Product Guide

VirusScan® for UNIX

version 6.00.0

McAfee®
System Protection

Industry-leading intrusion prevention solutions

McAfee®

Contents

1 Introducing VirusScan® for UNIX

VirusScan® for UNIX detects and removes viruses on UNIX-based systems. This section describes:

- Product features
- What's new in this release
- Using this guide
- Getting product information
- Contact information

Product features

The scanner runs from a command-line prompt, and provides an alternative to scanners that use a graphical user interface (GUI). Both types of scanner use the same anti-virus software. The scanner acts as an interface to the powerful scanning engine — the engine common to all our security products.

Although a few years ago, the UNIX operating system was considered a secure environment against potentially unwanted software, it is now seeing more occurrences of software specifically written to attack or exploit security holes in UNIX-based systems. Increasingly, UNIX-based systems interact with Windows-based computers, and although viruses written to attack Windows-based systems do not directly attack UNIX systems, the UNIX system can unknowingly harbor these viruses, ready to infect any client that connects to it.

When installed on your UNIX systems, VirusScan® for UNIX becomes an effective solution against viruses, Trojan-horse programs, and other types of potentially unwanted software.

The command-line scanner enables you to search for viruses in any directory or file in your computer *on demand* — in other words, at any time. The command-line scanner also features options that can alert you when the scanner detects a virus or that enable the scanner to take a variety of automatic actions.

When kept up-to-date with the latest virus-definition (DAT) files, the scanner is an important part of your network security. We recommend that you set up a security policy for your network, incorporating as many protective measures as possible.

What's new in this release

This release of VirusScan® for UNIX includes the following new features or enhancements:

- V2 DATs support — Supports the latest version of the anti-malware DAT files providing improved detection rates, smaller file downloads, and support for Engine component updates.
- Multi-threaded scanning support — Supports scanning in multiple threads using the THREADS switch. This results in improved performance; particularly on multi-core systems.
- XML reports — Supports the generation of XML reports using the XMLPATH switch. This facilitates easier automatic processing of scan results.

Using this guide

This guide provides information on configuring and using your product. These topics are included:

- *Introducing VirusScan® for UNIX* on page 4 (This section)
  An overview of the product, including a description of new or changed features; an overview of this guide; McAfee contact information.
- Detailed instructions for installing the software.
- Descriptions of product features.
- Detailed instructions for configuring and deploying the software.
- Procedures for performing tasks.

Audience

This information is intended primarily for two audiences:

- Network administrators who are responsible for their company's anti-virus and security program.
- Users who are responsible for updating virus definition (DAT) files on their workstations, or configuring the software's detection options.

Conventions

This guide uses the following conventions:

Bold Condensed	All words from the interface, including options, menus, buttons, and dialog box names. **Example:** Type the **User** name and **Password** of the appropriate account.

`Courier`	The path of a folder or program; text that represents something the user types exactly (for example, a command at the system prompt).
	Examples:
	The default location for the program is:
	`C:\Program Files\McAfee\EPO\3.5.0`
	Run this command on the client computer:
	`scan --help`

Italic	For emphasis or when introducing a new term; for names of product documentation and topics (headings) within the material. **Example:** Refer to the *VirusScan Enterprise Product Guide* for more information.
Blue	A web address (URL) and/or a live link. **Example:** Visit the McAfee website at: `http://www.mcafee.com`
<TERM>	Angle brackets enclose a generic term. **Example:** In the console tree, right-click <SERVER>.
	**Note:** Supplemental information; for example, another method of executing the same command.
	**Tip:** Suggestions for best practices and recommendations from McAfee for threat prevention, performance and efficiency.
	**Caution:** Important advice to protect your computer system, enterprise, software installation, or data.
	**Warning:** Important advice to protect a user from bodily harm when using a hardware product.

Getting product information

Unless otherwise noted, product documentation comes as Adobe Acrobat .PDF files, available on the product CD or from the McAfee download site.

Product Guide — Introduction to the product and its features; detailed instructions for configuring the software; information on deployment, recurring tasks, and operating procedures.

Help — Product information in the Help system that is accessed from within the application on its *man* pages.

Release Notes — *ReadMe*. Product information, resolved issues, any known issues, and last-minute additions or changes to the product or its documentation.

License Agreement — The McAfee License Agreement booklet that includes all of the license types you can purchase for your product. The License Agreement presents general terms and conditions for use of the licensed product.

Contacts — Contact information for McAfee services and resources: technical support, customer service, Security Headquarters (AVERT), beta program, and training.

Contact information

Threat Center: McAfee Avert® Labs http://www.mcafee.com/us/threat_center/default.asp

Avert Labs Threat Library
http://vil.nai.com

Avert Labs WebImmune & Submit a Sample *(Logon credentials required)*
https://www.webimmune.net/default.asp

Avert Labs DAT Notification Service
http://vil.nai.com/vil/signup_DAT_notification.aspx

Download Site http://www.mcafee.com/us/downloads/

Product Upgrades *(Valid grant number required)*

Security Updates (DATs, engine)

HotFix and Patch Releases

- **For Security Vulnerabilities** *(Available to the public)*
- **For Products** *(ServicePortal account and valid grant number required)*

Product Evaluation

McAfee Beta Program

Technical Support http://www.mcafee.com/us/support/

KnowledgeBase Search
https://kc.mcafee.com/

McAfee Technical Support ServicePortal *(Logon credentials required)*
https://mysupport.mcafee.com/eservice_enu/start.swe

Customer Service

Web
http://service.mcafee.com/
https://secure.nai.com/apps/support/customer_service/request_form.asp

Phone — US, Canada, and Latin America toll-free:
+1-888-VIRUS NO or **+1-888-847-8766** Monday – Friday, 8 a.m. – 8 p.m., Central Time

Professional Services

Enterprise: http://www.mcafee.com/us/enterprise/services/index.html

Small and Medium Business: http://www.mcafee.com/us/small/index.html
http://www.mcafee.com/us/medium/index.html

2 Installing VirusScan® for UNIX

We distribute the VirusScan® for UNIX software in two ways — on a CD, and as an archived file that you can download from our website or from other electronic services.

After you have downloaded a file or placed your disk in your CD drive, the installation steps are the same for each type of distribution version.

Review the *Installation requirements* on page 10 to verify that the software will run on your system, then follow the installation steps.

About the distributions

VirusScan® for UNIX software comes in several distribution versions, one for each supported operating system.

- IBM AIX 5.2, 5.3 and 6.1 for RS6000 with the latest Maintenance Packages installed.
- FreeBSD 6.1 and 7.0 for Intel (x86) 32-bit with legacy compatibility library libc.so.3 installed.
- Hewlett-Packard HP-UX 11.0, 11i, 11iv2 and 11iv3 for PA-RISC with the latest Standard HP-UX patch bundles installed.
- Linux for Intel 32-bit distributions with libstdc++.so.5.0.5 or later installed.
- Linux for Intel 64-bit distributions shipping with 2.6 production kernel, with libstdc++.so.6 installed.
- Sun Microsystems Solaris for SPARC versions 8, 9 and 10 (32 and 64-bit) with the latest Solaris OS Recommended Cluster installed.
- Sun Microsystems Solaris for X86 versions 10 (32 and 64-bit) with the latest Solaris OS Recommended Cluster installed.

For current information about the distribution versions, refer to the Release Notes.

If you install VirusScan® for UNIX software from CD, each version is in its own directory. Each distribution has its own installation script.

Installation requirements

To install and run the software, you need the following:

- The correct version of the UNIX distribution that you require, installed and running correctly on the target computer. See *About the distributions* on page 9 for information.
- 64 MB of free hard disk space for a full installation.
- 128 MB of hard disk space reserved for temporary files.
- A minimum of 128 MB RAM is required, 256 MB IS recommended.
- A CD drive, if you are not downloading the software from a website.

Other recommendations

- To install the software and perform on-demand scan operations of your file system, we recommend that you have root account permissions.
- To take full advantage of the regular updates to `DAT` files from our website, you need an Internet connection, either through your local area network, or via a high-speed modem and an Internet Service Provider.

Installing the software

This section shows how to install the software on any distribution. To install a specific distribution, substitute the correct file name for the distribution file.

To start the installation script:

1. Download the appropriate VirusScan® for UNIX software distribution from our website or insert the installation CD.

   If you are using the installation CD to obtain the software, you can mount the CD on to the file system.

2. Copy the distribution file to a directory on your system.

   

   We recommend that you use a separate (possibly a temporary) directory — not the directory where you intend to install the software.

3. Change the directory to that containing the distribution file. Use `cd`.

4. Type this line at the command prompt to decompress the file:

   ```
   gzip distribution-file | tar -xf -
   ```

   Here, *`distribution-file`* is the file you copied in Step 2.

5. Type this line at the command prompt to execute the installation script:

   `./install-uvscan` *installation-directory*

   Here, the *`installation-directory`* is the directory where you want to install the software.

If you do not specify an installation directory, the software is installed in `/usr/local/uvscan`.

If the installation directory does not exist, the installation script asks whether you want to create it. If you do not create the installation directory, the installation cannot continue.

6 The installation script asks whether you want to create symbolic links to the executable, the shared library and the man page. Type `Y` to create each link, or `N` to skip the step.

We recommend that you create these links. Otherwise, you will need to set one of the following environment variables to include the installation directory:

Table 2-1 Environment variables

Distribution	Variable
IBM AIX	`LIBPATH`
FreeBSD v4/v5/v6	`LD_LIBRARY_PATH`
HP-UX	`SHLIB_PATH`
Linux	`LD_LIBRARY_PATH`
Linux 64-bit	`LD_LIBRARY_PATH`
Solaris SPARC	`LD_LIBRARY_PATH`
Solaris X86	`LD_LIBRARY_PATH`

The program also looks in the `/usr/lib` or `/lib` directory or the current directory for the shared library.

The installation program copies the program files to your hard disk, then scans your home directory.

If the software discovers a virus, see *Handling viruses* on page 18 to learn about the actions you can take.

If the installation fails, see *Troubleshooting during installation* to learn about possible errors and suggested courses of action.

Troubleshooting during installation

The following table lists the most common error messages returned if the installation fails. The table also suggests a likely reason for the error and recommends any solutions.

Table 2-2 Error messages

Error	Cause or action
`Failed to create install_dir`	Verify that you have permission to create the installation directory.
`Cannot write to install_dir`	Verify that you have permission to write to the installation directory.
`The install_dir exists, but is not a subdirectory`	Choose another installation directory.
`<file> is missing`	The file might not exist.
`<file> is not correct`	The file did not install correctly.

Testing your installation

After it is installed, the program is ready to scan your computer for infected files. You can run a test to determine that the program is installed correctly and can properly scan for viruses. The test was developed by the European Institute of Computer Anti-virus Research (EICAR), a coalition of anti-virus vendors, as a method of testing any anti-virus software installation.

To test your installation:

1 Open a standard text editor, then type the following line:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

The line must appear as *one line* in the window of your text editor.

2 Save the file with the name EICAR.COM. The file size will be 68 or 70 bytes.

3 Type the following command to scan the EICAR.COM file:

```
uvscan -v eicar.com
```

When the program examines this file, it reports finding the EICAR test file, but you will not be able to clean or rename it.

This file is *not a virus* — it cannot spread or infect other files, or otherwise harm your computer. Delete the file when you have finished testing your installation to avoid alarming other users. Please note that products that operate through a graphical user interface do *not* return this same EICAR identification message.

4 When you have finished testing your installation, delete the test file to avoid alarming other users.

If the software appears not to be working correctly, check that you have Read permissions on the test file.

Troubleshooting when scanning

The following table lists the most common error messages returned if the `uvscan` program fails when scanning. The table also suggests a likely reason for the error and recommends possible solutions.

Table 2-3 Program messages

Program message	Remedy
`Cannot find shared object`	■ AIX — Install the correct version of -xlC.rte. The program does not run on versions earlier than 4.0 ■ HP-UX — Install the aCC run-time patch.
`Unable to find shared library`	Set the appropriate environment variable: ■ For AIX, use `LIBPATH`. ■ For HP-UX, use `SHLIB_PATH`. ■ For Solaris, FreeBSD and Linux, use `LD_LIBRARY_PATH`.
`Cannot execute: permission denied`	Check the file permissions. Incorrect file permissions can prevent the program running correctly. All executables (including the shared libraries) must have read and execute permissions (`r_x`), but we recommend `rwxr_xr_x` All DAT files must have read permissions.
`Missing or invalid DAT files`	Re-install the DAT files.
`The program has been altered; please replace with a good copy`	Re-install from the original media; the program might be infected.

Removing the program

A script is installed at the same time as the VirusScan® for UNIX software, which enables you to remove the product quickly and easily.

To remove the product from your system:

1 Run the script `uninstall-uvscan`, which is in the VirusScan® for UNIX program directory. For example, type the following command at the command prompt:

```
/usr/local/uvscan/uninstall-uvscan
```

2 Delete the script `uninstall-uvscan` from the program directory to remove the program completely from your system.

If you created your own links to the program and a shared library path when you installed the software, you must remove those links yourself.

Removing the software leaves your computer unprotected against threats. Remove the product only when you are sure that you can upgrade quickly to a new version.

If you are an administrator, ensure that your users cannot accidentally remove their VirusScan® for UNIX software.

3 Using VirusScan® for UNIX

VirusScan® for UNIX provides virus scanning from a command line. This section describes how to use its features and customize the program to meet your needs.

The following features offer optimum protection for your computer and network:

- On-demand scanning options let you start a scan immediately or schedule automatic scans.
- Advanced heuristic analysis detects previously unknown macro viruses and program viruses.
- Updates to virus definition files and to program components ensure that the program has the most current scanning technology to deal with viruses as they emerge.

Later sections in this guide describe each of these features in detail.

Running an on-demand scan

You can scan any file or directory on your file system from the command line by adding options to the basic command.

Only the Intel-based FreeBSD and Linux distributions of the VirusScan® for UNIX program can scan for boot-sector viruses.

When executed without options, the program displays a brief summary of its options. When executed with only a directory name specified, the program scans every file in that directory only, and issues a message if any infected files are found. The options fall into the following main groups:

- **Scanning options —** determine how and where the scanner looks for infected files. See page 23.
- **Response options —** determine how the scanner responds to any infected files. See page 26.
- **General options —** determine how the scanner reports its activities. See page 28.

Each group of options appears in its own table with a description of its function. See *Choosing the options* on page 22 for details.

Command-line conventions

Use the following conventions to add options to the command line:

- Follow the syntax correctly. The UNIX operating system is case-sensitive.
- Type each option in lower case and separate each with spaces.
- Do not use any option more than once on the command line.
- Type single consecutive options as one option. For example, instead of typing this:

  ```
  -c -r --one-file-system
  ```

  you can type this:

  ```
  -cr --one-file-system
  ```

- To start the program, at the command prompt, type:

  ```
  uvscan
  ```

  (This example assumes that the scanner is available in your search path.)

- To have the program examine a specific file or list of files, add the target directories or files to the command line after `uvscan`. You can also create a text file that lists your target files, then add the name of the text file to the command line. See *Configuring scans* on page 16.

  By default, the program examines all files, no matter what their extensions. You can limit your scan by adding only those extensions you want to examine to the command line after the `--extensions` option, or you may exclude certain files from scans with the `--exclude` option. See *Choosing the options* on page 22 for details.

General hints and tips

The following examples assume that the scanner is available in your search path.

- To display a list of all options, with a short description of their features, type the command:

  ```
  uvscan --help
  ```

- To display a list of all the viruses that the program detects, type the command:

  ```
  uvscan --virus-list
  ```

- To display information about the version of the program, type the command:

  ```
  uvscan --version
  ```

- To scan all subdirectories within a directory with maximum security, type the command:

  ```
  uvscan -r --secure target
  ```

- To ensure maximum protection from virus attack, you must regularly update your DAT files. See *Preventing Infections* on page 32 for details.

Configuring scans

Instead of running each scan with all its options directly from the command line, you can keep the options in a separate text file, known as a *task file*. In the file, you can specify the actions that the scanner must take when a virus is detected. This allows you to run complete scans with ease, and at any time; you need only specify the files or directories that you want to scan.

To configure a scan:

1. Choose the command options that you want to use.

   See *Choosing the options* on page 22 for a description of available options.

2. Type the command options into a text editor just as you might on the command line.
3. Save the text as a file — the task file.
4. Type one of these lines at the command prompt:

   `uvscan --load` *file target*

   `uvscan --config` *file target*

   Here, *file* is the name of the task file you created, and *target* is the file or directory you want to scan.

   If the scanner detects no virus infections, it displays no output.

To learn how to specify the options, see *Command-line conventions* on page 15.

The following examples show how you can configure scans using task files. The examples assume the scanner is available in the search path.

Example 1

To scan files in the `/usr/docs` directory according to the settings you stored in the task file, `/usr/local/config1`, type the command:

```
uvscan --load /usr/local/config1 /usr/docs
```

The contents of the task file `/usr/local/config1`, are:

```
-m /viruses --ignore-compressed --maxfilesize 4
```

They instruct the scan to move any infected files to `/viruses`, to ignore any compressed files in the target directory, and to examine only files smaller than 4MB.

As an alternative, you can arrange the contents of the task file as separate lines:

```
-m /usr/local/viruses
--ignore-compressed
--maxfilesize 4
```

Example 2

To scan only files smaller than 4MB and to ignore any compressed files in three separate directories, type the command:

```
uvscan --load /usr/local/config1 --file mylist
```

The contents of the task file `/usr/local/config1`, are:

```
--ignore-compressed
--maxfilesize 4
```

The contents of the other file, `mylist`, are:

```
/usr/local/bin
/tmp
/etc
```

Scheduling scans

You can use the UNIX `cron` scheduler to run automated scans. `Cron` stores the scheduling commands in its `crontab` files. For further information about `cron` and `crontab`, refer to your UNIX documentation or view the Help text, using the commands, `man cron` or `man crontab`.

Examples

To schedule a scan to run at 18:30 (6:30 p.m.) every weekday, add the following to your `crontab` file:

```
30 18 * * 1-5 /usr/local/bin/uvscan
```

To schedule a scan to run and produce a summary at 11:50 p.m. every Sunday, add the following to your `crontab` file:

```
50 23 * * 0 /usr/local/bin/uvscan --summary
```

To schedule a scan to run on the `work` directory at 10:15 a.m. every Saturday in accordance with options specified in a configuration file `conf1`, add the following to your `crontab` file:

```
15 10 * * 6 /usr/local/bin/uvscan --load conf1 /work
```

To schedule a scan to run at 8:45 a.m. every Monday on the files specified in the file `mylist`, add the following to your `crontab` file:

```
45 8 * * 1 /usr/local/bin/uvscan --f /usr/local/mylist
```

Handling viruses

If the scanner discovers a virus while scanning, it returns exit code number 13. See *Exit codes* on page 31 for a full description of each code.

To clean infected files or directories, or move them to a quarantine location on your network, you can configure your scanner using one or more response options, which are described in *Response options* on page 26,

The following examples show how you can use these options to respond to a virus attack. The examples assume that the scanner is available in your search path.

Example 1

To scan and clean all files in the `/usr/docs` directory and all of its subdirectories, type the command:

```
uvscan -cr /usr/docs
```

Example 2

To scan and clean all files in the `/usr/docs` directory and its subdirectories, but ignore any other file systems that are mounted, type the command:

```
uvscan -cr --one-file-system /usr/docs
```

Example 3

To scan all files except compressed files in the `/usr/docs` directory and its subdirectories, and to move any infected files to `/viruses`, type the command:

```
uvscan -m /viruses -r --ignore-compressed /usr/docs
```

Example 4

To scan a file with a name prefixed with "`-`", type the command:

```
uvscan -c -v - -myfile
```

The program scans the named file. It cleans any detected viruses and issues a progress message. This format avoids confusion between the names of the options and the name of the target. Without the "`-`" option, the `uvscan` command appears to have three options and no target:

```
uvscan -c -v -myfile
```

Using heuristic analysis

A scanner uses two techniques to detect viruses: signature matching and heuristic analysis.

A *virus signature* is simply a binary pattern that is found in a virus-infected file. Using information in the DAT files, the scanner searches for those patterns. However, this approach cannot detect a new virus because its signature is not yet known, therefore the scanner uses another technique — *heuristic analysis*.

Programs, documents or e-mail messages that carry a virus often have distinctive features. They might attempt unprompted modification of files, invoke mail clients, or use other means to replicate themselves. The scanner analyzes the program code to detect these kinds of computer instructions. The scanner also searches for legitimate non-virus-like behavior, such as prompting the user before taking action, and thereby avoids raising false alarms.

In an attempt to avoid being detected, some viruses are encrypted. Each computer instruction is simply a binary number, but the computer does not use all the possible numbers. By searching for unexpected numbers inside a program file, the scanner can detect an encrypted virus. By using these two techniques, the scanner can detect both known viruses and many new viruses and variants. Options that use heuristic analysis include `---analyze`, `--manalyze`, `--panalyze`. See *Scanning options* on page 23.

Handling an infected file that cannot be cleaned

If the scanner cannot clean an infected file, it renames the file to prevent its use. When a file is renamed, only the file extension (typically three letters) is changed. The following table shows the method of renaming.

Table 3-1 Renaming infected files

Original	Renamed	Description
`Not v??`	`v??`	File extensions that do not start with *v* are renamed with `v` as the initial letter of the file extension. For example, myfile.doc becomes myfile.voc.
`v??`	`vir`	File extensions that start with v are renamed as .vir. For example, myfile.vbs becomes myfile.vir.
`vir, v01-v99`		These files are recognized as already infected, and are not renamed again.
<blank>	`vir`	Files with no extensions are given the extension, .vir.

For example, if an infected file called `bad.com` is found, the scanner attempts to rename the file to bad.vom. However, if a file of that name already exists in the directory, the scanner attempts to rename the file to `bad.vir`, `bad.v01`, `bad.v02`, and so on.

For file extensions with more than three letters, the name is usually not truncated. For example, `notepad.class` becomes `notepad.vlass`. However, an infected file called `water.vapor` becomes `water.vir`.

Producing reports

The program might take some time to complete a scan, particularly over many directories and files. However, the scanner can keep you informed of its progress, any viruses it finds, and its response to them.

The program displays this information on your screen if you add the `--summary` or `--verbose` option to the command line. To learn more about each option, see *Response options* on page 26.

The `--verbose` option tells you which files the program is examining.

When the scan finishes, the `--summary` option identifies the following:

- How many files were scanned.
- How many files were cleaned.
- How many files were not scanned.
- How many infected files were found.

Example

In the report below, both the `--summary` and `--verbose` options were used for scanning files in the `/usr/data` directory.

```
$ uvscan --summary  --verbose /usr/data

Scanning /usr/data/*

Scanning file /usr/data/command.com

Scanning file /usr/data/grep.com

Summary report on /usr/data/*

File(s)

    Total files: ..........       2

    Clean: ................       2

    Not scanned: ..........       0

    Possibly Infected: .....      0
```

To determine the time taken for the scan, you may use the UNIX `time` command.

XML reports

You can generate an XML format report using the XMLPATH switch. For example, run the following command from the install directory:

```
scan . /XMLPATH=report.xml /RPTALL
```

This will generate a file called **report.xml** with the following content.

```
<?xml version="1.0" encoding="utf-8"?>

<!-- Scan Results -->
```

```
<Scan>
<Preamble>
<Product_name value="McAfee VirusScan Command Line for Win32" />
<Version value="6.0.0.162" />
<AV_Engine_version value="5301.4018" />
<Dat_set_version value="5594" />
</Preamble>
<Date_Time value="2009-Jun-19 14:26:18" />
<Options value=". /xmlpath=report.xml /rptall " />
<File name="D:\vcl\avvclean.dat" status="ok" />
<File name="D:\vcl\avvnames.dat" status="ok" />
<File name="D:\vcl\avvscan.dat" status="ok" />
<File name="D:\vcl\config.dat" status="ok" />
<File name="D:\vcl\mc5300up.001" status="ok" />
<File name="D:\vcl\mcscan32.dll" status="ok" />
<File name="D:\vcl\report.xml" status="ok" />
<File name="D:\vcl\runtime.dat" status="ok" />
<File name="D:\vcl\scan.exe" status="ok" />
<File name="D:\vcl\vcl6wpg.pdf" status="ok" />
<summary On-Path="D:\vcl" Total-files="14" Clean="10" Not-Scanned="4"
Possibly-Infected="0" />
<Time value="00:00.01" />
</Scan>
```

See *Schema for the XML reports* on page 38 for information on the formal schema for XML reports.

Choosing the options

The following sections describe the options you can use to target your scan:

- *Scanning options*.
- *Response options* on page 26.
- *General options* on page 28.
- *Options in alphabetic order* on page 29.

The descriptions use the following conventions to identify the options or required parameters:

- Short versions of each command option appear after a single dash (-).
- Long versions of each command option, if any, appear after two dashes (--).
- Variables, such as file names or paths, appear in italics within brackets (< >).

To learn how to add these options to the command line, see *Command-line conventions* on page 15.

Scanning options

Scanning options describe how and where each scan looks for infected files. You can use a combination of these options to customize the scan to suit your needs.

Table 3-2 Scanning options

Option	Description
`--afc <size>`	Specify the file cache size. By default, the cache size is 12MB. A larger cache size can improve scanning performance in some cases, unless the computer has low memory. The range of sizes allowed is 8MB to 512MB. Specify the size in megabytes. For example, `--afc 64` specifies 64MB of cache.
`--allole`	Check every file for OLE objects.
`--analyze` `--analyse`	Use heuristic analysis to find possible new viruses in *clean* files. This step occurs after the program has checked the file for other viruses. See also *Using heuristic analysis* on page 19. For macro viruses only, use `--manalyze`. For program viruses only, use `--panalyze`.

`--atime-preserve`
`-p`
`--plad`
	Preserve the last-accessed time and date for files that are scanned.
	Some backup software archives only changed files, and determines this information from each file's last-accessed date (or 'a-time'). Normally, scanning changes that date. This option will preserve the date, enabling the backup software to work as intended. Sometimes when this option is used, the file date is not preserved; if a file contains a virus, or the scan was started by a user who does not own the file, the file date is changed.

`--config <file>`	Run the options specified in <*file*>. You cannot nest configuration files within other configuration files. See also *Configuring scans* on page 16 and *Scheduling scans* on page 17.

`-d <directory>`
`--dat <directory>`
`--data-directory <directory>`
	Specify the location of the DAT files — `avvscan.dat`, `avvnames.dat`, and `avvclean.dat`.
	If you do not use this option in the command line, the program looks in the same directory from where it was executed.
	If it cannot find these data files, the program issues exit code 6.

`--decompress`	Decompress DAT files after an update.
`--exclude <file>`	Exclude the directories or files from the scan as specified in <*file*>. List the complete path to each directory or file on its own line. You may use wildcards, * and ?.
`-e` `--exit-on-error`	Quit and display an error message if an error occurs. The error message indicates the severity of the error. See page 31 for an explanation of exit codes.

Table 3-2 Scanning options (continued)

Option	Description
`--extensions <EXT1[,EXT2,...]>`	Examine files that have the specified extensions. You can specify as many extensions as you want. Separate each with a comma, but without a space. If you choose this option, the program scans only susceptible files, files with execute permissions, and those you specify here. To see the list of susceptible files, use the `--extlist` option on page 28.
`--extra <file>`	Specify the full path and file name of any `extra.dat` file. If you do not specify this option in the command line, the program looks in the same directory from where it was executed. If it cannot find this file, the program issues exit code 6.
`--fam`	Find all macros, not just macros suspected of being infected. The scanner treats any macro as a possible virus and reports that the file contains macros. However, the macros are not removed. If you suspect that you have an infection in a file, you can remove all macros from the file using the `--fam` and `--cleandocall` or `--dam` options (on page 26) together, although you should only do this with caution.
`-f <file>` `--file <file>`	Scan the directories or files as specified in <*file*>.
`--floppya` `--floppyb`	Scan the boot sector of the disk in drive A or B. This option is for Intel-based UNIX systems only, namely FreeBSD and Linux.
`--ignore-compressed` `--nocomp`	Ignore compressed files. By default, the program scans files saved in these compression formats: ICE, LZEXE, PKLITE, Cryptcom, COM2EXE, Diet, Teledisk, Microsoft Expand and GZIP. Although this option reduces the scanning time, it increases the threat because these file types are not scanned.
`--ignore-links`	Do not resolve any symbolic links and do not scan the link targets. Normally, the program *follows* each symbolic link and scans the linked file.
`--load <file>`	See `--config` option.
`--mailbox`	Scan plain-text mailboxes. These include Eudora, PINE, and Netscape. Most mailboxes will be in MIME format, and therefore the `--mime` option is also required. This option does not clean or rename infected mail items; you must first extract them from the mailbox.

`--manalyze`
`--manalyse`
`--macro-heuristics`
	Use heuristic analysis to identify potential macro viruses.
	(In Microsoft Word, you can automate a task by using a *macro* - a group of Word commands that run as a single command.)
	This is a subset of `--analyze`. See also *Using heuristic analysis* on page 19.

`--maxfilesize <size>`	Examine only those files smaller than the specified size. Specify the file size in megabytes. For example, `maxfilesize 5` means scan only files that are smaller than 5MB.

Table 3-2 Scanning options (continued)

Option	Description
`--mime`	Scan MIME-encoded files. This type of file is not scanned by default.
`--noboot`	Do not scan the boot sector.
`--nodecrypt`	Do not decrypt Microsoft Office compound documents that are password-protected. By default, macros inside password-protected compound documents are scanned by employing *password cracking* techniques. If, for reasons of security, you do not require these techniques, use this option. Password cracking does not render the file readable.
`--nocomp`	`See ignore-compressed`.
`--nodoc`	Do not scan Microsoft Office document files. This includes Microsoft Office documents, OLE2, CorelDraw, PowerPoint, WordPerfect, RTF, Visio, Adobe PDF 5, Autodesk Autocad 2000, and Corel PhotoPaint 9 files.
`--noexpire`	Do not issue a warning if the DAT files are out of date.
`--nojokes`	Do not report any joke programs.
`--noscript`	Do not scan files that contain HTML, JavaScript, Visual Basic, or Script Component Type Libraries. This type of file is normally scanned by default. Stand-alone Javascript and Visual Basic Script files will still be scanned.
`--one-file-system`	Scan an entire directory tree without scanning mounted file systems, if you use this option in conjunction with the `--sub` option. Normally, the program treats a mount point as a subdirectory and scans that file system. This option prevents the scan from running in subdirectories that are on a different file system to the original directory.
`--panalyze` `--panalyse`	Use heuristic analysis to identify potential program viruses. By default, the program scans only for known viruses. The `--panalyze` option is a subset of `--analyze`. See also *Using heuristic analysis* on page 19.
`--program`	Scan for potentially unwanted applications. Some widely available applications, such as “password crackers” can be used maliciously or can pose a security threat.

`-r`
`--recursive`
`--sub`
	Examine any subdirectories in addition to the specified target directory.
	By default, the scanner examines only the files within the specified directory.

`--secure`	Examine all files, unzip archive files and use heuristic analysis. This option activates the `--analyze` and `--unzip` options. If the `--selected` and `--extensions` options are in the command line, they are ignored.
`--showcomp`	Report any files that are packaged.

Table 3-2 Scanning options (continued)

Option	Description
`-s` `--selected`	Look for viruses in any file that has execute permissions, and in all files that are susceptible to virus infection. By default, all files are scanned. By scanning only files that are susceptible to virus infection, the program can scan a directory faster. To see the list of susceptible files, use the `--extlist` option (page 28).
`--sub`	See `-r`.
`--threads <nThreads>`	Scan multithreaded with specified number of threads.
`--timeout <seconds>`	Set the maximum time to scan any one file.
`--unzip`	Scan inside archive files, such as those saved in ZIP, LHA, PKarc, ARJ, TAR, CHM, and RAR formats. If used with `--clean`, this option attempts to clean non-compressed files inside .ZIP files only. No other archive formats can be cleaned. The program cannot clean infected files found within any other archive format; you must first extract them manually from the archive file.
`--xmlpath <filename>`	Create XML report

Response options

Response options determine how your scanner responds to an infection. You can use a combination of these options to customize the scan. None of the options in Table 3-3 occur automatically. To activate each option, specify it in the command line.

Table 3-3 Response options

Option	Description
`-c` `--clean`	Automatically remove any viruses from infected files. By default, the program states only that infections were found but does not try to clean the infected file. If the program cannot clean the file, it displays a warning message. If you use this option, repeat the scan to ensure that there are no more infections.
`--cleandocall`	Delete all macros in a file if an infected macro is found.
`--dam`	If you suspect that a file is infected, you can choose to remove all macros from the file to prevent any exposure to a virus. To pre-emptively delete all macros in a file, use this option with `--fam` (on page 24), although you should do this with caution. If you use these two options together, all found macros are deleted, regardless of the presence of an infection.
`--delete`	Automatically delete any infected files that are found.
`-m <directory>` `--move <directory>`	Move any infected files to a quarantine location as specified. When the program moves an infected file, it replicates the full directory path of the infected file inside the quarantine directory so you can determine the original location of the infected file. If you use this option with `--clean`, the program copies the infected files to a quarantine location and tries to clean the original. If the program cannot clean the original, it deletes the file.
`--norename`	Do not rename an infected file that cannot be cleaned. See *Handling an infected file that cannot be cleaned* on page 19 for information about renaming.

General options

General options provide help or give extra information about the scan. You may use a combination of these options to customize the scan. None of the options in Table 3-4 occur automatically. To activate each option, specify it as part of the command line.

Table 3-4 General options

Option	Description
`--extlist`	Display a list of all file extensions that are susceptible to infection. In other words, those file extensions that are scanned when `--selected` is set.
`-h` `--help`	List the most commonly used options, with a short description. For a full description, use `man uvscan`.
`--summary`	Produce a summary of the scan. This includes the following: ■ How many files were examined. ■ How many infected files were found. ■ How many viruses were removed from infected files.
`-v` `--verbose`	Display a progress summary during the scan.See also *Producing reports* on page 20.
`--version`	Display the scanner's version number.
`--virus-list`	Display the name of each virus that the scanner can detect. This option produces a long list, which is best viewed from a text file. To do this, redirect the output to a file for viewing. For full details about each virus, see the Virus Information Library under *Contact information* on page 8.

Options in alphabetic order

For convenience, the options are repeated in this section in alphabetic order. For fuller descriptions, see the previous sections.

Table 3-5 Options in alphabetic order

Option	Description	See ...
`--afc <size>`	Specify the file cache size.	page 23
`--allole`	Check every file for OLE objects.	page 23
`--analyse`	Same as `--analyze`.	page 23
`--analyze`	Use heuristic analysis to find possible new viruses in clean files.	page 23
`--atime-preserve`	Preserve the last-accessed time and date for files that are scanned.	page 23
`-c`	Same as `--clean`.	page 26
`--clean`	Automatically remove any viruses from infected files.	page 26
`--cleandocall`	Same as `--dam`.	page 26
`--config <file>`	Run the options specified in <file>.	page 23
`-d <directory>`	Same as `--dat <directory>`.	page 23
`--dam`	Delete all macros in a file if an infected macro is found.	page 26
`--dat <directory>`	Specify the location of the DAT files — avvscan.dat, avvnames.dat, and avvclean.dat.	page 23
`--data-directory <directory>`	Same as `--dat <directory>`.	page 23
`--decompress`	Decompress DAT files after an update.	page 23
`--delete`	Automatically delete any infected files that are found.	page 26
`-e`	Same as `--exit-on-error`.	page 23
`--exclude <file>`	Exclude the directories or files from the scan as specified in <file>.	page 23
`--exit-on-error`	Quit and display an error message if an error occurs.	page 23
`--extensions <EXT1[,EXT2,...]>`	Examine files that have the specified extensions.	page 24
`--extlist`	Display a list of all file extensions that are susceptible to infection.	page 28
`--extra <file>`	Specify the full path and file name of any extra.dat file.	page 24
`-f <file>`	Same as `--file <file>`.	page 24
`--fam`	Find all macros, not just macros suspected of being infected.	page 24
`--file <file>`	Scan the directories or files as specified in <file>.	page 24
`--floppya`	Scan the boot sector of the disk in drive A or B.	page 24
`--floppyb`		page 24
`-h`	Same as `--help`.	page 28
`--help`	List the most commonly used options, with a short description.	page 28
`--ignore-compressed`	Ignore compressed files.	page 24

Table 3-5 Options in alphabetic order (continued)

Option	Description	See ...
`--ignore-links`	Do not resolve any symbolic links and do not scan the link targets.	page 24
`--load <file>`	Same as `--config <file>`.	page 23
`-m <directory>`	Same as `--move <directory>`.	page 26
`--macro-heuristics`	Same as `--manalyze`.	page 24
`--mailbox`	Scan plain-text mailboxes.	page 24
`--manalyse`	Same as `--manalyze`.	page 24
`--manalyze`	Use heuristic analysis to identify potential macro viruses.	page 24
`--maxfilesize <size>`	Examine only those files smaller than the specified size.	page 24
`--memsize`	Set maximum size of file that will be cached in memory for scanning, in Kb.	page 30
`--mime`	Scan MIME-encoded files.	page 25
`--move <directory>`	Move any infected files to a quarantine location as specified.	page 26
`--noboot`	Do not scan the boot sector.	page 25
`--nocomp`	Same as `--ignore-compressed`.	page 24
`--nodecrypt`	Do not decrypt Microsoft Office compound documents that are password-protected.	page 25
`--nodoc`	Do not scan Microsoft Office document files.	page 25
`--noexpire`	Do not issue a warning if the DAT files are out of date.	page 25
`--nojokes`	Do not report any joke programs.	page 25
`--norename`	Do not rename an infected file that cannot be cleaned.	page 26
`--noscript`	Do not scan files that contain HTML, JavaScript, Visual Basic, or Script Component Type Libraries.	page 25
`--one-file-system`	Scan an entire directory tree without scanning mounted file systems, if you use this option in conjunction with the –sub option.	page 25
`-p`	Same as `--atime-preserve`.	page 23
`--panalyse`	Same as `--panalyze`.	page 25
`--panalyze`	Use heuristic analysis to identify potential program viruses.	page 25
`--plad`	Same as `--atime-preserve`.	page 23
`--program`	Scan for potentially unwanted applications.	page 25
`-r`	Same as `--sub`.	page 25
`--recursive`	Same as `--sub`.	page 25
`-s`	Same as`--selected`.	page 26
`--secure`	Examine all files, unzip archive files and use heuristic analysis.	page 25
`--selected`	Look for viruses in any file that has execute permissions, and in all files that are susceptible to virus infection.	page 26
`--showcomp`	Report any files that are packaged.	page 25
`--sub`	Examine any subdirectories in addition to the specified target directory.	page 25

Table 3-5 Options in alphabetic order (continued)

Option	Description	See ...
`--summary`	Produce a summary of the scan.	page 28
`--threads <nThreads>`	Scan multithreaded with specified number of threads.	page 26
`--timeout <seconds>`	Set the maximum time to scan any one file.	page 26
`--unzip`	Scan inside archive files, such as those saved in ZIP, LHA, PKarc, ARJ, TAR, CHM, and RAR formats.	page 26
`-v`	Same as `--verbose`.	page 28
`--verbose`	Display a progress summary during the scan.See also Producing reports on page 20.	page 28
`--version`	Display the scanner's version number.	page 28
`--virus-list`	Display the name of each virus that the scanner can detect.	page 28
`--xmlpath <filename>`	Create XML report.	page 31

Exit codes

When it exits, VirusScan® for UNIX returns a code to identify any viruses or problems that were found during a scan.

Table 3-6 Exit codes

Code	Description
0	The scanner found no viruses or other potentially unwanted software and returned no errors.
2	Integrity check on a DAT file failed.
6	A general problem occurred.
8	The scanner could not find a DAT file.
12	The scanner tried to clean a file, and that attempt failed for some reason, and the file is still infected.
13	The scanner found one or more viruses or hostile objects — such as a Trojan-horse program, joke program, or test file.
15	The scanner's self-check failed; it may be infected or damaged.
19	The scanner succeeded in cleaning all infected files.

4 Preventing Infections

VirusScan® for UNIX is an effective tool for preventing infections, and it is most effective when combined with regular backups, meaningful password protection, user training, and awareness of threats from viruses and other potentially unwanted software.

To create a secure system environment and minimize the chance of infection, we recommend that you do the following:

- Install VirusScan® for UNIX software and other McAfee anti-virus software where applicable.
- Include a `uvscan` command in a `crontab` file.
- Make frequent backups of important files. Even if you have VirusScan® for UNIX software to prevent infections, damage from fire, theft, or vandalism can render your data unrecoverable without a recent backup.

Detecting new and unidentified viruses

To offer the best virus protection possible, we continually update the definition (DAT) files that the VirusScan® for UNIX software uses to detect viruses and other potentially unwanted software. For maximum protection, you should regularly retrieve these files.

We offer free online DAT file updates for the life of your product, but cannot guarantee they will be compatible with previous versions. By updating your software to the latest version of the product and updating regularly to the latest DAT files, you ensure complete virus protection for the term of your software subscription or maintenance plan.

Why do I need new DAT files?

Hundreds of new viruses appear each month. Often, older DAT files cannot detect these new variations. For example, the DAT files with your original copy of VirusScan® for UNIX might not detect a virus that was discovered after you bought the product.

If you suspect you have found a new virus, use AVERT WebImmune. See *Contact information* on page 8 for the address.

Updating your DAT files

The DAT files are contained in a single compressed file. Download the new file from either of the following sources:

- **FTP server**: Open a connection to the FTP site, ftp://ftp.mcafee.com.

  Use anonymous as your user name and your email address as your password to gain access. Look for a compressed file in the directory commonupdater. The file has the format avvdat-nnnn.zip, where nnnn is the DAT version number. For example: avvdat-5650.zip.

- **Website**: Start your browser, go to the Downloads area for the latest file. See the Download site in *Contact information* on page 9 for the address.

To use the new DAT files:

1. Create a download directory.
2. Change to the download directory, and download the new compressed file from the source you have chosen.
3. To extract the DAT files, type the command:

   ```
   unzip file
   ```

   Here, *file* is the name of the zip file you downloaded.

4. Type this command to move the DAT files to the directory where your software is installed. Name the file using lower case.

   ```
   mv *.dat installation-directory
   ```

   Here, *installation-directory* is the directory where you installed the software. (See *Installing the software* on page 10.)

   Your computer overwrites the old DAT files with the new files. Your anti-virus software will now use the new DAT files to scan for viruses.

After an update, run the following command once to decompress the newly downloaded DATs and accelerate the time for subsequent initializations.

```
SCAN /DECOMPRESS
```

This product is not suitable for on-access (single file) scanning.

Sample update script for UNIX

The following script retrieves the most recent DAT files from the McAfee website. It is provided as an example, for you to use and modify for your environment. It has not been thoroughly tested.

This script should be excuted from the directory where uvscan has been installed. The `EMAIL_ADDRESS` variable needs to be set to a valid email address.

```
#!/bin/sh
# (c) 2008 McAfee, Inc. All Rights Reserved.
# required programs: unzip, ftp, awk, echo, cut, ls, printf

### defaults: do not modify
unset md5checker leave_files debug

#============================================================
### change these variables to match your information
# Set the following to your own e-mail address
EMAIL_ADDRESS=""

### change these variables to match your environment
# install_dir must be a directory and writable
install_dir=`dirname "$0"`
# tmp_dir must be a directory and writable
tmp_dir="/tmp/dat-update"
# optional: this prg is responsible for calculating the md5 for a file
md5checker="md5sum"

### set your preferences
# set to non-empty to leave downloaded files after the update is done
#leave_files="true"
# show debug messages (set to non-empty to enable)
debug=yes

# these variables are normally best left unmodified
UVSCAN_EXE="uvscan"
UVSCAN_SWITCHES=""
#============================================================

Cleanup()
{
    if [ -z "$leave_files" ] ; then
        for f in "$update_ini" "$download" ; do
            [ -n "$f" -a -e "$f" ] && rm -f "$f"
        done
    fi
}

exit_error()
{
    [ -n "$1" ] && printf "$prgname: ERROR: $1\n"
    Cleanup ; exit 1
}

print_debug()
{
    [ -n "$debug" ] && printf "$prgname: [debug] $@\n"
}

# Function to parse update.ini and return, via stdout, the
# contents of a specified section. Requires the update.ini
# file to be available on stdin.
# $1 - Section name
FindINISection()
{
    unset section_found

    section_name="[$1]"
    while read line ; do
        if [ "$line" = "$section_name" ] ; then
            section_found="true"
        elif [ -n "$section_found" ] ; then
            if [ "`echo $line | cut -c1`" != "[" ] ; then
                [ -n "$line" ] && printf "$line\n"
            else
                unset section_found
```

```
            fi
        fi
    done
}

# Function to return the DAT version currently installed for
# use with the command line scanner
# $1 - uvscan exe (including path)
# $2 - any extra switches for uvscan
GetCurrentDATVersion()
{
    dirname=`dirname "$1"`
    uvscan_bin=`basename "$1"`

    output=`(cd "$dirname"; "./$uvscan_bin" $2 --version )`
    [ $? -eq 0 ] || return 1

    lversion=`printf "$output\n" | grep "Virus data file v" |
        cut -d' ' -f4 | cut -c2-`
    printf "${lversion}.0\n"

    return 0
}

# Function to download a specified file from ftp.nai.com
# $1 - Path on ftp server
# $2 - name of file to download.
# $3 - download type (either bin or ascii)
# $4 - download directory
DownloadFile()
{
    [ "$3" = "bin" -o "$3" = "ascii" ] || return 1
    dtype="$3"

    # An e-mail address must be set in this environment variable
    [ -n "$EMAIL_ADDRESS" ] || return 1

    print_debug "downloading file '$2' into '$4'"
    echo "
open ftp.nai.com
user anonymous $EMAIL_ADDRESS
cd $1
lcd $4
$dtype
get $2
" | ftp -i -n || return 1

    return 0
}

# Function to check the specified file against its expected size, checksum and MD5
checksum.
# $1 - File name (including path)
# $2 - expected size
# $3 - MD5 Checksum
ValidateFile()
{
    # Check the file size matches what we expect...
    size=`ls -l "$1" | awk ' { print $5 } '`
    [ -n "$size" -a "$size" = "$2" ] || return 1

    # make md5 check optional. return "success" if there's no support
    [ -z "$md5checker" -o "(" ! -x "`which $md5checker 2> /dev/null`" \
        ")" ] && return 0
    # Check the md5 checksum...
    md5_csum=`$md5checker "$1" 2>/dev/null | cut -d' ' -f1`
    [ -n "$md5_csum" -a "$md5_csum" = "$3" ] # return code
}

# Function to extract the listed files from the given zip file.
# $1 - directory to install to
# $2 - downloaded file.
# $3 - list of files to install
Update_ZIP()
{
    unset flist
    for file in $3 ; do
        fname=`printf "$file\n" | awk -F':' ' { print $1 } '`
        flist="$flist $fname"
    done
```

```
    # Backup any files about to be updated...
    [ ! -d "backup" ] && mkdir backup 2>/dev/null
    [ -d "backup" ] && cp $flist "backup" 2>/dev/null

    # Update the DAT files.
    print_debug "uncompressing '$2'..."
    unzip -o -d $1 $2 $flist >/dev/null || return 1
    for file in $3 ; do
        fname=`printf "$file\n" | awk -F':' ' { print $1 } '`
        permissions=`printf "$file\n" | awk -F':' ' { print $NF } '`
        chmod "$permissions" "$1/$fname"
    done
    return 0
}

# globals
prgname=`basename "$0"`
unset perform_update update_ini download

# sanity checks
[ -d "$tmp_dir" ] || mkdir -p "$tmp_dir" 2>/dev/null
[ -d "$tmp_dir" ] || exit_error "directory '$tmp_dir' does not exist."
[ -x "$install_dir/$UVSCAN_EXE" ] \
    || exit_error "could not find uvscan executable"

DownloadFile "pub/datfiles/english" "update.ini" "ascii" "$tmp_dir" \
    || exit_error "downloading update.ini"
update_ini="$tmp_dir/update.ini"
# Did we get update.ini?
[ -r "$update_ini" ] || exit_error "unable to get update.ini file"

ini_section=ZIP
file_list="avvscan.dat:444 avvnames.dat:444 avvclean.dat:444"

# Get the version of the installed DATs...
current_version=`GetCurrentDATVersion "$install_dir/$UVSCAN_EXE"
"$UVSCAN_SWITCHES"`
[ -n "$current_version" ] \
    || exit_error "unable to get currently installed DAT version."
current_major=`echo "$current_version" | cut -d. -f-1`
current_minor=`echo "$current_version" | cut -d. -f2-`

INISection=`FindINISection "$ini_section" < $update_ini`
[ -n "$INISection" ] \
    || exit_error "unable to get section $ini_section from update.ini"

unset major_ver file_name file_path file_size md5
# Some INI sections have the MinorVersion field missing.
minor_ver=0 # To work around this, we will initialise it to 0.

# Parse the section and keep what we are interested in.
for field in $INISection ; do
    name=`echo "$field" | awk -F'=' ' { print $1 } '`
    value=`echo "$field" | awk -F'=' ' { print $2 } '`

    case $name in
        "DATVersion")   major_ver=$value    ;;  # available: major
        "MinorVersion") minor_ver=$value    ;;  # available: minor
        "FileName")     file_name="$value"  ;;  # file to download
        "FilePath")     file_path=$value    ;;  # path on FTP server
        "FileSize")     file_size=$value    ;;  # file size
        "MD5")          md5=$value          ;;  # MD5 checksum
    esac
done
# sanity check
[ -n "$major_ver" -a -n "$minor_ver" -a -n "$file_name" \
    -a -n "$file_path" -a -n "$file_size" -a -n "$md5" ] \
    || exit_error "update.ini: '[$ini_section]' has incomplete data"

[ "(" "$current_major" -lt "$major_ver" ")" -o "(" \
    "$current_major" -eq "$major_ver" -a \
    "$current_minor" -lt "$minor_ver" ")" ] && perform_update="yes"

if [ -n "$perform_update" ] ; then
    printf "$prgname: Performing an update ($current_version ->
$major_ver.$minor_ver)\n"

    # Download the dat files...
    DownloadFile "$file_path" "$file_name" "bin" "$tmp_dir" \
        || exit_error "downloading '$file_name'"
    download="$tmp_dir/$file_name"
```

```
    # Did we get the dat update file?
    [ -r "$download" ] || exit_error "unable to get $file_name file"

    ValidateFile "$download" "$file_size" "$md5" \
        || exit_error "DAT update failed - File validation failed"
    Update_ZIP "$install_dir" "$download" "$file_list" \
        || exit_error "updating DATs from file '$download'"

    # Check the new version matches the downloaded one.
    new_version=`GetCurrentDATVersion "$install_dir/$UVSCAN_EXE"
"$UVSCAN_SWITCHES"`

    new_major=`echo "$new_version" | cut -d. -f-1`
    new_minor=`echo "$new_version" | cut -d. -f2-`

    if [ "$new_major" = "$major_ver" -a "$new_minor" = "$minor_ver" ]
   then printf "$prgname: DAT update succeeded $current_version -> $new_version\n"
  else exit_error "DAT update failed - installed version different than expected\n"
    fi
else
    printf "$prgname: DAT already up to date ($current_version)\n"
fi

Cleanup ; exit 0
```

A Schema for the XML reports

The formal schema for the XML reports is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema for the VSCL 6.0 XML Report format-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Scan">
<xs:complexType>
<xs:sequence>
<xs:element ref="Preamble"/>
<xs:element ref="Date_Time"/>
<xs:element ref="Options"/>
<xs:group ref="FileSummary" maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref="Time"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Preamble">
<xs:complexType>
<xs:sequence>
<xs:element ref="Product_name"/>
<xs:element ref="Version"/>
<xs:element ref="License_info"/>
<xs:element ref="AV_Engine_version"/>
<xs:element ref="Dat_set_version"/>
<xs:element ref="Extra_Dat_Info" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Date_Time">
<xs:complexType>
```

```
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Options">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Time">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:group name="FileSummary">
<xs:sequence>
<xs:element ref="File" maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref="Summary" maxOccurs="unbounded"/>
</xs:sequence>
</xs:group>
<xs:element name="File">
<xs:complexType>
<xs:attribute name="status" type="xs:string" use="required"/>
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="virus-name" type="xs:string" use="optional"/>
<xs:attribute name="detection-type" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Summary">
<xs:complexType>
<xs:attribute name="Total-processes" type="xs:int" use="optional"/>
<xs:attribute name="On-Path" type="xs:string" use="optional"/>
<xs:attribute name="Total-files" type="xs:int" use="optional"/>
<xs:attribute name="Total-Objects" type="xs:int" use="optional"/>
<xs:attribute name="Possibly-Infected" type="xs:int" use="optional"/>
<xs:attribute name="Objects-Possibly-Infected" type="xs:int"
use="optional"/>
<xs:attribute name="Not-Scanned" type="xs:int" use="optional"/>
<xs:attribute name="Clean" type="xs:int" use="optional"/>
<xs:attribute name="Possibly-Infected-MBR" type="xs:int" use="optional"/>
```

```
<xs:attribute name="Possibly-Infected-BootSector" type="xs:int"
use="optional"/>

<xs:attribute name="Master-Boot-Records" type="xs:int" use="optional"/>

<xs:attribute name="Boot-Sectors" type="xs:int" use="optional"/>

<xs:attribute name="Cleaned" type="xs:int" use="optional"/>

<xs:attribute name="Moved" type="xs:int" use="optional"/>

<xs:attribute name="Deleted" type="xs:int" use="optional"/>

</xs:complexType>

</xs:element>

<xs:element name="Product_name">

<xs:complexType>

<xs:attribute name="value" type="xs:string" use="required"/>

</xs:complexType>

</xs:element>

<xs:element name="Version">

<xs:complexType>

<xs:attribute name="value" type="xs:string" use="required"/>

</xs:complexType>

</xs:element>

<xs:element name="License_info">

<xs:complexType>

<xs:attribute name="value" type="xs:string" use="required"/>

</xs:complexType>

</xs:element>

<xs:element name="AV_Engine_version">

<xs:complexType>

<xs:attribute name="value" type="xs:decimal" use="required"/>

</xs:complexType>

</xs:element>

<xs:element name="Dat_set_version">

<xs:complexType>

<xs:attribute name="value" type="xs:short" use="required"/>

</xs:complexType>

</xs:element>

<xs:element name="Extra_Dat_Info">

<xs:complexType>

<xs:attribute name="Path" type="xs:string" use="required"/>

<xs:attribute name="Additional_Viruses" type="xs:string" use="required"/>

</xs:complexType>
```

```
</xs:element>
</xs:schema>
```

Index