet implemented. The *default value* for these parameters is Off.

EXIT STATUS

0 Update has been successfully completed.

1 There are no files to update - you have latest files.

2 Configuration error - check your rup.conf file

3 Another ravupdate process is running.

4 Can't find the temporary directory or the temporary directory does not exist.

5 System error (e.g fork, malloc etc).

6 Update error (other than above). It is recommended to manually run ravupdate with -S and -v to see what is wrong.

FILES

`/etc/opt/rav/rup.conf`

BUGS

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

SEE ALSO

ravmd(8)

RAVCGATE configuration file

NAME

ravcgate - external filter for CommuniGate Pro

SYNOPSIS

`/opt/rav/bin/ravcgate`

DESCRIPTION

This is the external filter program executed by CommuniGate Pro server in order to scan all messages for virus protection and/or content filtering. The program runs as a "filter client" for **RAV AntiVirus** mail scanning daemon (**ravmd**).

USAGE

- Set your domain names as values for the domain field in **ravmd.conf** (you can find it in `/etc/opt/rav/ravmd.conf`).
- Open your web browser and connect to the CommuniGate Pro's web administration interface. If you installed CommuniGatePro using the default values, type http://127.0.0.1:8010 in the address bar.
- Open **Settings->General->Helpers.**
- Check **Content filtering** and add the following path in **Program Path**:

`/opt/rav/bin/ravcgate`

- Click on the **Update** button.
- Go to **Settings->Rules** and create a new rule (i.e. **rav**) and choose **Action** as **External Filter.**
- Create **ravms** account (this account will be used for sending warning mails via the SMTP local port opened by CGate Pro):

Go to Accounts->Create Account ravms

Important: For more details on how to make a rule for an External Filter, visit http://www.stalker.com/CommuniGatePro/VirusScan.html#Scanning.

Note: ravmd can scan multiple files in parallel. To activate this facility you must select more processors in **SETTINGS->Queue->Message Enqueuer.**

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5), ravav(8)

RAVCOURIER configuration file

NAME

ravcourier - external filter for the Courier MTA

SYNOPSIS

```
/opt/rav/bin/ravcourier
```

DESCRIPTION

This is the external filter program executed by the Courier MTA in order to scan all messages for virus protection and/or content filtering. The program runs as a “filter client” for **RAV AntiVirus** mail scanning daemon (**ravmd**).

USAGE

- To install **ravcourier**:

```
/usr/lib/courier/sbin/filterctl start ravcourier
```

- To uninstall the **ravcourier**:

```
/usr/lib/courier/sbin/filterctl stop ravcourier
```

- To start **ravcourier**:

```
/usr/lib/courier/sbin/courierfilter start
```

Filter configuration

When **ravcourier** is started it reads the runtime parameters from the file:

```
/etc/opt/rav/ravcourier.conf
```

If the configuration file doesn't exist then the following default values will be used instead.

nthreads

Description: Working threads started in parallel.

Accepted values: between 1 and 128.

Default value: 20.

max_connections

Description: Maximum accepted connections.

Accepted values: between 1 and 256.

Default value: 100.

use_allfilters

Description: Install the filter in the '**allfilters**' directory. If you specify "No" here, the filter will be installed in the '**filters**' directory.

Default value: Yes.

filters_dir

Description: The full path to the filters root directory.

Default value: `/usr/lib/courier/var.`

queue_dir

Description: The full path to the queue directory.

Default value: "". This parameter must be used only if Courier is configured to use relative paths to its queue files.

Important: If you want to modify some of these parameters then you have to edit the `/etc/opt/rav/ravcourier.conf` file and reinstall the filter:

```
     /usr/lib/courier/sbin/filterctl stop ravcourier

     /usr/lib/courier/sbin/filterctl start ravcourier
```

FILES

`/etc/opt/rav/ravcourier.conf`

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5), ravav(8)

USER GUIDE

RAVDMAIL configuration file

NAME

ravdmail - external filter of ravmd for DMail.

SYNOPSIS

`/opt/rav/bin/ravdmail`

DESCRIPTION

This is the external filter program executed by DMail server in order to scan all messages for virus protection and/or content filtering. The program runs as a "filter client" for RAV AntiVirus mail scanning daemon (ravmd).

USAGE

In the DMail configuration file the following line must by added: `virus_robot /opt/rav/bin/ravdmail`. After adding this line, the SMTP server configuration has to be reloded: `tellsmtp reload`.

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5)

RAVEXIM configuration file

NAME

ravexim - **RAV AntiVirus** filter client” for **Exim.**

SYNOPSIS

```
/opt/rav/bin/ravexim $sender_address $recipients
```

DESCRIPTION

The program is executed by the **exim** system filter each time a new message is processed. **ravexim** used to scan **all** incoming/outgoing mails for virus protection and/or content filtering. It runs as a “**filter client**” for **RAV AntiVirus** mail scanning daemon (**ravmd**).

USAGE

The following parameters are **exim** internal variables:

$sender_address

The mail sender address.

$recipients

A list with the recipient addresses of a message.

EXIT CODES

The program returns some of the exit codes defined in the `sysexits.h` file:

0 - EX_OK

Successful termination.

64 - EX_USAGE

The command line parameters are not specified correctly.

66 - EX_NOINPUT

Cannot open a temporary file.

69 - EX_UNAVAILABLE

The sendmail executable is not found. You have to create a symbolic link to the exim executable: ln -s /usr/exim/bin/exim /usr/sbin/sendmail

71 - EX_OSERR

Cannot create a pipe or a new process.

73 - EX_CANTCREAT

Cannot create a temporary file.

74 - EX_IOERR

An error has occurred on a read/write operation.

75 - EX_TEMPFAIL

Temporary error: ravmd is not running or it doesn't temporarily accept connections.

77 - EX_NOPERM

The mail file is infected and its delivery is denied.

Note: All the other exit codes are those returned by the command: `sendmail -oMr mail-ok -i -f $sender $recipients`

BUGS

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

SEE ALSO

ravmd.conf(5), ravmd(8)

RAVPOSTFIX configuration file

NAME

ravpostfix - RAV AntiVirus “filter client” for Postfix.

SYNOPSIS

```
/opt/rav/bin/ravpostfix [-hvSsCctflug] [--help] [--version]
[--server_ip=IP_ADDRESS] [--server_port=PORT]
[--client_ip=IP_ADDRESS] [--client_port=PORT]
[--timeout=SECONDS] [--foreground] [--log_level=NUMBER]
[--user=user_name] [--group=group_name]
```

DESCRIPTION

This is an external filter program integrated with postfix MTA and used for scanning all incoming/outgoing messages for virus protection and/or content filtering. It implements the protocol described in the FILTER_README file from postfix snapshot >= 20000531. The program runs as filter client for RAV AntiVirus mail scanning daemon (ravmd).

USAGE

-h, --help

Displays the help screen.

-v, --version

Displays ravpostfix version.

-S, --server_ip=IP_ADDRESS

The computer IP address where ravpostfix is running. Default: 127.0.0.1

-s, --server_port=PORT

The Inet port on `server_ip` where ravpostfix is listening. Default: 10025

-C, --client_ip=IP_ADDRESS

The computer ip address where second postfix SMTP server is running. Default: **127.0.0.1**

-c, --client_port=PORT

The Inet port where second **postfix** SMTP server is listening. Default: **10026.**

-t, --timeout=SECONDS

The timeout in seconds used by ravpostfix for SMTP communication. Default: **120s.**

-f, --foreground

Runs the daemon in foreground instead of background (background is default value).

-l, --log_level=0|1|2

Control the logging details. Default: **1.**

-u, --user=user_name
-g, --group=group_name

Use **user_name** and **group_name** as real user and real group for ravpostfix processes. By default the current user *uid* and *gid* are used.

Important: For security reasons, when executing **ravpostfix** as root, it is highly recommended to use these options in order to drop the superuser privileges to an unprivileged user.

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5), ravav(8)

USER GUIDE

RAVQMAIL configuration file

NAME

ravqmail - RAV AntiVirus “filter client” for qmail.

SYNOPSIS

```
/opt/rav/bin/ravqmail
```

INSTALL

In order to install ravqmail you must have installed qmail-1.03 or newer in directory /var/qmail. You must stop qmail-send from sending it the SIGTERM signal. After this you must rename the original qmail-queue to qmail-queue.orig and make a symbolic link named qmail-queue to ravqmail:

```
mv /var/qmail/bin/qmail-queue /var/qmail/bin/qmail-queue.orig

ln -s /usr/local/rav8/bin/ravqmail /var/qmail/bin/qmail-queue
```

USAGE

After installing ravqmail you must start ravmd daemon. Then you can start qmail system.

UNINSTALL

In order to uninstall ravqmail you must stop qmail-send from sending it the SIGTERM signal. After this you must remove the link file qmail-queue and copy the original qmail-queue.orig in qmail-queue:

```
rm -f /var/qmail/bin/qmail-queue

mv /var/qmail/bin/qmail-queue.orig /var/qmail/bin/qmail-queue
```

For more details, please read the Notes section bellow.

FILES

After the installation process `/opt/rav/bin/ravqmail` must have the same owner, group and permissions as `/var/qmail/bin/qmail-queue.orig`. The `/var/opt/rav/tmp` and `/var/opt/rav/run` directories must have the same owner and group as `/var/qmail/bin/qmail-queue.orig` and the permissions set to

USER GUIDE

0750.

EXIT CODES

The same as **qmail-queue(8)**.

NOTES

There are some programs or scripts automating the starting and stopping processes. If you are using one of them it might be wrong to send **SIGTERM** signal to **qmail-send** instead of using them to stop **qmail-send**, since these programs might respawn the **qmail-send** during the installation process therefore your **qmail** won't be able to placed mails in its queue.

When a mail is handled by **qmail** first it is put in queue by using the **qmail-queue** program. Because in the installation process the original **qmail-queue** is replaced with **ravqmail** the mail is first scanned and after that ravqmail pass the mail to the **qmail-queue.orig** and replies with **qmail-queue.orig** exit code.

If **ravmd** is not started or **ravqmail** can't communicate with **ravmd**, it doesn't pass the mail to the original **qmail-queue.orig** and will exit with error code **71**, meaning that the mail server temporarily refuses to send the messages to any of the recipients.

If **ravmd** is configured to reject the mails that cannot be cleaned **ravqmail** doesn't pass the mail to **qmail-queue.orig** and exits with error code 31 meaning that the mail server permanently refuses to send the message to any recipients. If there is a license limit **ravmd** will not scan the mails not belonging to the domains specified in the configuration file. These mails will be passed **unchanged** to **qmail-queue.orig**.

BUGS

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

SEE ALSO

ravmd.conf(5), ravmd(8), qmail-queue(8)

RAVSENDMAIL configuration file

NAME

ravsendmail - RAV AntiVirus “filter client” for Sendmail.

SYNOPSIS

```
/opt/rav/bin/ravsendmail
```

DESCRIPTION

The program will be executed by Sendmail MTA as a local mailer. It is used to scan all incoming/outgoing mails for virus protection and/or content filtering. The program runs as a “filter client” for RAV AntiVirus mail scanning daemon (ravmd).

USAGE

The following parameters are sendmail internal macros.

$h

The recipient host.

$f

The sender’s mail address.

$u

The recipient mail address.

$I

Queue ID used to identify a mail file in the queue.

FILES

```
/etc/mail/sendmail.cf
```

NOTES

This product uses two sendmail executables running at the same time. It can introduce a significant delay in the mail delivery process depending on how often the queues are processed. This is why it is highly recommended to use RAVMilter instead of RAVSendmail.

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5), ravav(8), sendmail(8)

RAVMILTER configuration file

NAME

RAVMilter - **RAV AntiVirus** "filter client" for sendmail with **libmilter**.

SYNOPSIS

```
RAVMilter [-dV] [-C sendmail.cf] [-g 0|1|2|3] [-X[on|off]]
[--configuration=sendmail.cf] [--daemon] [--debug=0|1|2|3]
[--addheader[=on|off]] [--help] [--usage] [--version]
```

DESCRIPTION

This is an external filter program used by **sendmail** (compiled with **libmilter** feature) MTA in order to scan **all** the incoming/outgoing messages for virus protection and/or content filtering. It uses the **libmilter feature** included in sendmail version 8.11 or later. The program runs as a client for **RAV AntiVirus** mail scanning daemon (**ravmd**).

USAGE

-d, --daemon

Run as daemon.

-g, --debug=0|1|2|3

Enable debug features (0->silent, 3->verbose).

-C, --configuration=sendmail.cf

Read connection information from this file. See the **Notes** section bellow.

-X, --addheader[=on|off]

Add or not **X-Version** header to each message scanned.

-V, --version

Print to **stdout** RAVMilter version and exits.

--help

Print the help screen to the console.

--usage

Display information about RAVMilter usage.

NOTES

The **Sendmail** version must be 8.11 or later to compile the **libmilter** feature of **sendmail**. Please read the `libmilter/README` file from the **sendmail** package for instructions on how to compile **sendmail** with this feature. After compiling **sendmail** and installing it with the **libmilter** feature, you can proceed with the instalation of **ravmd**.

To configure **RAVMilter**, you must add the following two lines into `/etc/mail/sendmail.mc`:

```
define(`_FFR_MILTER', `true')

INPUT_MAIL_FILTER(`RAVMilter',`S=local:/var/opt/rav/run/
ravmilter.sock, F=R, T=S:10s;R:5m;E:5m')
```

After that, you must generate `/etc/mail/sendmail.cf` with the following command:

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

When RAVMilter starts it will parse `/etc/mail/sendmail.cf` to find out what socket to bind (in our example it will bind the UNIX socket `/var/opt/rav/run/ravmilter.sock`).

BUGS

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

SEE ALSO

ravmd(8), ravmd.conf(5), ravav(8), sendmail(8)

USER GUIDE

Appendix A: Bug Report Form

Although we have extensively tested **RAV AntiVirus for Mail Servers**, some bugs may get by us, or you may have incompatible hardware or software that we did not test.

If you experience any problems with **RAV AntiVirus for Mail Servers**, please print out this form and mail it to:

GeCAD Software S.R.L.
223, Mihai Bravu Blvd, 3rd district
Bucharest, ROMANIA,

or fax it at +40-21-3217803.

Alternatively, copy the **Bug Report Form** to your word processor, fill in the blanks and email the text file to support@ravantivirus.com or to betatest@ravantivirus.com (when the product you are reporting the bugs for has been beta released).

Thank you!

Today's Date: ___/___/___ (MM/DD/YY)
Title: [ ] Mrs. [ ] Miss [ ] Ms [ ] Mr. [ ] Other:_____
Name:
Company (if applicable):
Mail:
Address:
City:
State/Province:
ZIP/Postal Code:
Country:
Telephone (with Area & Country Codes):
Fax (with Area & Country Codes):

Program Name: RAV **AntiVirus for Mail Servers** version:__________ Registration Code:______________

Bugs, suggestions & comments:

(Please be as specific as possible about bugs. If we cannot duplicate your problem, we cannot help you fix it. Before sending a Bug Report, please read carefully the product documentation.)

SYSTEM CONFIGURATION

Type of CPU:

CPU frequency:

OS version:

Memory:

Hard disk:

Other:

USER GUIDE

Appendix B: Index