Table of Contents

1. SAVAPI

1.1 OS Support

Added support for Mac OS X v10.7

The extension of the libraries on Mac OS X is now changed to *.dylib*.

> **Note**
> The Mac OS X Support will only be available in the beta version.

1.2 SDK Delivery

The format in which the SDKs are delivered has changed from two archives containing all builds for UNIX and Windows respectively, to an archive containing the build for each supported operating system.

This also simplified the use of the SDK since there is no need to have an installation script for the SDK.

Following packages are now separately available:

- savapi-sdk-linux_glibc22.zip
- savapi-sdk-linux_glibc24_x86_64.zip
- savapi-sdk-win32.zip
- savapi-sdk-win64.zip
- savapi-sdk-solaris_sparc.zip
- savapi-sdk-solaris_sparc64_v8.zip
- savapi-sdk-freebsd_v62.zip
- savapi-sdk-openbsd_v39.zip

> **Note**
> The Mac OS X packages are also separately delivered, but they are in beta phase.

1.3 Functionality improvements

- Improved error reporting in SAVAPI Library
- Reorganized error reporting in SAVAPI Client Library

- Strict validation is done for SAVAPI API functions input parameters

> **Note**
> In isolated cases, SAVAPI 3.2 was not reporting an error, if a function call was not adequately executed, but this issue has been fixed in SAVAPI 3.3.

- Both libraries report similar error codes for similar error cases
- Partial error reporting reorganization in SAVAPI Service
- Enhanced validation errors for parsed options from command line and configuration file
- Consistent error messages and error codes for all parsed options
- Consistent option validation for all options from command line, configuration file, protocol, SAVAPI Library
- Now all options from SAVAPI are validated in the same way
- Unified logging system in SAVAPI libraries
- More logging messages added in SAVAPI Library
- Added logging support in SAVAPI Client Library
- APR was updated to version 1.4.2 and APR-util to version 1.3.10.
- Added verification for maximum path length to scanned object
- Add support for new engine error code `AVE_ERROR_MEMORY_LIMIT_REACHED`
- Fixed scanning issues for some Adobe installation kits
- Files with filename starting with whitespace can now be scanned
- Fixed some protocol command inconsistencies
- Added versioning support for Library

1.4 Usability improvements

- New SDK usage examples:
  - multithreaded example
  - directory scanners example
- Reorganized all SDK usage examples:
  - new code sources with consistent coding style
  - separate source for each usage example per library
- Cleanup and improved usability of SAVAPI interface *savapi3.h*:
  - more comments to library API and definition
  - document option default values
- Unicode support fixed for SAVAPI Client Library

- Added a SIGSEGV handler in SAVAPI Service
- Scan optimizations done in SAVAPI Library
- Updated solution files for Visual Studio 2010

1.5 Fixes

Fixed a race condition at SAVAPI Service startup.

1.6 New features

Scanning and reporting of PGP encrypted objects

There is a new option `SAVAPI3_OPTION_PGP_SCAN` which enables or disables the scanning and reporting of PGP encrypted objects. If you are integrating SAVAPI in a mail filtering solution and your solution filters and blocks emails which are PGP encrypted then activate this option. Once activated, it reports the objects it scans as encrypted and the product can block them. Its default value is 0 (deactivated).

FileName Encoding - this feature affects all scanning actions

When activated, the filename arguments of the `SCAN` command must be encoded in their hexadecimal representation: every byte of the filename must be represented by its hexadecimal value, with two hexadecimal digits.

For example, in ASCII encoding *savapi.exe* gives `7361766170692e657865` as hexadecimal equivalent. Also, the filenames in the answers received from SAVAPI will be encoded in the same way.

> **Note**
> The `ENCODE_FILENAMES` option is by default deactivated.

If the option `ENCODE_FILENAMES` is enabled, SAVAPI Server will assume that filenames received in scan requests are encoded and will decode them. If the received filename cannot be decoded, the error message "*350 filename decoding failed*" will be returned.

```
SET ENCODE_FILENAMES 1
100 ENCODE_FILENAMES:1
SCAN 2061620a63
310 Eicar-Test-Signature ; virus ; Contains code of the Eicar-Test-
Signature virus
319 OK
```

Scan an archive " *ab\nc.zip*" containing an "*eicar.com*" test file:

```
100 SAVAPI:3.1
SET ARCHIVE_SCAN 1
100 ARCHIVE_SCAN:1
SET ENCODE_FILENAMES 1
100 ENCODE_FILENAMES:1
SCAN 2061620a632e7a6970
310 65696361722e636f6d <<< Eicar-Test-Signature ; virus ; Contains
code of the Eicar-Test-Signature virus
319 OK
```

If the option `ENCODE_FILENAMES` is enabled, then any filename in scan responses written on socket by SAVAPI Server will have all characters encoded.

SAVAPI CALLBACK_PRE_SCAN is triggered for each file in an archive, except non-native containers

A non-native container is any file which contains embedded objects, pictures, other media types than the usual format of the file. For example, we can have a PDF or Word document which contains embedded objects like music, movies, executables, links, etc. These files are not archives (like ZIP, RAR, 7Z, etc.) but very complex files.

1.7 Backward compatibility issues

savapi_post.sh is no longer updated

This means that the integrator is allowed to change the file as needed. It will no longer be overwritten with each update.

SHUTDOWN command is obsolete

This means that it is no longer possible to stop the daemon via a command.

Callback registering

The callback registering changes in SAVAPI library (only one callback can be registered per id).

It is no longer possible to register one callback for multiple actions, as this can destabilize SAVAPI.

Removed user and group from SocketPermissions option

There are still the User/Group settings available to specify these settings.

Engine reloading

Daemon/ Service:

The `--reload-engine` command is now deprecated in daemon. In order to reload the engine, you need to start and stop SAVAPI. This behavior is different on Windows than on UNIX (incl. Mac OS). On UNIX, it is enough to start a new instance of the daemon and this new instance will take over, from where the old instance stopped. There should be no interruption of service. On Windows, the service must be stopped and started again.

Library:

The `SAVAPI3_reload_engine` function is enhanced by the function `SAVAPI3_reload_engine_ex`, which requires that the engine and VDFs are reloaded from a different location than the previous one.

Through the `SAVAPI3_reload_engine_ex()` function, SAVAPI library offers the possibility to reload the engine and .vdf files on-the-fly, without reloading/ restarting the whole library/ process.

The function has as input parameter the same type (structure) passed at global Initialization (`SAVAPI3_GLOBAL_INIT`) which contains, among others, the new paths to the engine and .vdf folders.

If the engine and .vdf files are placed in the same location, the user can only specify the path to the engine folder - `SAVAPI3_GLOBAL_INIT::engine_dirpath`. The new function is meant to replace the old reloading function `SAVAPI3_reload_engine()`.

The behavior of the function `SAVAPI3_reload_engine()` was until now pretty similar on UNIX and Windows but not identical. The only difference was that, on Windows, the function first created a temporary folder and copied the engine files (.ae + .vdf).

Starting with v1.3 of the SAVAPI library (SAVAPI v3.3 release - November 2011) this function no longer creates temporary directories on Windows. It will unload the old engine instance and will load it again from the same directory.

On both platform types (UNIX and Windows), the function `SAVAPI3_reload_engine()` performs the following tasks:

- Uninitializes the old engine
  If there is any instance still used, then it returns an error (*E_BUSY*) otherwise it uninitializes the engine
- Loads the new engine

Note
If any error occurs during the re-initialization, the function will return an error. In this case, SAVAPI can no longer continue to scan. This is why we recommend to always use the Avira Updater which tests the engine files before installing them in the final directory.

The reasons for no longer having two engines loaded simultaneously in memory are (sorted by priority):

1. The OS will not load twice a library with the same name and which resides in the same directory. Instead of loading again, it returns the old handle of the library already loaded in memory. This was actually the reason for which SAVAPI was copying the engine and the signatures in temporary directories. In order to stop copying these files, you need to unload the old engine first and then load the new one.
2. Engine load time - Currently the engine needs between 3-10 seconds (depending on hardware and OS) to load its data and the virus signatures in memory.
3. Engine size - Currently the engine needs about 80MB to load its data and the virus signatures in memory.

2. Updater

2.1 Compatibility issues

The Updater binary on Windows is a fully conformant console application - it doesn't daemonize anymore. Until this release, the Updater was decoupling itself from the console and was working as a service. From now on the Updater will decouple from the console only when it ends itself.

2.2 Improvements and fixes

Following problems were fixed:

- Sometimes updater hangs for some time
- Complete update takes about 2 min instead of couple of sec in win 7 x64
- The Updater digest authentication doesn't work

- "connect-timeout" parameter has no effect
- Incorrect behavior when a wrong module name or no module name is specified
- avupdate shouldn't require master-file if skip-master-file is specified
- Although the --quiet option is present, messages are still logged on the terminal
- A compute download progress problem that occurred only when the computer was suspended during update

Improvements:

- Added cache for Web Proxy Auto-Discovery (WPAD) protocol
- Added timeout for WPAD protocol
- Added new messages in *UpdateLibMsg.xml*