SAVAPI Frequently Asked Questions

Contents:

1. How many compressed formats does the Engine recognize?

Currently, the Engine unpacks the following formats:

- 7z, self extracting 7z
- Ace, self extracting Ace
- Arj, self extracting Arj
- AutoIt Installer
- Base64 encoding scheme
- Binhex (Mac)
- BZip2
- Cab, self extracting Cab
- CHM (Compressed HTML Help)
- CPIO SVR4
- GZip
- JAR Java ARchive
- LZH/LHA, self extracting LZH/LHA
- MSCOMPRESS
- MacBinary
- MIME type
- Nullsoft Installer
- PDF (Portable Document Format)
- PGP signed message
- Rar, self extracting Rar
- RPM package file format
- SAPCar
- TAR
- TNEF
- UUE encoding scheme
- Zip, self extracting Zip
- Zoo
- ISO 9660

A special category of archives are the mailboxes:

BSD , Netscape/Mozilla , Eudora, Squid Cache, Pegasus Mailbox, MS Outlook (PST version 2003)

2. Is SAVAPI a real-time scanner?

No, SAVAPI is only a scanning Engine and it doesn't monitor any file operations. It is the application's responsibility to implement the real-time scanner, which intercepts the file operations.

3. Does SAVAPI contain anti-adware and Registry protection?

No, please see Is SAVAPI a real-time scanner?

4. What is product ID 117?

This is the default product ID of SAVAPI. However, we do not deliver any keys with this product ID inside, so it can't be actually used. You have to change this value to the product ID you received from Avira Sales.

5. How do you manage license keys?

Each SAVAPI partner receives a unique Product ID from Avira in form of a license key.

This key allows you to receive updates to our components (SAVAPI binaries, Engine, Signatures) for a predefined period of time.

This key has to be integrated in your product which uses SAVAPI and always shipped to your customers along with SAVAPI.

6. What is hbedv.key?

This file represents the license file of SAVAPI (library and service).

Your company will receive one dedicated *hbedv.key* which can be delivered to all your customers together with your own key. Please note that you must always deliver the *hbedv.key* file, otherwise SAVAPI will not start.

This key is unique for you as a company. It contains (among other things) the expiration date and the product ID that you receive from Avira (in the same time with the new key).

Your application using SAVAPI (library or service) must send the product ID to SAVAPI, which uses it to initialize the Engine.

If the product ID and the key do not match, then the product cannot scan, nor update.

The key you receive from Avira is usually a maximum 5-year license, which is renewed or blacklisted whenever necessary. Practically, it is up to you if your customers will even see that they have two licenses. For them, *hbedv.key* can be just another file included in the package.

Of course, your product must make sure that it doesn't expire, otherwise the customers cannot update anymore.

Even if the key expired, the product scans, using its existing signatures.

7. Does SAVAPI SDK provide anti-spyware functionality?

No, SAVAPI is not an anti-spyware product, nor a dedicated anti-rootkit product. It provides only basic AntiRootkit detection via VDF signatures.

8. Could Avira provide a client template developed with SAVAPI?

We deliver the SAVAPI SDK with an example written in C and one in C++ which set, get various options and scan a file.

9. Can Avira develop a SAVAPI client in Java/C#/Perl/Python/Ruby?

Avira cannot do this for you. However, if you observe the SAVAPI protocol, you can implement such a client by yourself.

10. Do SAVAPI and the other tools needed for its development support 64-bit systems?

Yes, SAVAPI and all its components are natively compiled and can be used on 64-bit operating systems. Of course, you can still run the 32-bit version on a 64-bit OS in emulation mode.

11. What version of compiler is the SAVAPI development team using?

We are developing on both Windows and UNIX.

On Windows we are using the compiler from Visual Studio .Net 2008 (latest SP).

On UNIX, we compile with various gcc versions (whenever possible, the version used is over gcc 4.0), using the required libc version for the respective platform.

Please note that SAVAPI is completely written in C.

12. What is SAVAPI service and SAVAPI library?

SAVAPI is a simple scanning **service**. It receives a file to scan and it delivers a result. A client can communicate with the service via a socket (TCP/IP socket on Windows and Unix systems, or local socket on Unix systems) using a special protocol.

Additionally, there is also a library version, which comes in two flavours:

- There is the **SAVAPI client library**, which is just a wrapper over the socket protocol. A client can communicate with the SAVAPI service just by using the provided API without having to deal with the socket communication itself.
- Then there is the **SAVAPI library** which can be used to access the scanning features. It is useful if you want to implement your own scanning application, without using the SAVAPI service.

The two libraries use the same API. Practically, you write the code once and you can choose anytime, without any code changes, in which way to use SAVAPI: as a service (with SAVAPI client library or with the SAVAPI protocol itself) or as a library (with SAVAPI library).

13. How can I clean a file?

Scanning and cleaning can be used together in one step or separately.

If you use the SAVAPI service and choose to communicate with it through the SAVAPI protocol, you have to use the `REPAIR` option.

If you use the API (in both flavours SAVAPI client library or SAVAPI library) and the flag `SAVAPI3_OPTION_REPAIR` is set to 1 before the scan command, the Avira Engine will try to clean the file in one step.

If you want first to detect the malware and then ask the users if they want to try to clean it, then you need to scan the file twice: once without this flag set, then if the file is reported as infected, with the flag set.

Note that the Avira Engine can only clean the files which are "cleanable".

This means that the variable member "removable" from the structure `SAVAPI3_MALWARE_INFO` must be set to 1 after a successful scan by the Engine.

It will not remove or move files to quarantine, because it simply cannot do these actions.

Internally in the VDFs, each malware has assigned a flag and if this flag is set to "cleanable", then it points to a cleaning routine in the Engine's code.

Only some macroviruses and the old file infectors are cleanable. Nowadays this option cannot be really used like it was during the 90s, when the only malware found in the wild were the file infectors.

14. Are the DLLs signed by Avira? Can we sign them ourselves?

The binaries are specially signed by Avira, with its own algorithm, in order to check their integrity (not with a certificate). Once the DLL or EXE is changed, the product doesn't start anymore.

Because of this self-test, you cannot sign the binaries, nor change them in any way.

15. Is it possible to pass some kind of "userdata" to the callbacks for every scanned file?

Yes, it is. Please use the parameter "userdata" from `SAVAPI3_callback_data` structure and use the function `SAVAPI3_set_user_data` to set the value.

```
int SAVAPI3_EXP SAVAPI3_set_user_data(SAVAPI3_FD *savapi_fd, void
*user_data);
```

16. How can I load SAVAPI library in Windows from another directory?

Assuming the application is located in folder A, which loads *savapi3.dll* from folder B (A != B). All DLL dependencies of *savapi3.dll* are located in folder B.

The problem is, when the application calls `LoadLibrary("B\savapi3.dll")`, Windows tries to load the dependencies of *savapi3.dll* in the following directories, in this order:

1. The directory where the executable module for the current process is located.
2. The current directory.
3. The Windows system directory.
4. The Windows directory.
5. The directories listed in the PATH environment variable.

Since the B folder is not in one of those categories, the application will fail to load the dll.

The solution is very simple: Windows API has a function called `SetEnvironmentVariable`, which sets the contents of the specified environment variable for the current process.

So, all you have to do is the following:

Call this function

```
SetEnvironmentVariable("path","%path%;B");
```

and then

```
LoadLibrary("savapi3.dll")
```

17. How can I debug the SAVAPI test program from MS Visual Studio?

In order to be able to start the test program, you need to configure the project as follows:

1. Go to **Project properties - Debugging**
2. Set for all targets **Working Directory** to *../win32/bin*
3. Set **Command Arguments**:

   - for Clientlib: <full path to your test file>, eg: `c:\eicar.com`

   - for Savapi3lib: ./ ./ <full path to your test file>, eg: `./ ./ c:\eicar.com`

4. Now you can press **F5** and start debugging the program.

18. Can the Update feature get any files from an update server, as well as virus pattern files?

Avira Updater is a utility which allows Avira partners to mirror the files on their servers.

It is also allowed to ship this utility to your customers with the observation that they must update from your own servers and not from those of Avira.

The Avira Updater utility can download and install only the files which belong to the SAVAPI product. This includes SAVAPI binaries, Engine files containing detection routines and virus pattern files (VDF files).

19. How does the Updater (the avupdate program) actually work?

The Engine binaries and the VDF updates are updated by SAVAPI, using an external update program provided by Avira. Please read the document *avupdater.txt*, included in the documentation archive (available in the forum).

19.1 How to perform an update when using the library

this update method assumes that you use savapi3lib-win32-en.idx . For a different operating system, the steps are similar.

When loading the library in your program, you have the complete control over the SAVAPI functionality, because the avupdate program is not able to stop your program. In order to be able to update using the avupdate program, you need to split the update process in some steps:

1. Mirror the files locally on the client computer, using the "mirror" parameter of avupdate. You have to provide a special configuration file and install the files in some directory. At each scheduled interval, the avupdate will try to update the mirrored files. If it doesn't find anything new on the website, it simply returns 1 (nothing to update). If it updates

something correctly, it returns 0. You can even choose which SAVAPI modules to update. Just have a look at the 2 examples of *avupdate-*.conf* contained in the SDK (Scanner, Engine, Signatures).

2. If the program returns 0, then you know that you have to reinitialize the library. At this stage the files are still in the directory where you mirrored the structure from the update server. In order to perform the update on the file set which is in use, you must perform a so called "intranet update". This means nothing more than executing the avupdate in intranet mode, using as intranet server the directory where the mirrored structure resides (please read the documentation for more details).
   Before executing the `avupdate` program in intranet mode, you must terminate all SAVAPI instances and unload the library.
   Execute the avupdate in intranet mode. This will update from the directory you specified and install the files into the installation directory where SAVAPI resides.
3. Load the library again and re-create the instances.

19.2 How to perform an update when using the service

this update method assumes that you use savapi3lib-win32-en.idx . For a different operating system, the steps are similar.

The service is a separated process which is loaded by the operating system at startup. However, the steps required to update the files are very similar.

Your application is using the *savapi3client.dll* in order to be able to communicate with the service. Because this library and the SAVAPI (.exe) service must always be compatible with each other, the avupdate program will try to update both in the same time. This means that in step 2 of the update process, you must unload the *savapi3client* library as well. The avupdate program will restart the SAVAPI service, if required. In step 3, you must reload the library again.

If your application allows service interruptions, you can simply stop your application in step 2 and start it in step 3.

20. What would be the update server address and port?

There are many addresses Avira provides to its partners for getting Engine and Signatures updates. Please use the addresses provided as example in the configuration files from the SDK.