

ALWIL Software

iAus

administrators documentation

Contents

1	What is iAVS?	7
2	How does it work?	9
3	Security?	11
4	Mirroring	13
4.1	How to mirror - general description	13
4.2	Repeated tasks - general description	13
4.3	How to mirror - the solution	14
4.4	Periodic maintainance - completed solution	14
4.5	iAVS Client setup	15
5	How is server chosen?	17
6	File servers.def	19
7	Groups.def file	21
8	Index.*.def files	23
9	Packages.def file	25
10	Files *.stamp	27
11	File mirror.def	29
12	File lock.def.stamp	31
13	Syncer.ini file	33
14	Files with distribution description	35
15	At command	37
16	Program tmcr.exe	39
17	Mirror program.pl	41
18	Program cldistro.pl	43
19	Mkdistro.pl program	45
20	Chdistro.pl program	47
21	Uploader.pl program	49
22	Program doall.pl	51
23	Program md5.exe	53

24	Changes	55
25	Frequently asked questions	57
26	Frequently Asked Questions - Perl	59
27	Frequently Asked Questions - Unix	61
28	Acknowledgements	63

Picture index

1 What is iAVS?

iAVS is system providing differential updates of software. Now it's used for differential updates of VPS files only.

The updates process was changed because the speed of replication of new 'internet enabled' viruses was much higher than speed of releasing of new antivirus updates. Existing system was unsuitable because of its low flexibility.

Existing system of updates is still usable, see documentation of Avast32 and document AUPD.TXT with differences.

This manual is used for quick orientation and uses lots of hyperlinks. Therefore it's not suitable for 'linear' reading.

2 How does it work?

After running the program syncer.exe (no matter if directly or from Avast32) such things will happen:

From local file servers.def which contains server list, the server is chosen (how it's done, see this chapter).

From this server is downloaded newer file servers.def and server is chosen again.

If there's is file groups.def with group definitions on the server, its content is downloaded and the group is chosen following the rules in the file.

File index*.def gets downloaded. Its filename contains name of group from previous step. This file contains list of commands and packages which will be installed.

The the file packages.def with package definitions will be downloaded.

Files index*.def & packages.def will be interpreted and update process will start. If the program updates itself, it possible that the process will be restarted.

Process ends now. Program may ask for rebooting the system. (It will never do so if only antivirus database is updated).

Remark: all files *.def are downloaded only if they don't exist on local disk or have different modification times than files on the server.

3 Security?

Yes. :-)

Every binary file downloaded on local computer is checked for the following:

- same length
- same MD5 checksum - 128 bits
- valid signature of ALWIL Software - 320 bits

Used algorithm for digital signatures has private key of 1024 bits. This algorithm is asymmetric and it's practically unbreakable without having private key. Format of our binary files is internally protected and undocumented. Our web servers are protected too.

Again, all the steps for potential hacker:

- reverse engineer the file format of our binary files and create such a files - very hard
- compute MD5 checksum - very easy
- create digital signature of ALWIL Software - almost impossible
- create package.def file - very easy
- put the file on the server - depends on server's security

4 Mirroring

The mirroring means, that you put a copy (mirror) of our files to your local servers. For you, as administrator of larger network it is good idea to maintain such a mirror. The updates will be faster and more reliable from your servers and it also gives you an opportunity to supervise the package versions, which are installed onto the target workstations.

Both the general instructions and the automatic and easy solution of mirroring via tools written in Perl are described in the next subchapters.

4.1 How to mirror - general description

At first it is necessary to create a folder, where the newer files from our distribution directory will be regularly copied. Then download all the files to it. It is a good idea to have some utility for downloading, especially if it would be capable of downloading the new files only. The URL to our server can be found in servers.def files.

Then create the following files:

- servers.def with settings of your servers.
- groups.def with definition of computer groups.
- all files index.*.def with package selection according to your groups.

The groups.def and index.*.def files should be created only in case, that you would like to restrict installation of some packages. You can leave out these files if you would like to have the same installation as it is on our servers. The standard files defined by us will be used then.

If you would like to put your files onto the HTTP server, it is necessary to create *.stamp files.

Create a distribution directory by making a copy of files from a mirror of our server and prepared *.def files.

Then copy (upload) all the files to your distribution servers.

The solution in v Perl is described here.

The regular mirror maintainance is described here.

4.2 Repeated tasks - general description

Basically, it depends on you. It is possible, that you would like to adjust the servers.def or groups.def files from time to time.

The certain exception could be the index.*.def files, where some packages will be supplemented in the future. You should update your index.*.def file then.

Otherwise the regular maintenance is not very hard - you just need to mirror our files and copy the final distribution directory to your distribution servers.

4.3 How to mirror - the solution

This solution is written in Perl. See the frequently asked questions about Perl.

Decompress the utility package including the subfolders.
To mirror subfolder put servers.def file from our server. (It is e.g. part of the installation.)
Adjust the mirror.cfg configuration file in cfgs folder. Try to start mirror.pl program. It should download the up-to-date state of the files to mirror folder. You can check the error reports (file mirrorDATE.log) in logs folder.

By now, you should have the same distribution on all servers. V tomto okamžiku byste na všech serverech měli mít stejnou distribuci.

Now just setup clients and you are done.

4.4 Periodic maintainance - completed solution

This solution is written in Perl. See its FAQ.

It completely depends on you. It is possible that through the time you can feel the urge to change the servers.def and groups.def files.

Some exception is created by index.*.def files, which will be updated by more packages. You should than update your index.*.def file (if you have decided to restrict the maximum version of packages). Program mkdistro.pl will tell you , how different are yours and ours files index.*.def.

But the periodic maintainance itself is without problems - it is sufficient to mirror our files and copy the resulting distribution directory on your distribution servers.

Automatically, using for example at program, you should periodically run these files in this order:

```
mirror.pl
cldistro.pl
mkdistro.pl
chdistro.pl
uploader.pl
```

All these tasks together are performed by doall.pl program, which performs all of the above-mentioned tasks and sends e-mail to the administrator.

It's good to say, that program at doesn't allow change to shared drives, so the batch file should be authorized like this net use x: \\server\path /USER:DOMAIN\user password

4.5 iAVS Client setup

If the Avast32 is freshly installed, it is sufficient to add syncer.ini and servers.def files to the installation. More informations are in Administrator installation settings documentation.

If it is needet to change the client/server settings with Avast32, use the common update of the program using the widely known *.tm and *.tmd files. New program avupdatem which is in the latest installations, allows to update evet the syncer.ini and servers.def files. All you need is just to upload these files into the directory, where do you commonly place the *.tm files and create *.tm files using the tmc.exe program. More informations in AUPD.TXT file, added to Avast32 installation.

5 How is server chosen?

In the meantime it's random. It will change in near future (it will use priorities or ping times).

6 File servers.def

File servers.def contains information of all the servers containing the directory with all update files. Its format is Windows INI.

Remark: How to break everything: Put on your server invalid file servers.def. It will get distributed on all clients and all of them will suddenly be unable to do updates anymore.

Section [Servers]

count = number - number of server definitions in file.

Section [ServerN]

N in name of this group is number of server. First server has number 0.

name = string - display name of the server.

url = string - this url must start with http:// or ftp:// and must follow all rules for urls.

StatsCollectType = string - names the method for statistics upload.

- cgi - data are sent using HTTP POST request to cgi script

StatsCollectUrl = string - url to cgi script

Example:

```
[servers]
count=2
```

```
[Server0]
name=Main server
url=http://www.company.com/iavs
```

```
[Server1]
name=Other server
url=file://\server\share\iavs
```


7 Groups.def file

Groups.def file defines how the computers are divided in groups. The format of the file is:

```
group=rules;
```

Group is either the name of group (letter used for the definition of index.def files) or is empty and in this case the computer has the update forbidden.

The rules are specified like this:

```
variable:value
```

Variables available for testing:

- winname - Windows computer name
- ip - IP address of the computer
- ipname - DNS/WINS computer name
- sys - computer operating system

winname:

ipname:

The strings are tested. It is case insensitive and wildcharacters of ? and * are available.

sys:

Testing with these values:

- 9x - Windows 9x
- nt - Windows NT
- 95 - Windows 95
- 98 - Windows 98
- nt4 - Windows NT 4
- 2000 - Windows 2000

ip:

It is either the direct number testing (192.168.1.243) or mask testing (address/bitmask e.g. 192.168.1.0/24 for C subnet).

These operators are available (associated from the left, logical and has greater value than logical or)

- () - the sequence of processing
- & - logical and
- and
- | - logical or
- or
- ! - logical not
- not

The items are processed from the top to down. The matching will end in first time, when it is a match. If no match is found, the computer is automatically deleted from all group.

Example 1:

```
bleedingedge=( ip:192.168.11.0/24 | ip:192.168.12.0/24 ) & sys:nt;  
nt=sys:nt;  
others=winname:*;
```

This means, that only three groups are available ("all" group is not, because the last rule will `kill` all the unused). The first group is created by NT computers from 192.168.11 and 192.168.12 subnet. Second group is created just by NT computers and third group by all the others. **Example 2:**

```
vip=winname:boss;  
vip=winname:boss_secretary;  
vip=winname:manager1;
```

Here we have got two groups. In first one are just the "special" users, others are in "all" group.

Example 3:

```
manager=ip:192.168.11.0/24 | ip:192.168.10.234 | ip:192.168.10.235 | ip:192.168.10.236  
manager=ip:192.168.11.0/24  
manager=ip:192.168.10.234  
manager=ip:192.168.10.235  
manager=ip:192.168.10.236
```

Both choices above are equal. The lower one is more well-arranged.

If the groups.def file does not exist, the group all is default for all.

8 Index.*.def files

Index.*.def files describe the packages and the way, how they are installed. Instead of the star, name of group is inserted as is specified in groups.def file.

Each row has this format:
command:package;

or:
command:package version;

Command:

- install - will start installation of the package independently, even if the previous version exist on the local machine
- update - will update package
- ignore - no actions for the packages

Packages: name of the package as is specified in packages.def file. In present time only vps32 package is available.

Version: Optional parameter specifying, which maximum version should be updated. It could be useful, when the computers are divided in two groups: one for normal users, where stable and tested configuration is running, and second for powerusers who uses the newest versions.

each of the index.def files have default first line, which cannot be affected. It is the install syncer; line, which ensures, that the update program update as the first itself.

9 Packages.def file

Packages.def file includes rules for update. Into this file you should not modify this file. It's format will change through the time. For selection of installed packages files index.*.def exist.

Format is as follows:

```
packagedef( package_name rules_for_package_recognition )  
  
new_version: old_version  
  
command1;  
command2;  
...
```

Warning: you should not make any changes to this file, because the commands are not specified in any public documentation. Every change can result in inoperability of the system.

10 Files *.stamp

*.stamp files exist for every *.def file (whole name is than *.def.stamp). They include the time of creation of the *.def file. Format is a decadic number showing the number of seconds from 1.1.1970 (epoch). (For C programmers: time_t);

Before downloading a *.def file the stamp file is downloaded and checks, whether the time status of the file is the same as the file on local drive. If they are same, the work with the existing file is resumed. It saves time, when the files on the server have not been changed.

11 File mirror.def

File mirror.def includes data for mirror programs. Structure is simple, there are three columns divided by comma on the rows. They are: filename, filesize and MD5 hash.

This file is used by mirror.pl utility. It is not needed for the program iAVS.

12 File lock.def.stamp

File lock.def.stamp can appear on source server, if the update is in progress. The format is the same, as are another stamp files. Its time status shows the time of the start of the server update. If the time is far too old (this is very relative), it is possible, that during the update an error occurred and stayed 'locked'. We suggest that you inform the server administrator.

13 Syncer.ini file

In sincer.ini file is a client setup. This configuration is saved on the station, in the same directory where syncer.exe is located. It should be configured by network administrator and distributed by present update technology.

Group [Syncer]

Mode = number - specifies the type of windows open.

- 0 - large window with Cancel button
- 1 - large window with Cancel button, terminates itself in the end
- 2 - large window without the Cancel button
- 3 - widow has just the main progress bar
- 4 - just a tray icon without the possibility of enlarging
- 1000 - no window

Lang = number - specifies language.

- %Avast32% - uses the same configuration as Avast32
- 0409 - english
- 0405 - czech

HttpProxy = string - specifies the port of HTTP proxy server.

- proxy:port - full proxy identification
- %Avast32% - uses the same setting as Avast32

HideInAvast = number - specifies, if there should be the update icon in Avast32 visible.

- 0 - shown
- 1 - hidden

DoNotAskForReboot = number - specifies, if the client should ask about the restart of the system.

- 0 - will ask
- 1 - will not ask

Example:

```
[Syncer]
mode=1
lang=0405
httpproxy=proxy.organisation.com:8080
```


14 Files with distribution description

In version 1.10 there's a new possibility to work with more than one distribution. There's also possibility to work with one distribution only, you don't need to enter it's symbolic name. It's name is the 'default', def dir is 'deffiles' and dir dir ;) is 'distrib'.

Files have name symbolic_name.dst and reside in dir cfgs.

name = string - display name def = string - from which dir are files *.def. dir = string - where to save results (mkdistro.pl).

Example:

```
name=First Distribution
def=defsr
dir=distr
```

This file is named first.dst, is in dir cfgs and is referenced by name first.

15 At command

At command is simple task scheduler running under the WindowsNT environment. It is a standard part of WindowsNT. To use this, a Scheduler service must run. Run the Control panel and inside the services applet. Find the Scheduler service. If it is not running, change the status to Automatic, and run the service. If you have not found the service at all, you have probably installed MS Task Scheduler, which is instead the Scheduler service. Find the Task Scheduler service and make sure it will run

If the specific service is running, you can run from the command prompt at program. It will list all of the scheduled tasks. If you need help, try:
at /?

First run task

Let's take a look on, how to run a task. First of all, create a simple batch file run.bat in c:\.run.bat:
dir > c:\run.out

Schedule task like this:

```
at 16:30 c:\run.bat
```

In 16:30 a C:\run.out will be created. Its content will be the content of the directory where run.bat was ran. By easy analysis we can see, that it was C:\winnt\system32.

We love windows

System WindowsNT is an advanced operating system, which allows the creation of applications which communicate with user differently than through the command line. We will now try to schedule and starting of program like this.

```
at 10:00 notepad.exe
```

And in ten o'clock, nothing. By searching the task list in Task Manager we will see, that notepad is running, but invisible. It's because that it was ran in the system account and that the user currently logged in doesn't have rights of access. For scheduling we must now forget about all interactive tasks. Programs ran by must not require any user interference and must terminate themselves.

We love network

Tasks scheduled by the at program cannot access network drives, mapped or UNC. Scheduled tasks are ran under the local system account, which doesn't have the right to access network. You must count with this, while creating the scheduled tasks.

Periodic tasks

This, for a change, works. Using the /every: switch you can specify days in a week or month, when the task should run. Examples:

```
at 14:00 /EVERY:m,t,w,th,f,s,su c:\run.bat
```

will run this task every day at 14:00.

```
at 14:00 /EVERY:15 c:\run.bat
```

Avast32 Antivirus

will run this task every 15. day in the month at 14:00.

16 Program tmcr.exe

Program tmcr.exe serves for creating a file with time stamp. This file is then used for file distribution using the automatic update mechanism using tmd. examples:

tmcr.exe syncer.ini

After running this program a syncer.tm file is created in actual directory.

If you want to distribute your own files syncer.ini and servers.def on workstations, create a syncer.tm and servers.tm and upload them to directory, to where you save files for Avast32 update. (files sp32tm and sp32.tmd).

17 Mirror program.pl

Description:

Program mirror.pl is made to create automatic update into the mirror directory. Protocols supported are http:// and file://.

Program creates logs into the logs directory. Name of each file is mirrorYYYYM-MDD_HHMMSS.log.

Dependencies:

program md5.exe - is part of the shipping

modul LWP - in ActiveState of Perl distribution is installed automatically

Command line parameters:

-singlelog - output log file have mirror.log name

Exit codes:

-2 - some files have not been copied

-1 - fatale error

0 - no files have been transferred, source server was not changed

1 - some of the new files have been transferred

Configuration files:

In cfigs directory is mirror.cfg file.

retry_count = number - number of attempts, when the files have failed to download all files. (Default: 3)

retry_interval = number - time in minutes, whe program waits for the retry. (Default: 5)

http_proxy = string - address http proxy in http://server:port format.

company = string - company name. It is used due to the errors with mirroring.
company_admin = string - email on iAVS administrator.

Change to version 1.00: company/administrator identification is now compulsory, program will not run without it.

18 Program cldistro.pl

Description:

This program only deletes the contents of directory distrib.

Dependencies:

None.

Command line parameters:

None or distribution name.

Return values:

Undefined.

19 Mkdistro.pl program

Description:

This program create for all *.def files in deffiles directory files *.stamp and copies all of these files int the distrib directory. Into this directory it also copies all of the remaining files from mirror directory.

It can also find difference between your and our index.*.def files.

Program also creates a log file mkdistro.log into the logs directory.

Dependations:

program md5.exe - is a part of the shipping

modul File::Copy - in ActiveState of Perl distribution this module installed automaticaly

Command line parameters:

None or distribution name.

Exit codes:

-1 - fatal error

0 - everything is in order

1 - minimally one problem with index.*.def files.

20 Chdistro.pl program

Description:

This program tests distrib directory for possible present errors.

Test order:

- for all groups in groups.def file files index.*.def must exist.
- All packages, which are referred to from all index.*.def files must be defined in packages.def file.
- for all files, which are inscribed in mirror.def file, must read, that they have according lengths and check sums.

Program creates logs in logs directory. Name of every file is chdistroYYYYM-MDD_HHMMSS.log.

Dependations:

program md5.exe - is part of shipping

Command line parameters:

-singlelog - resulting log is named chdistro.log
it is possible to specify distribution name.

Exit codes:

-1 - fatal error
0 - all is in order

21 Uploader.pl program

Description:

This program uploads the content of distrib directory on distribution servers. It uses FTP transfer or direct copying of files.

Program creates logs into the logs directory. Name of each file is: uploaderYYYYM-MDD_HHMMSS.log.

Program creates *.upl files in cfgs directory. these files contains the last transferred status of files.

Dependancies:

program md5.exe - is part of the shipping

module Net::FTP - is not part of standart ActiveState installation. this module is needed for FTP update. You can install it by running the ppm.bat file in that directory, where perl.exe is located. Than write install libnet and than just answer the questions. Internet connection is required.

module File::Copy - is part of standart shipping of ActiveState Perl.

command line parameters:

-singlelog - finished log has uploader.log name

-all - update all files, not just changes.

it is possible to specify distribution name.

Exit codes:

-1 - fatal error

0 - no files transferend, target server is synchronized

1 - some files transferred

Configuration files:

For using this program, you need to have set the configuration files in cfgs directory:

Configuration file uploader.cfg

Only two interesting parameters are avaiable:

dry_run = number - if the program is on the dry run or is working normally.

- 0 - normal function
- 1 - just describes, what it would do

the names *.srv, devided by commas. Example: servers=srv1,srv2 will execute the update of servers as it is described in srv1.srv and srv2.srv.

If more than one distribution is used, use line 'servers_distroname' for each distribution instead of one 'servers' line. e.g. for distribution 'first' use assignment to 'servers_first'.

Configuration files parameters of *.srv

Parameters:

type = string - upload type

- ftp - uses FTP upload protocol
- file - just copy the files

dest_path = string - path of storage area for uploaded files (ftp and file)

retry_count = number - Number of retries when all of the files were not uploaded. (Default: 3)

retry_interval = number - time in minutes, in which the program should wait between retries. (Default: 5)

ftp_site = string - FTP server address (FTP only)

ftp_port = number - FTP server port (FTP only)

ftp_firewall = string - FTP proxy address (FTP only)

ftp_login = string - username for FTP server (FTP only)

ftp_password = string - password for FTP server access (FTP only)

Example 1:

```
type=file
dest_path=x:\path\avast
```

Example 2:

```
type=ftp
ftp_site=ftp.organistaion.com
ftp_login=uploader
ftp_password=heslo
dest_path=/pub/avast/iavs
```

22 Program doall.pl

Description:

This program just calls all the others. It takes care of their exit codes and creates a report from these informations.

Log is created in logs with doall.log filename.

Dependatopns:

It depends on all other programs.

module Net::SMTP - is not component of standard ActiveState installation. This module is needed only for sending mail. To install it, run program ppm.bat from the directory where you have perl.exe, write install libnet and then only answer the questions. Connection to the Internet is required.

Must have set path of perl.exe program.

Command line parameters:

None available.

Exit Codes:

-1 - fatal error

0 - all is in order.

Configuration files:

In cfgs directory is doall.cfg file located.

distros = strings delimited with commas - contains names of all necessary files with the information about distribution.

smtp_server = string - SMTP server name for outgoing mail.

smtp_port = number - port, on which SMTP server runs.

mail_to = string - e-mail address, where to send mail.

mail_from = string - optional parameter, it is setted, when server needs a valid username of the sender.

dont_send_positive_mails = number - if all called programs returns 0 as a exit code, it won't send an email.

- 0 - send mails everytime
- 1 - send mails only, when something interesting had happened

23 Program md5.exe

Command line:
md5.exe file

Returns MD5 message digest (checksum).

Program is used internally by our utilities.

24 Changes

This page describes differences against previous version of this document.

Version 1.00

- First version.

Version 1.10

- Added the possibility to work with more distributions.
- Tested work under UNIX. FAQ.
- Mandatory identification of administrator. See mirror.cfg.
- Bug fixes.

25 Frequently asked questions

Little Perl FAQ is here.

Q:We don't like Perl and command line. Will there be GUI version?

A:Don't count on it in near future. But in this manual are all formats and algorithms documented...so that you can do it by yourself :)

26 Frequently Asked Questions - Perl

Because most of our utilities for iAVS are written in Perl, here are few frequently asked questions:

Q:What is Perl?

A:Perl is something you already use so that you can skip this chapter or something you will really love :-)

It's widespread scripting language, mainly used in UNIX enviroments. If you know shel script, asw, sed & C, you will have no problems with Perl. In other case you shoud not have any problems, but knowledge of C is useful.

Q:Is it perfect language?

A:Probably not. But it's very powerful language and has high learning curve.

Q:Where to get Perl?

A:Here: <http://www.activestate.com>. Also visit <http://www.perl.com>.

Q:How much does it cost?

A:Nothing. It's free.

Q:Installation of Perl.

A:Follow the installation instructions in Perl distribution. All of our utilities were tested on ActiveState Perl 521/522.

Q:How to install additional packages?

A:Run program ppm supplied in ActiveState distribution and issue command install packagename and follow the instructions on screen.

Q:Is Perl necessary for running iAVS?

A:No. It's used only for administration utilities.

Q:Perl literature.

A:If the html documentation is not enough, get the 'Camel Book'. TBD.

27 Frequently Asked Questions - Unix

Q:Are there any difference in Unix installation?

A: Yes, it needs more knowledge. Unpack lindist.zip, set correct rights/owners for *.pl and *.pm files (chmod 755). Install needed packages for Perl. (perl -MCPAN -e shell;). We need LWP, POSIX, Bundle::libnet. If you don't have /usr/bin/perl, make symlink or edit scripts.

Q: Is specific Unix needed?

A: Should not be. But scripts were tested only under Red Hat Linux 7.0.

28 Acknowledgements

These tools and libraries were used in iAVS development:

Compilers:

Microsoft Visual C++ 6.0
GNU DJGPP 2.0

Scripting languages:

Perl

Libraries:

Compression library Zlib (Copyright (c)1995-1998 Jean-loup Gailly and Mark Adler)
Routines for computing MD5 (Copyright (c)1999 Aladdin Enterprises, L.Peter Deutsch)
Routines for binary differences (Copyright (c)2000 ALWIL Software, Jindrich Kubec,
inspired by JENDIF (c)1997 Viliam Mlich)
Routines for asymmetric signatures

Testing servers:

Red Hat Linux 6.1
Debian 2.0
BSDI 3.0

Testing HTTP servers:

Netscape FastTrack
Apache for Linux
Apache for Win32