

SAVAPI

Application Programming Interface reference for SAVAPI 4 library

API version 5.5

Wednesday, 1st of March, 2023

1 SAVAPI	1
2 Todo List	3
3 Module Index	5
3.1 Modules	5
4 Data Structure Index	7
4.1 Data Structures	7
5 File Index	9
5.1 File List	9
6 Module Documentation	11
6.1 SAVAPI hex2bin function pointers	11
6.1.1 Detailed Description	11
6.1.2 Typedef Documentation	11
6.1.2.1 bin2hex_t	11
6.1.2.2 hex2bin_t	11
6.2 SAVAPI constants	12
6.2.1 Detailed Description	12
6.2.2 Macro Definition Documentation	12
6.2.2.1 SAVAPI_SYMBOL	12
6.2.2.2 STR	12
6.3 API version	13
6.3.1 Detailed Description	13
6.3.2 Macro Definition Documentation	13
6.3.2.1 SAVAPI_API_MAJOR_VERSION	13
6.3.2.2 SAVAPI_API_MINOR_VERSION	13
6.4 Error or information categories	14
6.4.1 Detailed Description	14
6.4.2 Macro Definition Documentation	14
6.4.2.1 SAVAPI_ECAT_APC_REPORT_TTL	14
6.4.2.2 SAVAPI_ECAT_ERROR_GENERIC	14
6.4.2.3 SAVAPI_ECAT_ERROR_IO	14
6.4.2.4 SAVAPI_ECAT_ERROR_SCAN	14
6.4.2.5 SAVAPI_ECAT_ERROR_UNPACK	15
6.5 Error levels	15
6.5.1 Detailed Description	15
6.5.2 Macro Definition Documentation	15
6.5.2.1 SAVAPI_ELEVEL_ERROR	15
6.5.2.2 SAVAPI_ELEVEL_INFO	15
6.5.2.3 SAVAPI_ELEVEL_WARNING	15
6.6 Initialization flags	16

6.6.1 Detailed Description	16
6.6.2 Macro Definition Documentation	16
6.6.2.1 SAVAPI_FLAG_USE_LOCAL_SOCKET	16
6.6.2.2 SAVAPI_FLAG_USE_TCP	16
6.7 Scan warnings	16
6.7.1 Detailed Description	16
6.7.2 Macro Definition Documentation	17
6.7.2.1 SAVAPI_W_DAMAGED	17
6.7.2.2 SAVAPI_W_HEADER_MALFORMED	17
6.7.2.3 SAVAPI_W_MAX_EXTRACTED	17
6.7.2.4 SAVAPI_W_OLE_DAMAGED	17
6.7.2.5 SAVAPI_W_POTENTIAL_ARCH_BOMB	17
6.7.2.6 SAVAPI_W_PROGRESS_ABORT	17
6.7.2.7 SAVAPI_W_RATIO_EXCEEDED	17
6.7.2.8 SAVAPI_W_SUSPICIOUS	18
6.8 Iframes informations	18
6.8.1 Detailed Description	18
6.8.2 Macro Definition Documentation	18
6.8.2.1 SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL	18
6.8.2.2 SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE	18
6.8.2.3 SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS	18
6.8.2.4 SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS	19
6.9 Scan information	19
6.9.1 Detailed Description	19
6.9.2 Macro Definition Documentation	19
6.9.2.1 SAVAPI_I_ACTIVE_CONTENT_PRESENT	19
6.9.2.2 SAVAPI_I_ALL_MACROS_DELETED	19
6.9.2.3 SAVAPI_I_APC_RESULT_EXPIRY	19
6.9.2.4 SAVAPI_I_APC_SCAN_DURATION	20
6.9.2.5 SAVAPI_I_MACRO_AUTOSTART	20
6.9.2.6 SAVAPI_I_MACROS_PRESENT	20
6.9.2.7 SAVAPI_I_MAILBOX	20
6.9.2.8 SAVAPI_I_OLE_ENCRYPTED	20
6.9.2.9 SAVAPI_I_OLEFILE	20
6.9.2.10 SAVAPI_I_TEMPLATE	20
6.10 SAVAPI options	21
6.10.1 Detailed Description	21
6.10.2 Typedef Documentation	22
6.10.2.1 SAVAPI_OPTION	22
6.10.3 Enumeration Type Documentation	22
6.10.3.1 SAVAPI_option	22
6.11 SAVAPI global options	33

6.11.1 Detailed Description	33
6.11.2 Typedef Documentation	33
6.11.2.1 SAVAPI_GLOBAL_OPTION	33
6.11.3 Enumeration Type Documentation	33
6.11.3.1 SAVAPI_global_option	33
6.12 SAVAPI OnAccess result	38
6.12.1 Detailed Description	38
6.12.2 Enumeration Type Documentation	38
6.12.2.1 SAVAPI_OA_SCAN_RESULT	38
6.13 SAVAPI APC scan callback return values	39
6.13.1 Detailed Description	39
6.13.2 Enumeration Type Documentation	39
6.13.2.1 SAVAPI_APC_SCAN_RESULT	39
6.14 Callbacks' ids	39
6.14.1 Detailed Description	40
6.14.2 Macro Definition Documentation	40
6.14.2.1 SAVAPI_CALLBACK_APC_SCAN	40
6.14.2.2 SAVAPI_CALLBACK_ARCHIVE_OPEN	41
6.14.2.3 SAVAPI_CALLBACK_CONTENT_REPORT	41
6.14.2.4 SAVAPI_CALLBACK_OA_FILE_RESULT	41
6.14.2.5 SAVAPI_CALLBACK_PRE_SCAN	41
6.14.2.6 SAVAPI_CALLBACK_PROGRESS_REPORT	42
6.14.2.7 SAVAPI_CALLBACK_REPORT_ERROR	42
6.14.2.8 SAVAPI_CALLBACK_REPORT_FILE_STATUS	42
6.14.2.9 SAVAPI_CALLBACK_SCAN_DETAILS_REPORT	42
6.15 SAVAPI report scan details types	42
6.15.1 Detailed Description	42
6.15.2 Macro Definition Documentation	42
6.15.2.1 SAVAPI_REPORT_ALERTURL	43
6.15.2.2 SAVAPI_REPORT_REPAIRABLE	43
6.16 SAVAPI report content types	43
6.16.1 Detailed Description	43
6.16.2 Macro Definition Documentation	43
6.16.2.1 SAVAPI_REPORT_CONTENT_IFRAME	43
6.17 SAVAPI signals	43
6.17.1 Detailed Description	43
6.17.2 Macro Definition Documentation	43
6.17.2.1 SAVAPI_SIGNAL_SCAN_ABORT	44
6.18 SAVAPI scan statuses	44
6.18.1 Detailed Description	44
6.18.2 Macro Definition Documentation	44
6.18.2.1 SAVAPI_SCAN_STATUS_CLEAN	44

6.18.2.2 SAVAPI_SCAN_STATUS_ERROR	44
6.18.2.3 SAVAPI_SCAN_STATUS_FINISHED	45
6.18.2.4 SAVAPI_SCAN_STATUS_INFECTED	45
6.18.2.5 SAVAPI_SCAN_STATUS_SUSPICIOUS	45
6.19 File types	45
6.19.1 Detailed Description	45
6.19.2 Macro Definition Documentation	45
6.19.2.1 SAVAPI_FTYPE_ARCHIVE	45
6.19.2.2 SAVAPI_FTYPE_IN_ARCHIVE	46
6.19.2.3 SAVAPI_FTYPE_REGULAR	46
6.20 Filename flags	46
6.20.1 Detailed Description	46
6.20.2 Macro Definition Documentation	46
6.20.2.1 SAVAPI_FLAG_LAST_FILENAME_DEFAULT	46
6.21 SAVAPI structures	47
6.21.1 Detailed Description	49
6.21.2 Typedef Documentation	49
6.21.2.1 SAVAPI_ALERT_URL_DATA	50
6.21.2.2 SAVAPI_APC_GLOBAL_INIT	50
6.21.2.3 SAVAPI_APC_SCAN_DATA	50
6.21.2.4 SAVAPI_APC_SCAN_MODE	50
6.21.2.5 SAVAPI_ARCHIVE_OPEN_DATA	50
6.21.2.6 SAVAPI_CALLBACK_DATA	50
6.21.2.7 SAVAPI_COMMAND_DATA	51
6.21.2.8 SAVAPI_ENGINE_MODULE_TYPE	51
6.21.2.9 SAVAPI_ERROR_DATA	51
6.21.2.10 SAVAPI_FILE_INFO	51
6.21.2.11 SAVAPI_FILE_STATUS_DATA	51
6.21.2.12 SAVAPI_GLOBAL_INIT	52
6.21.2.13 SAVAPI_IFRAME_URL_DATA	52
6.21.2.14 SAVAPI_INSTANCE_INIT	52
6.21.2.15 SAVAPI_KEY_VALUE	52
6.21.2.16 SAVAPI_LOG_LEVEL	52
6.21.2.17 SAVAPI_MALWARE_INFO	53
6.21.2.18 SAVAPI_OA_FILE_RESULT_DATA	53
6.21.2.19 SAVAPI_OA_GLOBAL_INIT	53
6.21.2.20 SAVAPI_PRESCAN_DATA	53
6.21.2.21 SAVAPI_REPAIRABLE_DATA	53
6.21.2.22 SAVAPI_REPORT_CONTENT_DATA	54
6.21.2.23 SAVAPI_REPORT_PROGRESS_DATA	54
6.21.2.24 SAVAPI_REPORT_SCAN_DETAILS_DATA	54
6.21.2.25 SAVAPI_SIGNAL_DATA	54

6.21.2.26 SAVAPI_SIMPLE_SCAN_FILE_DATA	54
6.21.2.27 SAVAPI_SIMPLE_SCAN_OUTPUT	55
6.21.2.28 SAVAPI_SIMPLE_SCAN_STATISTICS	55
6.21.2.29 SAVAPI_VERSION	55
6.21.3 Enumeration Type Documentation	55
6.21.3.1 _SAVAPI_log_level	55
6.21.3.2 SAVAPI_APC_SCAN_ANSWER	56
6.21.3.3 SAVAPI_APC_scan_mode	56
6.21.3.4 SAVAPI_APC_SCAN_STAGE	56
6.21.3.5 SAVAPI_engine_module_type	56
6.22 SAVAPI typedefs	57
6.22.1 Detailed Description	57
6.22.2 Typedef Documentation	57
6.22.2.1 APC_set_report_info_t	57
6.22.2.2 SAVAPI_CALLBACK	58
6.22.2.3 SAVAPI_ENGINE_MODULE_CALLBACK	58
6.22.2.4 SAVAPI_FD	58
6.22.2.5 SAVAPI_LOG_CALLBACK	58
6.22.2.6 SAVAPI_OA_INSTANCE_CALLBACK	59
6.23 SAVAPI function pointers	59
6.23.1 Detailed Description	59
6.24 SAVAPI main function pointers	59
6.24.1 Detailed Description	60
6.24.2 Typedef Documentation	61
6.24.2.1 SAVAPI_APC_get_version_t	61
6.24.2.2 SAVAPI_APC_initialize_t	61
6.24.2.3 SAVAPI_APC_uninitialize_t	61
6.24.2.4 SAVAPI_create_instance_t	61
6.24.2.5 SAVAPI_engine_modules_get_t	61
6.24.2.6 SAVAPI_engine_versions_get_t	61
6.24.2.7 SAVAPI_extract_malware_names_t	62
6.24.2.8 SAVAPI_free_t	62
6.24.2.9 SAVAPI_get_fops_t	62
6.24.2.10 SAVAPI_get_t	62
6.24.2.11 SAVAPI_get_user_data_t	62
6.24.2.12 SAVAPI_get_version_t	62
6.24.2.13 SAVAPI_global_set_t	62
6.24.2.14 SAVAPI_initialize_t	63
6.24.2.15 SAVAPI_is_running_ex_t	63
6.24.2.16 SAVAPI_OA_create_instances_t	63
6.24.2.17 SAVAPI_OA_initialize_t	63
6.24.2.18 SAVAPI_OA_start_scan_t	63

6.24.2.19 SAVAPI_OA_stop_scan_t	63
6.24.2.20 SAVAPI_OA_uninitialize_t	63
6.24.2.21 SAVAPI_register_callback_t	64
6.24.2.22 SAVAPI_release_instance_t	64
6.24.2.23 SAVAPI_reload_engine_ex_t	64
6.24.2.24 SAVAPI_scan_t	64
6.24.2.25 SAVAPI_send_signal_t	64
6.24.2.26 SAVAPI_set_fops_t	64
6.24.2.27 SAVAPI_set_log_callback_t	64
6.24.2.28 SAVAPI_set_quickload_init_t	65
6.24.2.29 SAVAPI_set_t	65
6.24.2.30 SAVAPI_set_user_data_t	65
6.24.2.31 SAVAPI_simple_scan_t	65
6.24.2.32 SAVAPI_uninitialize_t	65
6.24.2.33 SAVAPI_unregister_callback_t	65
6.25 SAVAPI functions	66
6.25.1 Detailed Description	68
6.25.2 Function Documentation	68
6.25.2.1 SAVAPI_APC_get_version()	68
6.25.2.2 SAVAPI_APC_initialize()	68
6.25.2.3 SAVAPI_APC_uninitialize()	69
6.25.2.4 SAVAPI_create_instance()	69
6.25.2.5 SAVAPI_engine_modules_get()	70
6.25.2.6 SAVAPI_engine_versions_get()	70
6.25.2.7 SAVAPI_extract_malware_names()	71
6.25.2.8 SAVAPI_FPC_disable_preinit()	71
6.25.2.9 SAVAPI_free()	72
6.25.2.10 SAVAPI_get()	72
6.25.2.11 SAVAPI_get_dynamic_detect()	73
6.25.2.12 SAVAPI_get_fops()	73
6.25.2.13 SAVAPI_get_user_data()	74
6.25.2.14 SAVAPI_get_version()	74
6.25.2.15 SAVAPI_global_set()	74
6.25.2.16 SAVAPI_initialize()	75
6.25.2.17 SAVAPI_is_running()	75
6.25.2.18 SAVAPI_is_running_ex()	76
6.25.2.19 SAVAPI_OA_create_instances()	76
6.25.2.20 SAVAPI_OA_initialize()	77
6.25.2.21 SAVAPI_OA_start_scan()	78
6.25.2.22 SAVAPI_OA_stop_scan()	78
6.25.2.23 SAVAPI_OA_uninitialize()	78
6.25.2.24 SAVAPI_register_callback()	78

6.25.2.25 SAVAPI_release_instance()	79
6.25.2.26 SAVAPI_reload_engine()	80
6.25.2.27 SAVAPI_reload_engine_ex()	80
6.25.2.28 SAVAPI_scan()	81
6.25.2.29 SAVAPI_send_signal()	82
6.25.2.30 SAVAPI_set()	82
6.25.2.31 SAVAPI_set_fops()	83
6.25.2.32 SAVAPI_set_log_callback()	83
6.25.2.33 SAVAPI_set_quickload_init()	84
6.25.2.34 SAVAPI_set_user_data()	84
6.25.2.35 SAVAPI_simple_scan()	85
6.25.2.36 SAVAPI_uninitialize()	86
6.25.2.37 SAVAPI_unregister_callback()	86
6.26 SAVAPI return codes	87
6.26.1 Detailed Description	88
6.26.2 Typedef Documentation	88
6.26.2.1 SAVAPI_STATUS	88
6.26.3 Enumeration Type Documentation	88
6.26.3.1 SAVAPI_status	88
6.27 Plugin defines	101
6.27.1 Detailed Description	101
6.27.2 Macro Definition Documentation	101
6.27.2.1 _T	101
6.27.2.2 SAVAPI_PLG_EXP	102
6.27.2.3 SAVAPI_PLG_MAX_STR	102
6.27.2.4 SAVAPI_PLG_MAX_VER_STR	102
6.27.2.5 SAVAPI_PLG_tchar_t	102
6.27.2.6 SAVAPI_PLG_VER_MAJ	102
6.27.2.7 SAVAPI_PLG_VER_MIN	102
6.28 Plugin return statuses	102
6.28.1 Detailed Description	103
6.28.2 Macro Definition Documentation	103
6.28.2.1 SAVAPI_EPLG_INIT	103
6.28.2.2 SAVAPI_EPLG_INTERNAL	103
6.28.2.3 SAVAPI_EPLG_INVALID	103
6.28.2.4 SAVAPI_EPLG_NO_MEM	103
6.28.2.5 SAVAPI_EPLG_SUCCESS	103
6.28.2.6 SAVAPI_EPLG_UKNW_INTERFACE	103
6.29 Plugin's specific typedefs	104
6.29.1 Detailed Description	104
6.29.2 Typedef Documentation	104
6.29.2.1 SAVAPI_PLG_inst_options_t	104

6.29.2.2 SAVAPI_PLG_instance_t	104
6.29.2.3 SAVAPI_PLG_options_t	104
6.30 Plugin's general typedefs	104
6.30.1 Detailed Description	105
6.30.2 Typedef Documentation	105
6.30.2.1 SAVAPI_PLG_init_t	105
6.30.2.2 SAVAPI_PLG_instance_create_t	105
6.30.2.3 SAVAPI_PLG_instance_release_t	105
6.30.2.4 SAVAPI_PLG_status_t	105
6.30.2.5 SAVAPI_PLG_uninit_t	105
6.31 Plugin's structures definitions	106
6.31.1 Detailed Description	106
6.31.2 Macro Definition Documentation	106
6.31.2.1 SAVAPI_PLG_MAIN_FUNC	106
6.31.3 Typedef Documentation	106
6.31.3.1 SAVAPI_PLG_info_t	106
6.31.3.2 SAVAPI_PLG_main_t	107
6.31.4 Function Documentation	107
6.31.4.1 SAVAPI_PLG_MAIN_FUNC()	107
6.32 types for a SAVAPI FOPS plugin	107
6.32.1 Detailed Description	107
6.32.2 Macro Definition Documentation	107
6.32.2.1 SAVAPI_PLG_AVE_FOPS	107
6.32.3 Typedef Documentation	108
6.32.3.1 SAVAPI_PLG_fops_inst_options_t	108
6.32.3.2 SAVAPI_PLG_fops_instance_t	108
6.32.3.3 SAVAPI_PLG_fops_options_t	108
6.33 SAVAPI STCHAR function pointers	108
6.33.1 Detailed Description	108
6.33.2 Typedef Documentation	108
6.33.2.1 CharToSTCHAR_t	108
6.33.2.2 STCHARToChar_t	108
7 Data Structure Documentation	109
7.1 _AVE_STRUCT_FILE_OPERATIONS Struct Reference	109
7.1.1 Detailed Description	110
7.1.2 Field Documentation	110
7.1.2.1 fops_access	110
7.1.2.2 fops_close	111
7.1.2.3 fops_flush	111
7.1.2.4 fops_free	111
7.1.2.5 fops_get_last_error	112

7.1.2.6 fops_getc	112
7.1.2.7 fops_getfattr	113
7.1.2.8 fops_getfsize	113
7.1.2.9 fops_gets	113
7.1.2.10 fops_malloc	114
7.1.2.11 fops_open	114
7.1.2.12 fops_putc	115
7.1.2.13 fops_puts	115
7.1.2.14 fops_read	116
7.1.2.15 fops_rename	116
7.1.2.16 fops_seek	117
7.1.2.17 fops_setfattr	117
7.1.2.18 fops_tell	118
7.1.2.19 fops_ungetc	118
7.1.2.20 fops_unlink	119
7.1.2.21 fops_write	119
7.2 SAVAPI_report_content_data::_content_data Union Reference	120
7.2.1 Detailed Description	120
7.2.2 Field Documentation	120
7.2.2.1 iframeurl_data	120
7.3 _SAVAPI_PLG_fops_instance_t Struct Reference	120
7.3.1 Detailed Description	121
7.3.2 Field Documentation	121
7.3.2.1 fops	121
7.4 _SAVAPI_PLG_info_t Struct Reference	121
7.4.1 Detailed Description	121
7.4.2 Field Documentation	121
7.4.2.1 init	122
7.4.2.2 instance_create	122
7.4.2.3 instance_release	122
7.4.2.4 interface_ver_maj	122
7.4.2.5 interface_ver_min	122
7.4.2.6 name	122
7.4.2.7 res	122
7.4.2.8 uninit	123
7.4.2.9 version	123
7.5 SAVAPI_report_scan_details_data::_scan_details_data Union Reference	123
7.5.1 Detailed Description	123
7.5.2 Field Documentation	123
7.5.2.1 alert_url_data	123
7.5.2.2 repairable_data	124
7.6 SAVAPI_alert_url_data Struct Reference	124

7.6.1 Detailed Description	124
7.6.2 Field Documentation	124
7.6.2.1 alert_url	124
7.6.2.2 file_info	124
7.7 SAVAPI_APC_global_init Struct Reference	125
7.7.1 Detailed Description	125
7.7.2 Field Documentation	125
7.7.2.1 apc_mode	125
7.7.2.2 blackout_retries	125
7.7.2.3 blackout_timeout	126
7.7.2.4 cache_file_path	126
7.7.2.5 cache_size	126
7.7.2.6 cert_dir	126
7.7.2.7 dump_cache_file	127
7.7.2.8 lib_dir	127
7.7.2.9 proxy	127
7.7.2.10 temp_dir	127
7.8 SAVAPI_APC_REPORT_DATA Struct Reference	128
7.8.1 Detailed Description	128
7.8.2 Field Documentation	128
7.8.2.1 malware_info	128
7.8.2.2 scan_answer	128
7.8.2.3 store_cache	128
7.8.2.4 ttl	129
7.9 SAVAPI_apc_scan_data Struct Reference	129
7.9.1 Detailed Description	129
7.9.2 Field Documentation	129
7.9.2.1 file_info	129
7.9.2.2 flags	130
7.9.2.3 fops_context	130
7.9.2.4 fops_handle	130
7.9.2.5 hash	130
7.9.2.6 risk_rating_level	130
7.9.2.7 savapi_fd	131
7.9.2.8 set_report_info	131
7.9.2.9 stage	131
7.10 SAVAPI_archive_open_data Struct Reference	131
7.10.1 Detailed Description	131
7.10.2 Field Documentation	132
7.10.2.1 file_info	132
7.10.2.2 flags	132
7.11 SAVAPI_callback_data Struct Reference	132

7.11.1 Detailed Description	133
7.11.2 Field Documentation	133
7.11.2.1 callback_data	133
7.11.2.2 flags	133
7.11.2.3 type	133
7.11.2.4 user_data	133
7.11.2.5 version	133
7.12 SAVAPI_command_data Struct Reference	134
7.12.1 Detailed Description	134
7.12.2 Field Documentation	134
7.12.2.1 command_data	134
7.12.2.2 signal_id	134
7.13 SAVAPI_error_data Struct Reference	135
7.13.1 Detailed Description	135
7.13.2 Field Documentation	135
7.13.2.1 category	135
7.13.2.2 code	135
7.13.2.3 file_info	136
7.13.2.4 level	136
7.13.2.5 options	136
7.14 SAVAPI_file_info Struct Reference	136
7.14.1 Detailed Description	136
7.14.2 Field Documentation	136
7.14.2.1 level	137
7.14.2.2 name	137
7.14.2.3 type	137
7.15 SAVAPI_file_status_data Struct Reference	137
7.15.1 Detailed Description	137
7.15.2 Field Documentation	138
7.15.2.1 file_info	138
7.15.2.2 flags	138
7.15.2.3 info	138
7.15.2.4 malware_info	138
7.15.2.5 scan_answer	138
7.15.2.6 warning	139
7.16 SAVAPI_global_init Struct Reference	139
7.16.1 Detailed Description	139
7.16.2 Field Documentation	139
7.16.2.1 api_major_version	139
7.16.2.2 api_minor_version	139
7.16.2.3 avll_dirpath	140
7.16.2.4 engine_dirpath	140

7.16.2.5 key_file_name	140
7.16.2.6 program_type	140
7.16.2.7 vdfs_dirpath	140
7.17 SAVAPI_iframe_url_data Struct Reference	141
7.17.1 Detailed Description	141
7.17.2 Field Documentation	141
7.17.2.1 attribute	141
7.17.2.2 url	141
7.18 SAVAPI_instance_init Struct Reference	141
7.18.1 Detailed Description	142
7.18.2 Field Documentation	142
7.18.2.1 connection_timeout	142
7.18.2.2 flags	142
7.18.2.3 get_timeout	142
7.18.2.4 host_name	143
7.18.2.5 password	143
7.18.2.6 port	143
7.18.2.7 scan_timeout	143
7.18.2.8 set_timeout	144
7.18.2.9 username	144
7.19 SAVAPI_key_value Struct Reference	144
7.19.1 Detailed Description	144
7.19.2 Field Documentation	145
7.19.2.1 id	145
7.19.2.2 type	145
7.19.2.3 value	145
7.20 SAVAPI_malware_info Struct Reference	145
7.20.1 Detailed Description	145
7.20.2 Field Documentation	146
7.20.2.1 app_flags	146
7.20.2.2 message	146
7.20.2.3 name	146
7.20.2.4 removable	146
7.20.2.5 strict	146
7.20.2.6 type	146
7.21 SAVAPI_OA_file_result_data Struct Reference	147
7.21.1 Detailed Description	147
7.21.2 Field Documentation	147
7.21.2.1 filename	147
7.21.2.2 pid	147
7.21.2.3 sid	147
7.21.2.4 sid_len	148

7.22 SAVAPI_OA_global_init Struct Reference	148
7.22.1 Detailed Description	148
7.22.2 Field Documentation	148
7.22.2.1 scm_pending_time	148
7.22.2.2 threads_number	149
7.23 SAVAPI_pre_scan_data Struct Reference	149
7.23.1 Detailed Description	149
7.23.2 Field Documentation	149
7.23.2.1 file_info	149
7.23.2.2 flags	150
7.24 SAVAPI_repairable_data Struct Reference	150
7.24.1 Detailed Description	150
7.24.2 Field Documentation	150
7.24.2.1 file_info	150
7.24.2.2 malware_info	150
7.25 SAVAPI_report_content_data Struct Reference	151
7.25.1 Detailed Description	151
7.25.2 Field Documentation	151
7.25.2.1 content_data	151
7.25.2.2 file_info	151
7.25.2.3 flags	151
7.25.2.4 type	152
7.26 SAVAPI_report_progress_data Struct Reference	152
7.26.1 Detailed Description	152
7.26.2 Field Documentation	152
7.26.2.1 flags	152
7.26.2.2 message	152
7.27 SAVAPI_report_scan_details_data Struct Reference	153
7.27.1 Detailed Description	153
7.27.2 Field Documentation	153
7.27.2.1 flags	153
7.27.2.2 scan_details_data	153
7.27.2.3 type	153
7.28 SAVAPI_signal_data Struct Reference	154
7.28.1 Detailed Description	154
7.28.2 Field Documentation	154
7.28.2.1 signal_data	154
7.28.2.2 signal_id	154
7.29 SAVAPI_simple_scan_file_data Struct Reference	155
7.29.1 Detailed Description	155
7.29.2 Field Documentation	155
7.29.2.1 error_code	155

7.29.2.2 error_level	155
7.29.2.3 malware_name	156
7.29.2.4 malware_type	156
7.29.2.5 name	156
7.29.2.6 scan_answer	156
7.30 SAVAPI_simple_scan_output Struct Reference	156
7.30.1 Detailed Description	156
7.30.2 Field Documentation	157
7.30.2.1 count	157
7.30.2.2 files	157
7.30.2.3 stats	157
7.31 SAVAPI_simple_scan_statistics Struct Reference	157
7.31.1 Detailed Description	157
7.31.2 Field Documentation	158
7.31.2.1 errors	158
7.31.2.2 infections	158
7.31.2.3 suspicions	158
7.31.2.4 total_files	158
7.32 SAVAPI_version Struct Reference	158
7.32.1 Detailed Description	159
7.32.2 Field Documentation	159
7.32.2.1 build_major	159
7.32.2.2 build_minor	159
7.32.2.3 major	159
7.32.2.4 minor	159
7.33 SAVAPI_callback_data::specific_data Union Reference	160
7.33.1 Detailed Description	160
7.33.2 Field Documentation	160
7.33.2.1 apc_scan_data	160
7.33.2.2 archive_open_data	160
7.33.2.3 error_data	160
7.33.2.4 file_status_data	161
7.33.2.5 oa_file_result_data	161
7.33.2.6 pre_scan_data	161
7.33.2.7 private_data	161
7.33.2.8 report_content_data	161
7.33.2.9 report_progress_data	161
7.33.2.10 report_scan_details_data	161
8 File Documentation	163
8.1 asc_bin.h File Reference	163
8.1.1 Function Documentation	163

8.1.1.1 asc2bin()	163
8.1.1.2 bin2asc()	164
8.1.1.3 bin2hex()	164
8.1.1.4 hex2bin()	165
8.2 asc_bin.h	166
8.3 fops.h File Reference	166
8.3.1 Detailed Description	167
8.3.2 Macro Definition Documentation	168
8.3.2.1 FOPS_ERROR	168
8.3.2.2 FOPS_INVALID_HANDLE	168
8.3.2.3 OPEN_CR	169
8.3.2.4 OPEN_RO	169
8.3.2.5 OPEN_RW	169
8.3.2.6 SEEK_CUR	169
8.3.2.7 SEEK_END	169
8.3.2.8 SEEK_SET	169
8.3.3 Typedef Documentation	169
8.3.3.1 _fattr_t	169
8.3.3.2 _fpos_t	170
8.3.3.3 AVE_FOPS	170
8.3.3.4 FOPS_HANDLE	170
8.3.4 Function Documentation	170
8.3.4.1 check_free_mem()	170
8.3.4.2 e_tempname()	170
8.3.5 Variable Documentation	170
8.3.5.1 _user_fops	170
8.3.5.2 f_check_mem	171
8.4 fops.h	171
8.5 fopstypes.h File Reference	172
8.5.1 Macro Definition Documentation	173
8.5.1.1 _cdecl	173
8.5.2 Typedef Documentation	173
8.5.2.1 INT16	173
8.5.2.2 INT32	173
8.5.2.3 INT64	173
8.5.2.4 INT8	173
8.5.2.5 UINT16	174
8.5.2.6 UINT32	174
8.5.2.7 UINT64	174
8.5.2.8 UINT8	174
8.6 fopstypes.h	174
8.7 savapi.h File Reference	175

8.8 savapi.h	183
8.9 savapi_errors.h File Reference	191
8.10 savapi_errors.h	193
8.11 savapi_plg.h File Reference	194
8.12 savapi_plg.h	195
8.13 savapi_plg_fops.h File Reference	196
8.14 savapi_plg_fops.h	197
8.15 stchar.h File Reference	197
8.15.1 Detailed Description	197
8.15.2 Macro Definition Documentation	198
8.15.2.1 SAVAPI_EXP	198
8.15.2.2 SAVAPI_SIZE_T	198
8.15.2.3 SAVAPI_TCHAR	198
8.15.3 Function Documentation	198
8.15.3.1 CharToSTCHAR()	198
8.15.3.2 STCHARToChar()	199
8.16 stchar.h	199
Index	201

Chapter 1

SAVAPI

SAVAPI stands for Secure AntiVirus Application Programming Interface. Its main purpose is to offer a very simple scanning interface for clients who want to programmatically integrate scanning services into their applications.

This document explains how to use the SAVAPI interface shared libraries written and provided by Avira Operations GmbH & Co. KG. The provided LICENSE file explains the terms and conditions for using the SAVAPI interface shared libraries.

By utilizing the implemented SAVAPI interface, it becomes very simple to quickly integrate SAVAPI into your applications. There is no need to be concerned about protocol details, since they have all been implemented by Avira Operations GmbH & Co. KG.

See: [SAVAPI options](#), [SAVAPI constants](#), [SAVAPI defines](#), [SAVAPI structures](#), [SAVAPI functions](#)

Chapter 2

Todo List

Global [SAVAPI_command_data::command_data](#)

Add specific data as soon as new signals, which require data will be defined.

Global [SAVAPI_signal_data::signal_data](#)

Add specific data as soon as new signals, which require data will be defined.

Global [SAVAPI_SIGNAL_SCAN_ABORT](#)

Add new signals as needed.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

SAVAPI constants	12
API version	13
Error or information categories	14
Error levels	15
Initialization flags	16
Scan warnings	16
Iframes informations	18
Scan information	19
Callbacks' ids	39
SAVAPI report scan details types	42
SAVAPI report content types	43
SAVAPI signals	43
SAVAPI scan statuses	44
File types	45
Filename flags	46
SAVAPI options	21
SAVAPI global options	33
SAVAPI OnAccess result	38
SAVAPI APC scan callback return values	39
SAVAPI structures	47
SAVAPI typedefs	57
SAVAPI function pointers	59
SAVAPI hex2bin function pointers	11
SAVAPI main function pointers	59
SAVAPI STCHAR function pointers	108
SAVAPI functions	66
SAVAPI return codes	87
Plugin defines	101
Plugin return statuses	102
Plugin's specific typedefs	104
Plugin's general typedefs	104
Plugin's structures definitions	106
types for a SAVAPI FOPS plugin	107

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

_AVE_STRUCT_FILE_OPERATIONS	
The File OPerationS (FOPS) structure. Here are defined all the function prototypes which must be implemented in order to allow the scanning engine to process a special type of object . . .	109
SAVAPI_report_content_data::_content_data	120
_SAVAPI_PLG_fops_instance_t	120
_SAVAPI_PLG_info_t	
Structure containing the plugin's information (name, version etc.)	121
SAVAPI_report_scan_details_data::_scan_details_data	123
SAVAPI_alert_url_data	
Structure associated with the ALERTURL report	124
SAVAPI_APC_global_init	
The structure used for initializing APC	125
SAVAPI_APC_REPORT_DATA	
Information to report when calling a APC_set_report_info_t function	128
SAVAPI_apc_scan_data	
The structure associated with the SAVAPI_CALLBACK_APC_SCAN callback	129
SAVAPI_archive_open_data	
Contains the data sent to a archive_open callback	131
SAVAPI_callback_data	
Structure passed by SAVAPI to a user defined callback, containing all the necessary data . . .	132
SAVAPI_command_data	
The structure to be passed when sending a command	134
SAVAPI_error_data	
The structure associated with report error callback	135
SAVAPI_file_info	
Contains data about the scanned file	136
SAVAPI_file_status_data	
Contains the data sent to a report file status callback	137
SAVAPI_global_init	
The structure used at SAVAPI initialization	139
SAVAPI_iframe_url_data	
Structure associated with the iframe report	141
SAVAPI_instance_init	
The structure used at SAVAPI instance creation	141
SAVAPI_key_value	
Generic container	144

SAVAPI_malware_info	Contains data about the found malware in an infected/suspicious file	145
SAVAPI_OA_file_result_data	Contains the data sent to an OnAccess file status callback	147
SAVAPI_OA_global_init	The structure used for initializing OnAccess	148
SAVAPI_pre_scan_data	Contains the data sent to a prescan callback	149
SAVAPI_repairable_data	Structure associated with the REPAIRABLE report	150
SAVAPI_report_content_data	The structure associated with report content callback	151
SAVAPI_report_progress_data	The structure associated with report progress callback	152
SAVAPI_report_scan_details_data	The structure associated with report scan details callback	153
SAVAPI_signal_data	The structure to be passed when sending a signal	154
SAVAPI_simple_scan_file_data	The structure contains information about each infected, suspicious, or erroneous file scanned by	
SAVAPI_simple_scan	155
SAVAPI_simple_scan_output	The structure containing the output for SAVAPI_simple_scan	156
SAVAPI_simple_scan_statistics	The structure contains statistics for the simple scan	157
SAVAPI_version	The structure used to retrieve SAVAPI version	158
SAVAPI_callback_data::specific_data	Callbacks specific data	160

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

asc_bin.h	163
fops.h	
This file defines the interface to allow the engine to scan any type of object	166
fopstypes.h	172
savapi.h	175
savapi_errors.h	191
savapi_plg.h	194
savapi_plg_fops.h	196
stchar.h	
Conversion between SAVAPI_TCHAR and char/TCHAR	197

Chapter 6

Module Documentation

6.1 SAVAPI hex2bin function pointers

Handle types for exported SAVAPI hex2bin functions.

Typedefs

- typedef int(* [bin2hex_t](#)) (const char *binblock, [SAVAPI_SIZE_T](#) binblock_size, char *hexblock, [SAVAPI_SIZE_T](#) *hexblock_size)
- typedef int(* [hex2bin_t](#)) (const char *hexblock, [SAVAPI_SIZE_T](#) hexblock_size, char *binblock, [SAVAPI_SIZE_T](#) *binblock_size)

6.1.1 Detailed Description

Handle types for exported SAVAPI hex2bin functions.

6.1.2 Typedef Documentation

6.1.2.1 bin2hex_t

```
typedef int(* bin2hex_t) (const char *binblock, SAVAPI\_SIZE\_T binblock_size, char *hexblock,  
SAVAPI\_SIZE\_T *hexblock_size)
```

6.1.2.2 hex2bin_t

```
typedef int(* hex2bin_t) (const char *hexblock, SAVAPI\_SIZE\_T hexblock_size, char *binblock,  
SAVAPI\_SIZE\_T *binblock_size)
```

6.2 SAVAPI constants

Modules

- [API version](#)
- [Error or information categories](#)
- [Error levels](#)
- [Initialization flags](#)
- [Scan warnings](#)
- [Iframes informations](#)
- [Scan information](#)
- [Callbacks' ids](#)
- [SAVAPI report scan details types](#)
- [SAVAPI report content types](#)
- [SAVAPI signals](#)
- [SAVAPI scan statuses](#)
- [File types](#)
- [Filename flags](#)

Macros

- `#define SAVAPI_SYMBOL(s) STR(s)`
Macro to be used when loading a SAVAPI symbol while using dynamic linking.
- `#define STR(s) #s`

6.2.1 Detailed Description

6.2.2 Macro Definition Documentation

6.2.2.1 SAVAPI_SYMBOL

```
#define SAVAPI_SYMBOL(  
    s ) STR(s)
```

Macro to be used when loading a SAVAPI symbol while using dynamic linking.

Note

When using dynamic linking, it is strongly recommended to use the SAVAPI_SYMBOL macro for loading a symbol, instead of the plain function name. Check the `lib_loadlibrary_example` for implementation details.

6.2.2.2 STR

```
#define STR(  
    s ) #s
```

6.3 API version

Macros

- #define [SAVAPI_API_MAJOR_VERSION](#) 5
Major API version.
- #define [SAVAPI_API_MINOR_VERSION](#) 5
Minor API version.

6.3.1 Detailed Description

Note

The API version must be passed to the [SAVAPI_GLOBAL_INIT](#) structure and is used to ensure compatibility between the API used by the library and the API used by the application which integrates the library.

Examples of API compatibility:

Version used by application	Version used by library	Status
4.0	4.0	OK
4.0	4.1	OK
4.1	4.0	Error
3.6	4.0	Error
4.0	3.6	Error

6.3.2 Macro Definition Documentation

6.3.2.1 SAVAPI_API_MAJOR_VERSION

```
#define SAVAPI_API_MAJOR_VERSION 5
```

Major API version.

Note

This version must match exactly the version used by the library, otherwise SAVAPI library will not initialize

6.3.2.2 SAVAPI_API_MINOR_VERSION

```
#define SAVAPI_API_MINOR_VERSION 5
```

Minor API version.

Note

This version indicates which subset of features the API provides.

The version used by the library must be at least as high as this version.

6.4 Error or information categories

Macros

- `#define SAVAPI_ECAT_ERROR_IO 0`
- `#define SAVAPI_ECAT_ERROR_SCAN 1`
- `#define SAVAPI_ECAT_ERROR_UNPACK 2`
- `#define SAVAPI_ECAT_ERROR_GENERIC 3`
- `#define SAVAPI_ECAT_APC_REPORT_TTL 4`

6.4.1 Detailed Description

Note

Used by the error callbacks to categorize the errors or the information they return

6.4.2 Macro Definition Documentation

6.4.2.1 SAVAPI_ECAT_APC_REPORT_TTL

```
#define SAVAPI_ECAT_APC_REPORT_TTL 4
```

APC report ttl category

6.4.2.2 SAVAPI_ECAT_ERROR_GENERIC

```
#define SAVAPI_ECAT_ERROR_GENERIC 3
```

uncategorized error category

6.4.2.3 SAVAPI_ECAT_ERROR_IO

```
#define SAVAPI_ECAT_ERROR_IO 0
```

i/o error category

6.4.2.4 SAVAPI_ECAT_ERROR_SCAN

```
#define SAVAPI_ECAT_ERROR_SCAN 1
```

scan error category

6.4.2.5 SAVAPI_ECAT_ERROR_UNPACK

```
#define SAVAPI_ECAT_ERROR_UNPACK 2
```

unpack error category

6.5 Error levels

Macros

- #define SAVAPI_ELEVEL_ERROR 0
- #define SAVAPI_ELEVEL_WARNING 1
- #define SAVAPI_ELEVEL_INFO 2

6.5.1 Detailed Description

Note

Used by the error callbacks to categorize the returned errors

6.5.2 Macro Definition Documentation

6.5.2.1 SAVAPI_ELEVEL_ERROR

```
#define SAVAPI_ELEVEL_ERROR 0
```

error level

6.5.2.2 SAVAPI_ELEVEL_INFO

```
#define SAVAPI_ELEVEL_INFO 2
```

info level

6.5.2.3 SAVAPI_ELEVEL_WARNING

```
#define SAVAPI_ELEVEL_WARNING 1
```

warning level

6.6 Initialization flags

Macros

- `#define SAVAPI_FLAG_USE_TCP 1`
- `#define SAVAPI_FLAG_USE_LOCAL_SOCKET 2`

6.6.1 Detailed Description

Note

Initialization flags will be extended on the fly when needed! The `SAVAPI_FLAG_USE_TCP` and `SAVAPI_FLAG_USE_LOCAL_SOCKET` must not be set simultaneously

6.6.2 Macro Definition Documentation

6.6.2.1 SAVAPI_FLAG_USE_LOCAL_SOCKET

```
#define SAVAPI_FLAG_USE_LOCAL_SOCKET 2
```

local sockets will be used for communication (SAVAPI client-mode only)

6.6.2.2 SAVAPI_FLAG_USE_TCP

```
#define SAVAPI_FLAG_USE_TCP 1
```

TCP sockets will be used for communication (SAVAPI client-mode only)

6.7 Scan warnings

Macros

- `#define SAVAPI_W_DAMAGED 1`
- `#define SAVAPI_W_OLE_DAMAGED 2`
- `#define SAVAPI_W_SUSPICIOUS 4`
- `#define SAVAPI_W_PROGRESS_ABORT 8`
- `#define SAVAPI_W_HEADER_MALFORMED 16`
- `#define SAVAPI_W_POTENTIAL_ARCH_BOMB 32`
- `#define SAVAPI_W_RATIO_EXCEEDED 64`
- `#define SAVAPI_W_MAX_EXTRACTED 128`

6.7.1 Detailed Description

Note

Warnings that can be received during the scanning process

6.7.2 Macro Definition Documentation

6.7.2.1 SAVAPI_W_DAMAGED

```
#define SAVAPI_W_DAMAGED 1
```

File has potentially been damaged by virus

6.7.2.2 SAVAPI_W_HEADER_MALFORMED

```
#define SAVAPI_W_HEADER_MALFORMED 16
```

A malformed archive header was detected

6.7.2.3 SAVAPI_W_MAX_EXTRACTED

```
#define SAVAPI_W_MAX_EXTRACTED 128
```

Unpacking has reached the maximum limit of extracted data

6.7.2.4 SAVAPI_W_OLE_DAMAGED

```
#define SAVAPI_W_OLE_DAMAGED 2
```

OLE-File is potentially damaged

6.7.2.5 SAVAPI_W_POTENTIAL_ARCH_BOMB

```
#define SAVAPI_W_POTENTIAL_ARCH_BOMB 32
```

This file could be an archive bomb, ratio might be exceeded or something else might have happened to trigger that detection

6.7.2.6 SAVAPI_W_PROGRESS_ABORT

```
#define SAVAPI_W_PROGRESS_ABORT 8
```

An abort was triggered by the progress callback

6.7.2.7 SAVAPI_W_RATIO_EXCEEDED

```
#define SAVAPI_W_RATIO_EXCEEDED 64
```

The ratio set by the application regarding unpacking size in archives has been exceeded

6.7.2.8 SAVAPI_W_SUSPICIOUS

```
#define SAVAPI_W_SUSPICIOUS 4
```

File is suspicious

6.8 Iframes informations

Macros

- `#define SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE 1`
- `#define SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL 2`
- `#define SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS 4`
- `#define SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS 8`

6.8.1 Detailed Description

Note

Informations that can be received during the scanning process
These options are deprecated

6.8.2 Macro Definition Documentation

6.8.2.1 SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL

```
#define SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL 2
```

The object is very small and as such almost invisible to the user surfing the site

6.8.2.2 SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE

```
#define SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE 1
```

The object is invisible to the user surfing the site

6.8.2.3 SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS

```
#define SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS 8
```

The object is likely of a malicious nature

6.8.2.4 SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS

```
#define SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS 4
```

The object is inserted at a very uncommon position in the HTML code

6.9 Scan information

Macros

- #define SAVAPI_I_OLEFILE 1
- #define SAVAPI_I_TEMPLATE 2
- #define SAVAPI_I_MACROS_PRESENT 4
- #define SAVAPI_I_ALL_MACROS_DELETED 8
- #define SAVAPI_I_OLE_ENCRYPTED 16
- #define SAVAPI_I_ACTIVE_CONTENT_PRESENT 32
- #define SAVAPI_I_MAILBOX 64
- #define SAVAPI_I_MACRO_AUTOSTART 128
- #define SAVAPI_I_APC_SCAN_DURATION 256
- #define SAVAPI_I_APC_RESULT_EXPIRY 512

6.9.1 Detailed Description

Note

Information that can be received during the scanning process

6.9.2 Macro Definition Documentation

6.9.2.1 SAVAPI_I_ACTIVE_CONTENT_PRESENT

```
#define SAVAPI_I_ACTIVE_CONTENT_PRESENT 32
```

SCRIPT: html contains active content (JS/VBS, etc.) - this flag is deprecated

6.9.2.2 SAVAPI_I_ALL_MACROS_DELETED

```
#define SAVAPI_I_ALL_MACROS_DELETED 8
```

OLE: all macros were deleted - this flag is deprecated

6.9.2.3 SAVAPI_I_APC_RESULT_EXPIRY

```
#define SAVAPI_I_APC_RESULT_EXPIRY 512
```

APC: report the time (in seconds) after which an APC detection result expires

6.9.2.4 SAVAPI_I_APC_SCAN_DURATION

```
#define SAVAPI_I_APC_SCAN_DURATION 256
```

APC: report an estimation time (in seconds) until a response will be available from the APC

Note

The estimation for an APC scan may be updated by several calls with this information

6.9.2.5 SAVAPI_I_MACRO_AUTOSTART

```
#define SAVAPI_I_MACRO_AUTOSTART 128
```

OLE: contains macros with autostart enabled

6.9.2.6 SAVAPI_I_MACROS_PRESENT

```
#define SAVAPI_I_MACROS_PRESENT 4
```

OLE: contains macros

6.9.2.7 SAVAPI_I_MAILBOX

```
#define SAVAPI_I_MAILBOX 64
```

ARCHIVE: Mailbox detected

6.9.2.8 SAVAPI_I_OLE_ENCRYPTED

```
#define SAVAPI_I_OLE_ENCRYPTED 16
```

OLE: encrypted marker, only for DOCs

6.9.2.9 SAVAPI_I_OLEFILE

```
#define SAVAPI_I_OLEFILE 1
```

OLE: file is a compound doc (OLE2)

6.9.2.10 SAVAPI_I_TEMPLATE

```
#define SAVAPI_I_TEMPLATE 2
```

OLE: contains a word template

6.10 SAVAPI options

Typedefs

- typedef enum SAVAPI_option SAVAPI_OPTION

Enumerations

- enum SAVAPI_option {
 SAVAPI_OPTION_CWD = 1, SAVAPI_OPTION_CONF,
 SAVAPI_OPTION_ARCHIVE_SCAN, SAVAPI_OPTION_ARCHIVE_MAX_SIZE,
 SAVAPI_OPTION_ARCHIVE_MAX_REC, SAVAPI_OPTION_ARCHIVE_MAX_RATIO,
 SAVAPI_OPTION_ARCHIVE_MAX_COUNT, SAVAPI_OPTION_MAILBOX_SCAN,
 SAVAPI_OPTION_HEUR_MACRO, SAVAPI_OPTION_HEUR_LEVEL,
 SAVAPI_OPTION_SCAN_TEMP, SAVAPI_OPTION_SCAN_TIMEOUT,
 SAVAPI_OPTION_REPAIR, SAVAPI_OPTION_NOTIFY_REPAIR,
 SAVAPI_OPTION_NOTIFY_OFFICE, SAVAPI_OPTION_NOTIFY_OFFICE_MACRO,
 SAVAPI_OPTION_NOTIFY_ALERTURL, SAVAPI_OPTION_DETECT_ADSPY,
 SAVAPI_OPTION_DETECT_APPL, SAVAPI_OPTION_DETECT_BDC,
 SAVAPI_OPTION_DETECT_DIAL, SAVAPI_OPTION_DETECT_GAME,
 SAVAPI_OPTION_DETECT_HIDDENEXT, SAVAPI_OPTION_DETECT_JOKE,
 SAVAPI_OPTION_DETECT_PCK, SAVAPI_OPTION_DETECT_PHISH,
 SAVAPI_OPTION_DETECT_SPR, SAVAPI_OPTION_IFRAMES_URL,
 SAVAPI_OPTION_REPORT_ENCRYPTED_MIME, SAVAPI_OPTION_SCAN_MODE,
 SAVAPI_OPTION_MIME_SCAN, SAVAPI_OPTION_PGP_SCAN,
 SAVAPI_OPTION_SCAN_PROGRESS, SAVAPI_OPTION_DETECT_ADWARE,
 SAVAPI_OPTION_DETECT_PFS, SAVAPI_OPTION_RESERVED1,
 SAVAPI_OPTION_RESERVED2, SAVAPI_OPTION_APC_CONNECTION_TIMEOUT,
 SAVAPI_OPTION_APC_SCAN_TIMEOUT, SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL,
 SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL, SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART
 ,
 SAVAPI_OPTION_DETECT_PUA, SAVAPI_OPTION_APC_REPORT_SCAN_TTL,
 SAVAPI_OPTION_FPC, SAVAPI_OPTION_FPC_TIMEOUT,
 SAVAPI_OPTION_APC_PE_MODE, SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY,
 SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED, SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY
 ,
 SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL, SAVAPI_OPTION_APC_ELF_MODE,
 SAVAPI_OPTION_APC_MACH_O_MODE, SAVAPI_OPTION_PRODUCT = 1000,
 SAVAPI_OPTION_DETECT_ALLTYPES, SAVAPI_OPTION_SCAN_TIMEOUTS,
 SAVAPI_OPTION_SAVAPI = 2000, SAVAPI_OPTION_AVE_VERSION,
 SAVAPI_OPTION_VDF_VERSION, SAVAPI_OPTION_PID,
 SAVAPI_OPTION_EXPIRE, SAVAPI_OPTION_VDFSIGCOUNT,
 SAVAPI_OPTION_SELECTABLE_DETECT, SAVAPI_OPTION_DESCR_ADSPY,
 SAVAPI_OPTION_DESCR_APPL, SAVAPI_OPTION_DESCR_BDC,
 SAVAPI_OPTION_DESCR_DIAL, SAVAPI_OPTION_DESCR_GAME,
 SAVAPI_OPTION_DESCR_HIDDENEXT, SAVAPI_OPTION_DESCR_JOKE,
 SAVAPI_OPTION_DESCR_PCK, SAVAPI_OPTION_DESCR_PHISH,
 SAVAPI_OPTION_DESCR_SPR, SAVAPI_OPTION_VDF_DATE,
 SAVAPI_OPTION_MALWARE_NAMES_FILE, SAVAPI_OPTION_DESCR_ADWARE,
 SAVAPI_OPTION_DESCR_PFS, SAVAPI_OPTION_DESCR_PUA }

6.10.1 Detailed Description

Remarks

Almost each option used to configure the SAVAPI instance (the paths to the temporary folders, the scanning options) has a default value that is written in its description as a note (i.e. Default value: <value>).

In client-mode, the default values are dependent to the configuration used to start the SAVAPI service, so the provided defaults only applies in library-mode!!!

The options that has no default (i.e. unsupported options, obsolete, ignored) will be marked with the "Default value: None" string.

6.10.2 Typedef Documentation**6.10.2.1 SAVAPI_OPTION**

```
typedef enum SAVAPI_option SAVAPI_OPTION
```

6.10.3 Enumeration Type Documentation**6.10.3.1 SAVAPI_option**

```
enum SAVAPI_option
```

Enumerator

SAVAPI_OPTION_CWD	<p>Specifies current working directory for SAVAPI.</p> <p>Note</p> <p>This eliminates the need to specify full paths in filenames.</p> <p>Available only in client-mode.</p> <p>Default value: None</p>
SAVAPI_OPTION_CONF	<p>Specifies the configuration file that is used.</p> <p>Note</p> <p>The configuration file will be (re-)read as part of this request.</p> <p>Available only in client-mode.</p> <p>Default value: None</p>
SAVAPI_OPTION_ARCHIVE_SCAN	<p>Activates archive detection and scanning.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>

Enumerator

SAVAPI_OPTION_ARCHIVE_MAX_SIZE	<p>Set the maximum allowed size (in bytes) for any file within an archive.</p> <p>Note</p> <p>A value of "0" means the maximum allowed value (INT64_MAX bytes).</p> <p>This setting has no meaning if ARCHIVE_SCAN is deactivated.</p> <p>Default value: 1073741824 (1G)</p>
SAVAPI_OPTION_ARCHIVE_MAX_REC	<p>Set the maximum allowed recursion within an archive.</p> <p>Note</p> <p>A value of "0" means the maximum allowed value (1000 recursion levels).</p> <p>This setting has no meaning if ARCHIVE_SCAN is deactivated.</p> <p>Default value: 200</p>
SAVAPI_OPTION_ARCHIVE_MAX_RATIO	<p>Set the maximum allowed decompressing-ratio within an archive.</p> <p>Note</p> <p>A value of "0" means the maximum allowed value (INT32_MAX).</p> <p>This setting has no meaning if ARCHIVE_SCAN is deactivated.</p> <p>Default value: 150</p>
SAVAPI_OPTION_ARCHIVE_MAX_COUNT	<p>Set the maximum allowed number of files within an archive.</p> <p>Note</p> <p>A value of "0" means the maximum allowed value (INT64_MAX).</p> <p>This setting has no meaning if ARCHIVE_SCAN is deactivated.</p> <p>Default value: 0</p>
SAVAPI_OPTION_MAILBOX_SCAN	<p>Activates detection and scanning of mailboxes.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_HEUR_MACRO	<p>Activates heuristic macro detection.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>

Enumerator

SAVAPI_OPTION_HEUR_LEVEL	<p>Set the heuristic level for the engine. The available levels are:</p> <ul style="list-style-type: none"> • 0 - Disable heuristic detection. • 1 - Lazy heuristic detection. This is the lowest possible mode, detection is not very good, but the false positives number will be low. • 2 - Normal heuristic detection. • 3 - High heuristic detection. This is the highest possible mode, but the false positives number will be high. <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_SCAN_TEMP	<p>Set the temporary directory used for scanning files.</p> <p>Note</p> <p>SAVAPI may use other temporary directories for files that are not being scanned. These other directories can be specified with command-line arguments or in a configuration file.</p> <p>Default value: The system temporary folder</p> <p>It is recommended for the location to not be a directory that contains sensitive files, such as SAVAPI binaries or configuration files.</p>
SAVAPI_OPTION_SCAN_TIMEOUT	<p>Set the maximum number of seconds allowed to scan a file before aborting.</p> <p>Note</p> <p>Available values: 0 - 86400 (1 second - 24 hours)</p> <p>Default value: 0 (no timeout)</p>
SAVAPI_OPTION_REPAIR	<p>Activates the repairing of infected files.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_NOTIFY_REPAIR	<p>Activates the notification of reparable infected files.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_NOTIFY_OFFICE	<p>Activates the detection of Microsoft Office OLE documents.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>

Enumerator

SAVAPI_OPTION_NOTIFY_OFFICE_MACRO	<p>Activates the detection of macros within Microsoft Office OLE documents.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_NOTIFY_ALERTURL	<p>Activates the notification of virus description url.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_ADSPY	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_DETECT_APPL	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_BDC	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_DETECT_DIAL	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_DETECT_GAME	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_HIDDENEXT	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_DETECT_JOKE	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_PCK	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p> <p>This option is deprecated.</p>
SAVAPI_OPTION_DETECT_PHISH	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>

Enumerator

SAVAPI_OPTION_DETECT_SPR	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_IFRAMES_URL	<p>Activate IFRAME detection.</p> <p>Note</p> <p>Default value: 0 (disabled)</p> <p>This option is deprecated.</p>
SAVAPI_OPTION_REPORT_ENCRYPTED_MIME	<p>Activate reporting of encrypted mails.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_SCAN_MODE	<p>Set the scanning method. Available options are:</p> <ul style="list-style-type: none"> • SMART - Smart Extensions scan mode. The files scanned for malware are chosen by SAVAPI. The choice is made based on the files content. This is the recommended setting. • ALL - All scan mode. Files are scanned for malware, no matter their content or extension. • EXTLIST - Extensions List scan mode. Only files with specific extensions are scanned for malware content. <p>Note</p> <p>Default value: SMART</p>
SAVAPI_OPTION_MIME_SCAN	<p>Activate detection and scanning of mails.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_PGP_SCAN	<p>Activate scanning and reporting of PGP encrypted files.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_SCAN_PROGRESS	<p>Activate regular interval messages during scanning to confirm that the SAVAPI service is still alive.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_ADWARE	<p>Activate detection for the ADWARE type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>

Enumerator

SAVAPI_OPTION_DETECT_PFS	<p>Activate detection for the PFS (possible fraudulent software) type.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_RESERVED1	This option is reserved and must not be used.
SAVAPI_OPTION_RESERVED2	This option is reserved and must not be used.
SAVAPI_OPTION_APC_CONNECTION_TIMEOUT	<p>Set the maximum number of seconds before an APC connection establish attempt times out.</p> <p>Note</p> <p>Available values: 0 - 86400 (1 second - 24 hours)</p> <p>If 0, SAVAPI will wait indefinitely for a connection to establish.</p> <p>Default value: 20</p>
SAVAPI_OPTION_APC_SCAN_TIMEOUT	<p>Set the maximum number of seconds before an APC file scan times out.</p> <p>Note</p> <p>Available values: 0 - 86400 (1 second - 24 hours)</p> <p>If 0, SAVAPI will wait indefinitely for data transfer to/from APC.</p> <p>Default value: 30</p>
SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL ↔ LEVEL	<p>Set the APC_CHECK_RISK_RATING_LEVEL threshold.</p> <p>Note</p> <p>A hash request will be sent to APC only if the malware probability is greater than or equal to this threshold.</p> <p>Available values: 0 - 7 (Very Low Risk - Very High Risk)</p> <p>Default value: 4</p>

Enumerator

SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL	<p>Set the APC_UPLOAD_RISK_RATING_LEVEL threshold.</p> <p>Note</p> <p>An unknown file will be uploaded to APC only if the malware probability is greater than or equal to this threshold.</p> <p>If APC_CHECK_RISK_RATING_LEVEL is greater than APC_UPLOAD_RISK_RATING_LEVEL, then the file will be uploaded only if the malware probability is greater than or equal to APC_CHECK_RISK_RATING_LEVEL.</p> <p>Available values: 0 - 7 (Very Low Risk - Very High Risk)</p> <p>Default value: 4</p>
SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART	<p>Activates the detection of macros with autostart enabled within Microsoft Office OLE documents.</p> <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_DETECT_PUA	<p>Activate detection for the specified type.</p> <p>Note</p> <p>Default value: 1 (enabled)</p>
SAVAPI_OPTION_APC_REPORT_SCAN_TTL	<p>Activate reporting of APC TTLs. If enabled, more SAVAPI_CALLBACK_REPORT_ERROR triggers will be made, having the following information:</p> <ul style="list-style-type: none"> error_data.level = SAVAPELEVEL_INFO error_data.category = SAVAPECAT_APC_REPORT_TTL error_data.code = SAVAPE_I_APC_RESULT_EXPIRY or SAVAPE_I_APC_SCAN_DURATION error_data.options->type = the actual TTL (in seconds) error_data.file_info.name = the file name or the hash of the file, in case of hash scanning <p>Note</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_FPC	<p>Activate the FPC detection check.(1 = enabled, 0 = disabled)</p> <p>Note</p> <p>Default value: 0</p>

Enumerator

SAVAPI_OPTION_FPC_TIMEOUT	<p>Set the maximum number of seconds before a FPC check times out.</p> <p>Note</p> <p>Available values: 0 - 86400 (1 second - 24 hours)</p> <p>If 0, SAVAPI will wait indefinitely for a FPC check.</p> <p>Default value: 20</p>
SAVAPI_OPTION_APC_PE_MODE	<p>Specifies the APC scanning mode for PE files.</p> <p>Note</p> <p>Available values:</p> <ul style="list-style-type: none"> • DISABLED - no PE file will be scanned with APC • CHECK-ONLY - only file hashes of PE files will be sent to APC, no uploads • FULL - PE files will be completely scanned with APC (hash checks and uploads) <p>Default value: FULL</p> <p>This option depends on the value of <code>apc_mode</code> from the SAVAPI_APC_GLOBAL_INIT structure</p>
SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY	<p>Specifies the policy of the files that will be scanned with APC.</p> <p>Note</p> <p>Available values: AUTO (all files extensions supported by SAVAPI internal list will be scanned with APC) CUSTOM (user-defined list of extensions to be scanned with APC)</p> <p>The extensions defined by "CUSTOM" option must exist also in SAVAPI internal list, otherwise will not be scanned with APC</p> <p>When changing to a different policy, all the extensions filters (SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED, SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY, SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL) will be reset</p> <p>Default value: CUSTOM</p>

Enumerator

SAVAPI_OPTION_APC_FILE_EXTENSIONS_↔ DISABLED	<p>Specifies a list of extensions of the files that will not be scanned with APC.</p> <p>Note</p> <p>Available values: a string containing semicolon separated extensions (including the dot).</p> <p>Maximum extension length is 255 characters (including the dot). The maximum number of extensions is 128.</p> <p>This option has a higher priority and will refine SAVAPI_OPTION_APC_FILE_EXTENSIONS_↔_POLICY option</p> <p>Default value: none Example: .xls;.bin;.doc</p>
SAVAPI_OPTION_APC_FILE_EXTENSIONS_↔ CHECK_ONLY	<p>Specifies a list of extensions of the files that will be hashed-scanned with APC.</p> <p>Note</p> <p>Available values: a string containing semicolon separated extensions (including the dot).</p> <p>Maximum extension length is 255 characters (including the dot). The maximum number of extensions is 128.</p> <p>This option has a higher priority and will refine SAVAPI_OPTION_APC_FILE_EXTENSIONS_↔_POLICY option</p> <p>This option depends on the value of <code>apc_mode</code> from the SAVAPI_APC_GLOBAL_INIT structure</p> <p>Default value: none Example: .xls;.bin;.doc</p>
SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL	<p>Specifies a list of extension for the files that will be hashed-checked or uploaded to APC.</p> <p>Note</p> <p>Available values: a string containing semicolon separated extensions (including the dot).</p> <p>Maximum extension length is 255 characters (including the dot). The maximum number of extensions is 128.</p> <p>This option has a higher priority and will refine SAVAPI_OPTION_APC_FILE_EXTENSIONS_↔_POLICY option</p> <p>This option depends on the value of <code>apc_mode</code> from the SAVAPI_APC_GLOBAL_INIT structure</p> <p>Default value: none Example: .xls;.bin;.doc</p>

Enumerator

SAVAPI_OPTION_APC_ELF_MODE	<p>Specifies the APC scanning mode for ELF files.</p> <p>Note</p> <p>Available values:</p> <ul style="list-style-type: none"> • DISABLED - no ELF file will be scanned with APC • CHECK-ONLY - only file hashes of ELF files will be sent to APC, no uploads • FULL - ELF files will be completely scanned with APC (hash checks and uploads) <p>Default value: DISABLED</p> <p>This option depends on the value of <code>apc_mode</code> from the SAVAPI_APC_GLOBAL_INIT structure</p>
SAVAPI_OPTION_APC_MACH_O_MODE	<p>Specifies the APC scanning mode for Mach-O and Apple Universal Binary files.</p> <p>Note</p> <p>Available values:</p> <ul style="list-style-type: none"> • DISABLED - no Mach-O file will be scanned with APC • CHECK-ONLY - only file hashes of Mach-O files will be sent to APC, no uploads • FULL - Mach-O files will be completely scanned with APC (hash checks and uploads) <p>Default value: DISABLED</p> <p>This option depends on the value of <code>apc_mode</code> from the SAVAPI_APC_GLOBAL_INIT structure</p>
SAVAPI_OPTION_PRODUCT	<p>Set the key-id that is required by the application.</p> <p>Note</p> <p>SAVAPI will check if the key-id is within the license and that it is not expired. If it is available and is valid, the application is free to use SAVAPI. If not, most requests will result in an error response.</p>
SAVAPI_OPTION_DETECT_ALLTYPES	<p>Activate detection for all types.</p>
SAVAPI_OPTION_SCAN_TIMEOUTS	<p>Set all three timeouts (APC Connection Timeout, APC Scan Timeout, Scan Timeout) using a single command.</p> <p>Note</p> <p>The timeouts must be set in the following order, separated by space. (APC Connection Timeout, APC Scan Timeout, Scan Timeout). Ex: "20 30 40"</p>

Enumerator

SAVAPI_OPTION_SAVAPI	Retrieves SAVAPI Service version for client-mode, or, for library-mode, it retrieves the SAVAPI library version.
SAVAPI_OPTION_AVE_VERSION	Retrieve engine version number
SAVAPI_OPTION_VDF_VERSION	Retrieve vdf(-set) version number
SAVAPI_OPTION_PID	<p>Retrieve the process-id for the SAVAPI process that is currently handling the TCP/IP connection.</p> <p>Note</p> <p>Available only in client-mode.</p>
SAVAPI_OPTION_EXPIRE	Retrieve the expiration date of the SAVAPI license (YYYYMMDD)
SAVAPI_OPTION_VDFSIGCOUNT	Retrieve the number of signatures in the vdf(-set)
SAVAPI_OPTION_SELECTABLE_DETECT	<p>Retrieve the various types that can be detected (and dynamically turned on/off).</p> <p>Note</p> <p>The types are returned as a comma separated list. The current value would be: ADWARE,ADSPY,APPL,BDC,DIAL,GAME,HIDDENEXT,JOKE,PCK,PF</p>
SAVAPI_OPTION_DESCR_ADSPY	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_APPL	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_BDC	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_DIAL	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_GAME	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_HIDDENEXT	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_JOKE	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_PCK	Retrieve the English description for the given type. This option is deprecated.
SAVAPI_OPTION_DESCR_PHISH	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_SPR	Retrieve the English description for the given type.
SAVAPI_OPTION_VDF_DATE	Retrieve the creation date of the vdf(-set). Date is the form of YYYYMMDD.
SAVAPI_OPTION_MALWARE_NAMES_FILE	<p>Retrieve the path to the file containing the dump of the malware names.</p> <p>Note</p> <p>This option's value should only be checked after a successfully call to the SAVAPI_extract_malware_names function, otherwise it will contain an empty string.</p> <p>Default value: None</p>
SAVAPI_OPTION_DESCR_ADWARE	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_PFS	Retrieve the English description for the given type.
SAVAPI_OPTION_DESCR_PUA	Retrieve the English description for the given type.

6.11 SAVAPI global options

Typedefs

- typedef enum [SAVAPI_global_option](#) SAVAPI_GLOBAL_OPTION

Enumerations

- enum [SAVAPI_global_option](#) {
[SAVAPI_OPTION_G_OA_EXTENSIONS_LIST](#) = 3000 , [SAVAPI_OPTION_G_OA_EXCEPTED_FILES](#) ,
[SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES](#) , [SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES](#)
 ,
[SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES](#) , [SAVAPI_OPTION_G_OA_CACHE_SCAN_NETWORK_DRIVES](#)
 ,
[SAVAPI_OPTION_G_OA_SCAN_TIMEOUT](#) , [SAVAPI_OPTION_G_OA_MALWARE_RESPONSE_TTL](#) ,
[SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT](#) = 4000 , [SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES](#)
 ,
[SAVAPI_OPTION_G_PROXY](#) }

6.11.1 Detailed Description

Remarks

Almost each option used to configure the SAVAPI global has a default value that is written in its description as a note (i.e. Default value: <value>).

The options that have no default (i.e. unsupported options, obsolete, ignored) will be marked with the "Default value: None" string.

6.11.2 Typedef Documentation

6.11.2.1 SAVAPI_GLOBAL_OPTION

```
typedef enum SAVAPI\_global\_option SAVAPI_GLOBAL_OPTION
```

6.11.3 Enumeration Type Documentation

6.11.3.1 SAVAPI_global_option

```
enum SAVAPI\_global\_option
```

Enumerator

SAVAPI_OPTION_G_OA_EXTENSIONS_LIST	<p>Set OnAccess scanner file extensions list. If set to empty string, all files are scanned. Maximum extension length is 12 characters. The maximum extension number is 150 extensions. Global OnAccess options "Global SET" requests are available to configure the SAVAPI OnAccess scanner.</p> <p>Note</p> <p>The extensions specified here are only considered for files not being mapped for execution. Any file mapped for execution, no matter its extension, will be scanned. It is therefore possible that files having extensions others than those defined here will be scanned. The separator is the semicolon. White spaces, if present, are considered as being part of the extension. Example: .doc;.xls;.txt;.avi</p>
------------------------------------	--

Enumerator

SAVAPI_OPTION_G_OA_EXCEPTED_FILES	<p>Set OnAccess scanner excepted file objects list. If set to empty string, then no objects will be excluded from on-access scanning. Maximum length of the string is 6000 characters, string terminator included.</p> <p>Wildcards are accepted, only if present after the last path separator (backslash). Any files and folders specified here will be ignored, even if they will be mapped for execution. For each drive, you can specify a maximum of 20 exceptions by entering the complete path. The maximum number of exceptions without a complete path is 64.</p> <p>Note</p> <p>If a directory is excluded, all its sub-directories are automatically also excluded.</p> <p>Keeping this list short is HIGHLY RECOMMENDED, since every file accessed by the operating system will be cross-checked against this list.</p> <p>This option is case sensitive. The separator is the semicolon. White spaces, if present, are considered as being part of the path.</p> <p>Examples:</p> <pre>C:\Folder\file.exe C:\Folder\Subfolder*.doc? C:\Exclude\All\From\Here\ \Device\HarddiskDmVolumes\PhysicalDmVolumes\BlockVolume1\ *.mdb;*.md? F:</pre> <p>Statements like:</p> <pre>C:\Folder*\file.exe C:\Folde?\fi*.exe \\.\c:\file.exe \??\c:\file.exe</pre> <p>are considered invalid. When excluding an entire drive, not using the backslash after the drive quotation mark is faster. 'F:' performs faster than 'F:\'. Statements exclusively comprising the following characters are invalid: * (asterisk), ? (question mark), / (forward slash), \ (backslash), . (dot), : (colon).</p>
-----------------------------------	--

Enumerator

SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES	<p>Set OnAccess scanner excepted processes list. If set to empty string, then no processes will be excluded from on-access scanning. All file actions performed by processes defined here will be excluded from the on-access scan operation. Maximum number of excepted processes is 128. Maximum length is 6000 characters, string terminator included. Wildcards are accepted only if present after the last path separator(backslash). Excluding a process without full path details only applies to processes where the executable files are located on hard disk drives. Full network path is required for processes whose executable is located on remote drives. Do not specify any exceptions for processes where the executable files are located on dynamic drives(e.g.: USB keys).</p> <p>Note</p> <p>The Windows Explorer and the operating system itself cannot be excluded.</p> <p>The specified path and file name of each process must contain a maximum of 255 characters.</p> <p>This option is case sensitive. The separator is the semicolon. White spaces, if present, are considered as being part of the path.</p> <p>Examples: C:\Folder1\Subfolder\application.exe C:\Folder2\Subfolder\application?.exe C:\Folder3\Subfolder\app*.exe C:\Folder4\Subfolder*.exe Application.exe App*.exe</p> <p>Statements like: C:\Folder*\file.exe C:\Folde?\fi*.exe * \\.\c:\file.exe \??\c:\file.exe</p> <p>are considered invalid. Statements exclusively comprising the following characters are invalid: * (asterisk), ? (question mark), / (forward slash), \ (backslash), . (dot), : (colon).</p>
SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES ↔	<p>Set OnAccess scanner at file changes.</p> <p>Note</p> <p>All the files that were previously accessed and scanned will be scanned again, whenever possible, when modifications are detected.</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES ↔	<p>Set OnAccess to scan network files.</p> <p>Note</p> <p>All the files accessed on a network location will be scanned as well.</p> <p>Default value: 0 (disabled)</p>

Enumerator

SAVAPI_OPTION_G_OA_CACHE_SCAN_↔ NETWORK_DRIVES	<p>Set OnAccess to cache scan results for network files.</p> <p>Note</p> <p>The results are cached based on file modify time.</p> <p>Default value: 0 (disabled)</p>
SAVAPI_OPTION_G_OA_SCAN_TIMEOUT	<p>Set the maximum number of seconds allowed to scan an OnAccess file before aborting.</p> <p>Note</p> <p>Available values: 0 - 60 (1 second - 60 seconds)</p> <p>If 0, SAVAPI will wait maximum (60 seconds) for OnAccess file to be scanned.</p> <p>Default value: 25</p>
SAVAPI_OPTION_G_OA_MALWARE_↔ RESPONSE_TTL	<p>Set the interval in which a file detected as malware will not be scanned again when it is accessed.</p> <p>Note</p> <p>If a file was detected as malware and a second access for it comes before the end of the interval 0 - SAVAPI_OPTION_G_OA_↔ MALWARE_RESPONSE_TTL, the file will not be scanned again and the result for the second access will be the same as for the first access. If the callback SAVAPI_CALLBACK_OA_FILE_RESULT is defined, it is up to the callback implementation to grant or deny the access.</p> <p>Available values: 0 - 300 (0 seconds - 300 seconds)</p> <p>If 0, SAVAPI will scan the file at each access</p> <p>Default value: 5</p>
SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT	<p>Specifies the number of seconds after which SAVAPI will try to establish another connection to FPC.</p> <p>Note</p> <p>Available values: 1 - 86400 (0 seconds - 86400 seconds)</p> <p>Default value: 300</p>
SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES	<p>Specifies the maximum number of consecutive timeouts allowed before declaring FPC unreachable.</p> <p>Note</p> <p>Available values: 0 - INT16_MAX</p> <p>Default value: 5</p> <p>If set to 0, FPC will always try to check the files</p>

Enumerator

SAVAPI_OPTION_G_PROXY	<p>Explicitly set a proxy server to be used by all SAVAPI modules (APC, FPC)</p> <p>Note</p> <p>The parameter holds the host name or IP address. To specify a port number in this string, append <code>:[port]</code> to the end of the host name. If not specified, SAVAPI will use as default the port 1080.</p> <p>The proxy string may be prefixed with <code>[scheme]://</code> to specify the kind of proxy to be used. Supported schemes are: <code>http://</code>, <code>https://</code>, <code>socks4://</code>, <code>socks4a://</code> and <code>socks5://</code>.</p> <p>If no protocol is specified, the proxy will be treated as a HTTP proxy server.</p> <p>Only ASCII characters are supported.</p> <p>This option must be applied after <code>SAVAPI_initialize()</code>, but before <code>SAVAPI_create_instance()</code> in order to have any effect. All SAVAPI (and APC) instances created before calling this function will still use the proxy server that was defined at the time they were created.</p>
-----------------------	---

6.12 SAVAPI OnAccess result

Enumerations

- enum `SAVAPI_OA_SCAN_RESULT` {
`SAVAPI_OA_RESULT_ALLOW = 0` , `SAVAPI_OA_RESULT_DENY` ,
`SAVAPI_OA_RESULT_DELETE` , `SAVAPI_OA_RESULT_RENAME` ,
`SAVAPI_OA_RESULT_WIPE` }

6.12.1 Detailed Description

6.12.2 Enumeration Type Documentation

6.12.2.1 SAVAPI_OA_SCAN_RESULT

enum `SAVAPI_OA_SCAN_RESULT`

Enumerator

SAVAPI_OA_RESULT_ALLOW	Allow access to the file.
SAVAPI_OA_RESULT_DENY	Deny access to the file.
SAVAPI_OA_RESULT_DELETE	Deny access and delete the file.
SAVAPI_OA_RESULT_RENAME	Deny access and rename the file if it is executable.
SAVAPI_OA_RESULT_WIPE	Deny access and wipe the file.

6.13 SAVAPI APC scan callback return values

Enumerations

- enum [SAVAPI_APC_SCAN_RESULT](#) {
[SAVAPI_APC_SCAN_CONTINUE](#) , [SAVAPI_APC_SCAN_STOP](#) ,
[SAVAPI_APC_SCAN_REPORT](#) }

6.13.1 Detailed Description

6.13.2 Enumeration Type Documentation

6.13.2.1 SAVAPI_APC_SCAN_RESULT

enum [SAVAPI_APC_SCAN_RESULT](#)

Enumerator

SAVAPI_APC_SCAN_CONTINUE	Continue with the next stage of the scan
SAVAPI_APC_SCAN_STOP	Stop scanning the current file
SAVAPI_APC_SCAN_REPORT	Stop scanning the current file and report the information given in APC_set_report_info_t

6.14 Callbacks' ids

Macros

- #define [SAVAPI_CALLBACK_REPORT_FILE_STATUS](#) 0
Triggered after a file is scanned. The callback data contains the status of the last scanned file.
- #define [SAVAPI_CALLBACK_REPORT_ERROR](#) 3
Triggered to report an error or a warning.
- #define [SAVAPI_CALLBACK_PRE_SCAN](#) 4
Triggered before the scanning begins. Can be used to create filters. For example, if we want to scan only .exe files, we install a PRE_SCAN callback. Before each file is scanned, the PRE_SCAN callback will be called. Inside our implementation of the callback, we implement the filter. If the returned code is success, the file will be scanned, otherwise it will be skipped.
- #define [SAVAPI_CALLBACK_ARCHIVE_OPEN](#) 5
Triggered before opening an archive. If the returned code is success, the archive will be opened, otherwise it will be skipped from opening.
- #define [SAVAPI_CALLBACK_PROGRESS_REPORT](#) 6
Triggered when messages related to scan progress are available.
- #define [SAVAPI_CALLBACK_CONTENT_REPORT](#) 7
Triggered when messages related to scan (progress, warnings or infos that are not error_callback related) are available. IFRAME detection ([SAVAPI_OPTION_IFRAMES_URL](#)) will be reported through this callback.

- `#define SAVAPI_CALLBACK_SCAN_DETAILS_REPORT 8`
Triggered when messages related to scan process details are available. The virus description url ([SAVAPI_OPTION_NOTIFY_ALERTURL](#))
- `#define SAVAPI_CALLBACK_OA_FILE_RESULT 9`
Triggered after a file was scanned with OnAccess, in order to decide what action to take The action is taken depending on the return code of the callback (see [SAVAPI_OA_SCAN_RESULT](#))
- `#define SAVAPI_CALLBACK_APC_SCAN 10`

6.14.1 Detailed Description

6.14.2 Macro Definition Documentation

6.14.2.1 SAVAPI_CALLBACK_APC_SCAN

```
#define SAVAPI_CALLBACK_APC_SCAN 10
```

Triggered at various stages of scanning a file with APC:

- before any action is performed on the file (see [SAVAPI_APC_STAGE_PRE_FILTER](#));
- after the file passed the APC filter, but before the APC hash check (see [SAVAPI_APC_STAGE_PRE_HASH_CHECK](#));
- after the APC hash check, but before the APC upload (see [SAVAPI_APC_STAGE_PRE_UPLOAD](#));
- after the entire APC scan was finished (see [SAVAPI_APC_STAGE_POST_SCAN](#)).

An action is taken depending on the return code of the callback:

- [SAVAPI_APC_SCAN_CONTINUE](#) - the scan will continue and a new callback will be triggered at the next available stage;
- [SAVAPI_APC_SCAN_STOP](#) - stop the APC scan for the current file;
- [SAVAPI_APC_SCAN_REPORT](#) - stop the APC scan for the current file and use the information given in the [APC_set_report_info_t](#) function. This information will be reported on the [SAVAPI_CALLBACK_REPORT_FILE_STATUS](#) callback.

Note

If the current stage is [SAVAPI_APC_STAGE_POST_SCAN](#), the return codes [SAVAPI_APC_SCAN_CONTINUE](#) and [SAVAPI_APC_SCAN_STOP](#) have the same effect.

Currently, this callback is not triggered in client-mode.

6.14.2.2 SAVAPI_CALLBACK_ARCHIVE_OPEN

```
#define SAVAPI_CALLBACK_ARCHIVE_OPEN 5
```

Triggered before opening an archive. If the returned code is success, the archive will be opened, otherwise it will be skipped from opening.

Note

Currently, this callback is not triggered in client-mode.

6.14.2.3 SAVAPI_CALLBACK_CONTENT_REPORT

```
#define SAVAPI_CALLBACK_CONTENT_REPORT 7
```

Triggered when messages related to scan (progress, warnings or infos that are not error_callback related) are available. IFRAME detection ([SAVAPI_OPTION_IFRAMES_URL](#)) will be reported through this callback.

Note

This callback is deprecated.

6.14.2.4 SAVAPI_CALLBACK_OA_FILE_RESULT

```
#define SAVAPI_CALLBACK_OA_FILE_RESULT 9
```

Triggered after a file was scanned with OnAccess, in order to decide what action to take. The action is taken depending on the return code of the callback (see [SAVAPI_OA_SCAN_RESULT](#))

6.14.2.5 SAVAPI_CALLBACK_PRE_SCAN

```
#define SAVAPI_CALLBACK_PRE_SCAN 4
```

Triggered before the scanning begins. Can be used to create filters. For example, if we want to scan only .exe files, we install a PRE_SCAN callback. Before each file is scanned, the PRE_SCAN callback will be called. Inside our implementation of the callback, we implement the filter. If the returned code is success, the file will be scanned, otherwise it will be skipped.

Note

Currently, this callback is not triggered in client-mode.

6.14.2.6 SAVAPI_CALLBACK_PROGRESS_REPORT

```
#define SAVAPI_CALLBACK_PROGRESS_REPORT 6
```

Triggered when messages related to scan progress are available.

6.14.2.7 SAVAPI_CALLBACK_REPORT_ERROR

```
#define SAVAPI_CALLBACK_REPORT_ERROR 3
```

Triggered to report an error or a warning.

Note

Can be triggered at any time.

6.14.2.8 SAVAPI_CALLBACK_REPORT_FILE_STATUS

```
#define SAVAPI_CALLBACK_REPORT_FILE_STATUS 0
```

Triggered after a file is scanned. The callback data contains the status of the last scanned file.

6.14.2.9 SAVAPI_CALLBACK_SCAN_DETAILS_REPORT

```
#define SAVAPI_CALLBACK_SCAN_DETAILS_REPORT 8
```

Triggered when messages related to scan process details are available. The virus description url ([SAVAPI_OPTION_NOTIFY_ALERTURL](#))

6.15 SAVAPI report scan details types

Macros

- `#define SAVAPI_REPORT_ALERTURL 1`
- `#define SAVAPI_REPORT_REPAIRABLE 2`

6.15.1 Detailed Description

6.15.2 Macro Definition Documentation

6.15.2.1 SAVAPI_REPORT_ALERTURL

```
#define SAVAPI_REPORT_ALERTURL 1
```

Malware URL description

6.15.2.2 SAVAPI_REPORT_REPAIRABLE

```
#define SAVAPI_REPORT_REPAIRABLE 2
```

Malware found in the object can be repaired

6.16 SAVAPI report content types

Macros

- `#define SAVAPI_REPORT_CONTENT_IFRAME 0`

6.16.1 Detailed Description

6.16.2 Macro Definition Documentation

6.16.2.1 SAVAPI_REPORT_CONTENT_IFRAME

```
#define SAVAPI_REPORT_CONTENT_IFRAME 0
```

IFRAME URL report This option is deprecated.

6.17 SAVAPI signals

Macros

- `#define SAVAPI_SIGNAL_SCAN_ABORT 1`
Will cause the SAVAPI instance to abort scanning process as soon as possible.

6.17.1 Detailed Description

6.17.2 Macro Definition Documentation

6.17.2.1 SAVAPI_SIGNAL_SCAN_ABORT

```
#define SAVAPI_SIGNAL_SCAN_ABORT 1
```

Will cause the SAVAPI instance to abort scanning process as soon as possible.

Note

The signal have no associated specific data. When calling [SAVAPI_send_signal](#) function, "data" argument may be NULL.

Todo Add new signals as needed.

6.18 SAVAPI scan statuses

Macros

- #define [SAVAPI_SCAN_STATUS_CLEAN](#) 0
- #define [SAVAPI_SCAN_STATUS_INFECTED](#) 1
- #define [SAVAPI_SCAN_STATUS_SUSPICIOUS](#) 2
- #define [SAVAPI_SCAN_STATUS_ERROR](#) 3
- #define [SAVAPI_SCAN_STATUS_FINISHED](#) 4

6.18.1 Detailed Description

6.18.2 Macro Definition Documentation

6.18.2.1 SAVAPI_SCAN_STATUS_CLEAN

```
#define SAVAPI_SCAN_STATUS_CLEAN 0
```

Processed object is clean.

Note

This scan status is obsolete.

6.18.2.2 SAVAPI_SCAN_STATUS_ERROR

```
#define SAVAPI_SCAN_STATUS_ERROR 3
```

An error occurred during object processing

6.18.2.3 SAVAPI_SCAN_STATUS_FINISHED

```
#define SAVAPI_SCAN_STATUS_FINISHED 4
```

Object processing finished.

6.18.2.4 SAVAPI_SCAN_STATUS_INFECTED

```
#define SAVAPI_SCAN_STATUS_INFECTED 1
```

Viral code found during object processing.

6.18.2.5 SAVAPI_SCAN_STATUS_SUSPICIOUS

```
#define SAVAPI_SCAN_STATUS_SUSPICIOUS 2
```

Suspicious code found during object processing.

6.19 File types

Macros

- #define SAVAPI_FTYPE_REGULAR 4
- #define SAVAPI_FTYPE_ARCHIVE 1
- #define SAVAPI_FTYPE_IN_ARCHIVE 2

6.19.1 Detailed Description

Note

Used by the callbacks to report the type of the scanned file

6.19.2 Macro Definition Documentation

6.19.2.1 SAVAPI_FTYPE_ARCHIVE

```
#define SAVAPI_FTYPE_ARCHIVE 1
```

Known archive type

6.19.2.2 SAVAPI_FTYPE_IN_ARCHIVE

```
#define SAVAPI_FTYPE_IN_ARCHIVE 2
```

File is in an archive

6.19.2.3 SAVAPI_FTYPE_REGULAR

```
#define SAVAPI_FTYPE_REGULAR 4
```

Regular file (all files are regular)

6.20 Filename flags

Macros

- `#define SAVAPI_FLAG_LAST_FILENAME_DEFAULT 1 << 0`

The last filename is not the one reported by the engine, but a default one set by the lib.

6.20.1 Detailed Description

Note

Used by the callbacks to report the state of the last reported filename

6.20.2 Macro Definition Documentation

6.20.2.1 SAVAPI_FLAG_LAST_FILENAME_DEFAULT

```
#define SAVAPI_FLAG_LAST_FILENAME_DEFAULT 1 << 0
```

The last filename is not the one reported by the engine, but a default one set by the lib.

Note

This flag is passed to the callback data structures: [SAVAPI_PRESCAN_DATA](#), [SAVAPI_ARCHIVE_OPEN_DATA](#), [SAVAPI_FILE_STATUS_DATA](#) in the 'flags' field.

6.21 SAVAPI structures

Data Structures

- struct [SAVAPI_global_init](#)
The structure used at SAVAPI initialization.
- struct [SAVAPI_APC_global_init](#)
The structure used for initializing APC.
- struct [SAVAPI_instance_init](#)
The structure used at SAVAPI instance creation.
- struct [SAVAPI_file_info](#)
Contains data about the scanned file.
- struct [SAVAPI_malware_info](#)
Contains data about the found malware in an infected/suspicious file.
- struct [SAVAPI_pre_scan_data](#)
Contains the data sent to a prescan callback.
- struct [SAVAPI_archive_open_data](#)
Contains the data sent to a archive_open callback.
- struct [SAVAPI_key_value](#)
Generic container.
- struct [SAVAPI_file_status_data](#)
Contains the data sent to a report file status callback.
- struct [SAVAPI_OA_file_result_data](#)
Contains the data sent to an OnAccess file status callback.
- struct [SAVAPI_error_data](#)
The structure associated with report error callback.
- struct [SAVAPI_APC_REPORT_DATA](#)
Information to report when calling a [APC_set_report_info_t](#) function.
- struct [SAVAPI_apc_scan_data](#)
The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.
- struct [SAVAPI_report_progress_data](#)
The structure associated with report progress callback.
- struct [SAVAPI_iframe_url_data](#)
Structure associated with the iframe report.
- struct [SAVAPI_report_content_data](#)
The structure associated with report content callback.
- struct [SAVAPI_alert_url_data](#)
Structure associated with the ALERTURL report.
- struct [SAVAPI_repairable_data](#)
Structure associated with the REPAIRABLE report.
- struct [SAVAPI_report_scan_details_data](#)
The structure associated with report scan details callback.
- struct [SAVAPI_callback_data](#)
Structure passed by SAVAPI to a user defined callback, containing all the necessary data.
- struct [SAVAPI_simple_scan_file_data](#)
The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).
- struct [SAVAPI_simple_scan_statistics](#)
The structure contains statistics for the simple scan.
- struct [SAVAPI_simple_scan_output](#)
The structure containing the output for [SAVAPI_simple_scan](#).

- struct [SAVAPI_signal_data](#)
The structure to be passed when sending a signal.
- struct [SAVAPI_command_data](#)
The structure to be passed when sending a command.
- struct [SAVAPI_version](#)
The structure used to retrieve SAVAPI version.
- struct [SAVAPI_OA_global_init](#)
The structure used for initializing OnAccess.

Typedefs

- typedef struct [SAVAPI_global_init](#) SAVAPI_GLOBAL_INIT
The structure used at SAVAPI initialization.
- typedef enum [SAVAPI_APC_scan_mode](#) SAVAPI_APC_SCAN_MODE
Defines the APC scan mode.
- typedef struct [SAVAPI_APC_global_init](#) SAVAPI_APC_GLOBAL_INIT
The structure used for initializing APC.
- typedef struct [SAVAPI_instance_init](#) SAVAPI_INSTANCE_INIT
The structure used at SAVAPI instance creation.
- typedef struct [SAVAPI_file_info](#) SAVAPI_FILE_INFO
Contains data about the scanned file.
- typedef struct [SAVAPI_malware_info](#) SAVAPI_MALWARE_INFO
Contains data about the found malware in an infected/suspicious file.
- typedef struct [SAVAPI_pre_scan_data](#) SAVAPI_PRESCAN_DATA
Contains the data sent to a prescan callback.
- typedef struct [SAVAPI_archive_open_data](#) SAVAPI_ARCHIVE_OPEN_DATA
Contains the data sent to a archive_open callback.
- typedef struct [SAVAPI_key_value](#) SAVAPI_KEY_VALUE
Generic container.
- typedef struct [SAVAPI_file_status_data](#) SAVAPI_FILE_STATUS_DATA
Contains the data sent to a report file status callback.
- typedef struct [SAVAPI_OA_file_result_data](#) SAVAPI_OA_FILE_RESULT_DATA
Contains the data sent to an OnAccess file status callback.
- typedef struct [SAVAPI_error_data](#) SAVAPI_ERROR_DATA
The structure associated with report error callback.
- typedef struct [SAVAPI_apc_scan_data](#) SAVAPI_APC_SCAN_DATA
The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.
- typedef struct [SAVAPI_report_progress_data](#) SAVAPI_REPORT_PROGRESS_DATA
The structure associated with report progress callback.
- typedef struct [SAVAPI_iframe_url_data](#) SAVAPI_IFRAME_URL_DATA
Structure associated with the iframe report.
- typedef struct [SAVAPI_report_content_data](#) SAVAPI_REPORT_CONTENT_DATA
The structure associated with report content callback.
- typedef struct [SAVAPI_alert_url_data](#) SAVAPI_ALERT_URL_DATA
Structure associated with the ALERTURL report.
- typedef struct [SAVAPI_repairable_data](#) SAVAPI_REPAIRABLE_DATA
Structure associated with the REPAIRABLE report.
- typedef struct [SAVAPI_report_scan_details_data](#) SAVAPI_REPORT_SCAN_DETAILS_DATA
The structure associated with report scan details callback.
- typedef struct [SAVAPI_callback_data](#) SAVAPI_CALLBACK_DATA

- Structure passed by SAVAPI to a user defined callback, containing all the necessary data.*

 - typedef struct [SAVAPI_simple_scan_file_data](#) [SAVAPI_SIMPLE_SCAN_FILE_DATA](#)

The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).
- typedef struct [SAVAPI_simple_scan_statistics](#) [SAVAPI_SIMPLE_SCAN_STATISTICS](#)

The structure contains statistics for the simple scan.
- typedef struct [SAVAPI_simple_scan_output](#) [SAVAPI_SIMPLE_SCAN_OUTPUT](#)

The structure containing the output for [SAVAPI_simple_scan](#).
- typedef struct [SAVAPI_signal_data](#) [SAVAPI_SIGNAL_DATA](#)

The structure to be passed when sending a signal.
- typedef struct [SAVAPI_command_data](#) [SAVAPI_COMMAND_DATA](#)

The structure to be passed when sending a command.
- typedef struct [SAVAPI_version](#) [SAVAPI_VERSION](#)

The structure used to retrieve SAVAPI version.
- typedef enum [_SAVAPI_log_level](#) [SAVAPI_LOG_LEVEL](#)

The enumeration used to specify the SAVAPI's logging levels.
- typedef enum [SAVAPI_engine_module_type](#) [SAVAPI_ENGINE_MODULE_TYPE](#)

Defines the type of an engine module.
- typedef struct [SAVAPI_OA_global_init](#) [SAVAPI_OA_GLOBAL_INIT](#)

The structure used for initializing OnAccess.

Enumerations

- enum [SAVAPI_APC_scan_mode](#) { [SAVAPI_APC_SCAN_MODE_CHECK_ONLY](#) = 1 , [SAVAPI_APC_SCAN_MODE_FULL](#) }

Defines the APC scan mode.
- enum [SAVAPI_APC_SCAN_STAGE](#) { [SAVAPI_APC_STAGE_PRE_FILTER](#) , [SAVAPI_APC_STAGE_PRE_HASH_CHECK](#) , [SAVAPI_APC_STAGE_PRE_UPLOAD](#) , [SAVAPI_APC_STAGE_POST_SCAN](#) }

The stage in APC scanning in which the [SAVAPI_CALLBACK_APC_SCAN](#) callback was called.
- enum [SAVAPI_APC_SCAN_ANSWER](#) { [SAVAPI_APC_ANSWER_CLEAN](#) = 1 , [SAVAPI_APC_ANSWER_INFECTED](#) }

Scan answers that can be provided by the user for a reported file.
- enum [_SAVAPI_log_level](#) { [SAVAPI_LOG_DEBUG](#) = 0 , [SAVAPI_LOG_INFO](#) , [SAVAPI_LOG_WARNING](#) , [SAVAPI_LOG_ALERT](#) , [SAVAPI_LOG_ERROR](#) }

The enumeration used to specify the SAVAPI's logging levels.
- enum [SAVAPI_engine_module_type](#) { [SAVAPI_ENGINE_MODULE_AVE](#) = 0 , [SAVAPI_ENGINE_MODULE_VDF](#) }

Defines the type of an engine module.

6.21.1 Detailed Description

6.21.2 Typedef Documentation

6.21.2.1 SAVAPI_ALERT_URL_DATA

```
typedef struct SAVAPI_alert_url_data SAVAPI_ALERT_URL_DATA
```

Structure associated with the ALERTURL report.

6.21.2.2 SAVAPI_APC_GLOBAL_INIT

```
typedef struct SAVAPI_apc_global_init SAVAPI_APC_GLOBAL_INIT
```

The structure used for initializing APC.

6.21.2.3 SAVAPI_APC_SCAN_DATA

```
typedef struct SAVAPI_apc_scan_data SAVAPI_APC_SCAN_DATA
```

The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.

6.21.2.4 SAVAPI_APC_SCAN_MODE

```
typedef enum SAVAPI_apc_scan_mode SAVAPI_APC_SCAN_MODE
```

Defines the APC scan mode.

6.21.2.5 SAVAPI_ARCHIVE_OPEN_DATA

```
typedef struct SAVAPI_archive_open_data SAVAPI_ARCHIVE_OPEN_DATA
```

Contains the data sent to a `archive_open` callback.

6.21.2.6 SAVAPI_CALLBACK_DATA

```
typedef struct SAVAPI_callback_data SAVAPI_CALLBACK_DATA
```

Structure passed by SAVAPI to a user defined callback, containing all the necessary data.

6.21.2.7 SAVAPI_COMMAND_DATA

```
typedef struct SAVAPI_command_data SAVAPI_COMMAND_DATA
```

The structure to be passed when sending a command.

6.21.2.8 SAVAPI_ENGINE_MODULE_TYPE

```
typedef enum SAVAPI_engine_module_type SAVAPI_ENGINE_MODULE_TYPE
```

Defines the type of an engine module.

6.21.2.9 SAVAPI_ERROR_DATA

```
typedef struct SAVAPI_error_data SAVAPI_ERROR_DATA
```

The structure associated with report error callback.

The callback is triggered each time an error occurred on scanning process (an I/O error for instance). Also the callback can be called if warnings or infos reports during scanning are activated.

Note

See [SAVAPI_CALLBACK_REPORT_ERROR](#)

6.21.2.10 SAVAPI_FILE_INFO

```
typedef struct SAVAPI_file_info SAVAPI_FILE_INFO
```

Contains data about the scanned file.

6.21.2.11 SAVAPI_FILE_STATUS_DATA

```
typedef struct SAVAPI_file_status_data SAVAPI_FILE_STATUS_DATA
```

Contains the data sent to a report file status callback.

Note

See [SAVAPI_CALLBACK_REPORT_FILE_STATUS](#)

6.21.2.12 SAVAPI_GLOBAL_INIT

```
typedef struct SAVAPI_global_init SAVAPI_GLOBAL_INIT
```

The structure used at SAVAPI initialization.

6.21.2.13 SAVAPI_IFRAME_URL_DATA

```
typedef struct SAVAPI_iframe_url_data SAVAPI_IFRAME_URL_DATA
```

Structure associated with the iframe report.

Note

This structure is deprecated.

6.21.2.14 SAVAPI_INSTANCE_INIT

```
typedef struct SAVAPI_instance_init SAVAPI_INSTANCE_INIT
```

The structure used at SAVAPI instance creation.

6.21.2.15 SAVAPI_KEY_VALUE

```
typedef struct SAVAPI_key_value SAVAPI_KEY_VALUE
```

Generic container.

This kind of container is very useful in case of need to store many options. It offers a very elegant encapsulation and a very high flexibility (the user will not know how data will be stored).

The elements from container are accessed using a key ([SAVAPI_key_value::id](#) member). The element is accessed through [SAVAPI_key_value::value](#) member and its type through [SAVAPI_key_value::type](#) member

6.21.2.16 SAVAPI_LOG_LEVEL

```
typedef enum _SAVAPI_log_level SAVAPI_LOG_LEVEL
```

The enumeration used to specify the SAVAPI's logging levels.

6.21.2.17 SAVAPI_MALWARE_INFO

```
typedef struct SAVAPI_malware_info SAVAPI_MALWARE_INFO
```

Contains data about the found malware in an infected/suspicious file.

6.21.2.18 SAVAPI_OA_FILE_RESULT_DATA

```
typedef struct SAVAPI_OA_file_result_data SAVAPI_OA_FILE_RESULT_DATA
```

Contains the data sent to an OnAccess file status callback.

Note

See [SAVAPI_CALLBACK_OA_FILE_RESULT](#)

6.21.2.19 SAVAPI_OA_GLOBAL_INIT

```
typedef struct SAVAPI_OA_global_init SAVAPI_OA_GLOBAL_INIT
```

The structure used for initializing OnAccess.

Note

Only supported on Windows.

6.21.2.20 SAVAPI_PRESCAN_DATA

```
typedef struct SAVAPI_pre_scan_data SAVAPI_PRESCAN_DATA
```

Contains the data sent to a prescan callback.

6.21.2.21 SAVAPI_REPAIRABLE_DATA

```
typedef struct SAVAPI_repairable_data SAVAPI_REPAIRABLE_DATA
```

Structure associated with the REPAIRABLE report.

6.21.2.22 SAVAPI_REPORT_CONTENT_DATA

```
typedef struct SAVAPI_report_content_data SAVAPI_REPORT_CONTENT_DATA
```

The structure associated with report content callback.

Note

This structure is deprecated.

6.21.2.23 SAVAPI_REPORT_PROGRESS_DATA

```
typedef struct SAVAPI_report_progress_data SAVAPI_REPORT_PROGRESS_DATA
```

The structure associated with report progress callback.

6.21.2.24 SAVAPI_REPORT_SCAN_DETAILS_DATA

```
typedef struct SAVAPI_report_scan_details_data SAVAPI_REPORT_SCAN_DETAILS_DATA
```

The structure associated with report scan details callback.

6.21.2.25 SAVAPI_SIGNAL_DATA

```
typedef struct SAVAPI_signal_data SAVAPI_SIGNAL_DATA
```

The structure to be passed when sending a signal.

6.21.2.26 SAVAPI_SIMPLE_SCAN_FILE_DATA

```
typedef struct SAVAPI_simple_scan_file_data SAVAPI_SIMPLE_SCAN_FILE_DATA
```

The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).

6.21.2.27 SAVAPI_SIMPLE_SCAN_OUTPUT

```
typedef struct SAVAPI_simple_scan_output SAVAPI_SIMPLE_SCAN_OUTPUT
```

The structure containing the output for [SAVAPI_simple_scan](#).

Note

Memory management is done by SAVAPI so there is no need to allocate or free the structure.

6.21.2.28 SAVAPI_SIMPLE_SCAN_STATISTICS

```
typedef struct SAVAPI_simple_scan_statistics SAVAPI_SIMPLE_SCAN_STATISTICS
```

The structure contains statistics for the simple scan.

6.21.2.29 SAVAPI_VERSION

```
typedef struct SAVAPI_version SAVAPI_VERSION
```

The structure used to retrieve SAVAPI version.

6.21.3 Enumeration Type Documentation

6.21.3.1 _SAVAPI_log_level

```
enum _SAVAPI_log_level
```

The enumeration used to specify the SAVAPI's logging levels.

Enumerator

SAVAPI_LOG_DEBUG	Low level (debug, trace) messages. This the service MESSAGE level equivalent
SAVAPI_LOG_INFO	informative messages
SAVAPI_LOG_WARNING	warning messages
SAVAPI_LOG_ALERT	alert messages (i.e. malware found or any other alert)
SAVAPI_LOG_ERROR	error messages

6.21.3.2 SAVAPI_APC_SCAN_ANSWER

enum [SAVAPI_APC_SCAN_ANSWER](#)

Scan answers that can be provided by the user for a reported file.

Enumerator

SAVAPI_APC_ANSWER_CLEAN	The file is clean
SAVAPI_APC_ANSWER_INFECTED	The file is infected

6.21.3.3 SAVAPI_APC_scan_mode

enum [SAVAPI_APC_scan_mode](#)

Defines the APC scan mode.

Enumerator

SAVAPI_APC_SCAN_MODE_CHECK_ONLY	APC checks only hashes
SAVAPI_APC_SCAN_MODE_FULL	Full APC functionality (hash checking and file uploads)

6.21.3.4 SAVAPI_APC_SCAN_STAGE

enum [SAVAPI_APC_SCAN_STAGE](#)

The stage in APC scanning in which the [SAVAPI_CALLBACK_APC_SCAN](#) callback was called.

Enumerator

SAVAPI_APC_STAGE_PRE_FILTER	The file was not yet checked with APC
SAVAPI_APC_STAGE_PRE_HASH_CHECK	The file has passed the APC filter, but has not yet been checked
SAVAPI_APC_STAGE_PRE_UPLOAD	The hash of the file was checked with APC, but the file was not uploaded
SAVAPI_APC_STAGE_POST_SCAN	APC scan has finished for this file

6.21.3.5 SAVAPI_engine_module_type

enum [SAVAPI_engine_module_type](#)

Defines the type of an engine module.

Enumerator

SAVAPI_ENGINE_MODULE_AVE	
SAVAPI_ENGINE_MODULE_VDF	

6.22 SAVAPI typedefs

Typedefs

- typedef void * [SAVAPI_FD](#)
SAVAPI instance handle.
- typedef [SAVAPI_STATUS](#)(* [APC_set_report_info_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_APC_REPORT_DATA](#) *data)
Type of function that is called by the user to report findings regarding a scanned file.
- typedef int(* [SAVAPI_CALLBACK](#)) ([SAVAPI_CALLBACK_DATA](#) *data)
SAVAPI callback function pointer definition.
- typedef void(* [SAVAPI_LOG_CALLBACK](#)) ([SAVAPI_LOG_LEVEL](#) log_level, const [SAVAPI_TCHAR](#) *message, void *user_data)
SAVAPI callback for logging.
- typedef int(* [SAVAPI_ENGINE_MODULE_CALLBACK](#)) (const [SAVAPI_TCHAR](#) *name, [SAVAPI_ENGINE_MODULE_TYPE](#) type, void *user_data)
Callback function used to return one engine single module.
- typedef int(* [SAVAPI_OA_INSTANCE_CALLBACK](#)) ([SAVAPI_FD](#) savapi_fd)
Callback function used to configure the instances created by OnAccess.

6.22.1 Detailed Description

6.22.2 Typedef Documentation

6.22.2.1 APC_set_report_info_t

```
typedef SAVAPI\_STATUS(* APC\_set\_report\_info\_t) (SAVAPI\_FD savapi_fd, SAVAPI\_APC\_REPORT\_DATA *data)
```

Type of function that is called by the user to report findings regarding a scanned file.

Parameters

<i>savapi_fd</i>	[IN]: Handle to the savapi instance.
<i>data</i>	[IN]: Pointer to the structure containing the report data.

Return values

<i>SAVAPI_S_OK</i>	if successful
<i>SAVAPI_E_INVALID_PARAMETER</i>	if any of the parameters are invalid
<i>SAVAPI_E_NO_MEMORY</i>	if no memory is left for allocations

6.22.2.2 SAVAPI_CALLBACK

```
typedef int (* SAVAPI_CALLBACK) (SAVAPI_CALLBACK_DATA *data)
```

SAVAPI callback function pointer definition.

Parameters

<i>data</i>	[IN]: Pointer to the structure containing the callback data.
-------------	--

6.22.2.3 SAVAPI_ENGINE_MODULE_CALLBACK

```
typedef int (* SAVAPI_ENGINE_MODULE_CALLBACK) (const SAVAPI_TCHAR *name, SAVAPI_ENGINE_MODULE_TYPE type, void *user_data)
```

Callback function used to return one engine single module.

Parameters

<i>name</i>	: The name of the module
<i>type</i>	: The type of the returned module
<i>user_data</i>	: Pointer to the user-specific data

6.22.2.4 SAVAPI_FD

```
typedef void* SAVAPI_FD
```

SAVAPI instance handle.

6.22.2.5 SAVAPI_LOG_CALLBACK

```
typedef void (* SAVAPI_LOG_CALLBACK) (SAVAPI_LOG_LEVEL log_level, const SAVAPI_TCHAR *message, void *user_data)
```

SAVAPI callback for logging.

Parameters

<i>log_level</i>	[IN]: The log level for the given message
<i>message</i>	[IN]: The message to be logged
<i>user_data</i>	[IN]: The user context

Returns

Nothing

6.22.2.6 SAVAPI_OA_INSTANCE_CALLBACK

```
typedef int (* SAVAPI_OA_INSTANCE_CALLBACK) (SAVAPI_FD savapi_fd)
```

Callback function used to configure the instances created by OnAccess.

Parameters

<i>savapi_↔ _fd</i>	: Handle to savapi instance
-------------------------	-----------------------------

6.23 SAVAPI function pointers

Handle types for exported SAVAPI functions.

Modules

- [SAVAPI hex2bin function pointers](#)
Handle types for exported SAVAPI hex2bin functions.
- [SAVAPI main function pointers](#)
Handle types for exported SAVAPI main functions.
- [SAVAPI STCHAR function pointers](#)
Handle types for exported SAVAPI STCHAR functions.

6.23.1 Detailed Description

Handle types for exported SAVAPI functions.

6.24 SAVAPI main function pointers

Handle types for exported SAVAPI main functions.

Typedefs

- typedef SAVAPI_STATUS(* SAVAPI_set_log_callback_t) (SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL min_level, void *user_data)
- typedef SAVAPI_STATUS(* SAVAPI_initialize_t) (SAVAPI_GLOBAL_INIT *savapi_init)
- typedef void(* SAVAPI_set_quickload_init_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_uninitialize_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_get_version_t) (SAVAPI_VERSION *version)
- typedef SAVAPI_STATUS(* SAVAPI_engine_versions_get_t) (SAVAPI_VERSION *ave_version, SAVAPI_VERSION *avpack_version, SAVAPI_VERSION *vdf_version)
- typedef SAVAPI_STATUS(* SAVAPI_APC_get_version_t) (SAVAPI_VERSION *apc_version)
- typedef SAVAPI_STATUS(* SAVAPI_create_instance_t) (SAVAPI_INSTANCE_INIT *init, SAVAPI_FD *savapi_fd)
- typedef SAVAPI_STATUS(* SAVAPI_release_instance_t) (SAVAPI_FD *savapi_fd)
- typedef SAVAPI_STATUS(* SAVAPI_set_user_data_t) (SAVAPI_FD savapi_fd, void *user_data)
- typedef SAVAPI_STATUS(* SAVAPI_get_user_data_t) (SAVAPI_FD savapi_fd, void **user_data)
- typedef SAVAPI_STATUS(* SAVAPI_is_running_ex_t) (const SAVAPI_TCHAR *hostname, unsigned int port)
- typedef SAVAPI_STATUS(* SAVAPI_register_callback_t) (SAVAPI_FD savapi_fd, unsigned int callback_id, SAVAPI_CALLBACK callback)
- typedef SAVAPI_STATUS(* SAVAPI_unregister_callback_t) (SAVAPI_FD savapi_fd, unsigned int callback_id, SAVAPI_CALLBACK callback)
- typedef SAVAPI_STATUS(* SAVAPI_scan_t) (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name)
- typedef SAVAPI_STATUS(* SAVAPI_simple_scan_t) (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name, SAVAPI_SIMPLE_SCAN_OUTPUT *output)
- typedef SAVAPI_STATUS(* SAVAPI_set_t) (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer)
- typedef SAVAPI_STATUS(* SAVAPI_get_t) (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer, SAVAPI_SIZE_T *buffer_size)
- typedef SAVAPI_STATUS(* SAVAPI_send_signal_t) (SAVAPI_FD savapi_fd, unsigned int signal_id, SAVAPI_SIGNAL_DATA *data)
- typedef SAVAPI_STATUS(* SAVAPI_set_fops_t) (SAVAPI_FD savapi_fd, void *fops_pointer, void *fops↵ context)
- typedef SAVAPI_STATUS(* SAVAPI_get_fops_t) (SAVAPI_FD savapi_fd, void **fops_pointer, void **fops↵ _context)
- typedef void(* SAVAPI_free_t) (void **ptr)
- typedef SAVAPI_STATUS(* SAVAPI_reload_engine_ex_t) (const SAVAPI_GLOBAL_INIT *global_init)
- typedef SAVAPI_STATUS(* SAVAPI_extract_malware_names_t) (const SAVAPI_TCHAR *dir_path)
- typedef SAVAPI_STATUS(* SAVAPI_engine_modules_get_t) (const SAVAPI_GLOBAL_INIT *init, SAVAPI_ENGINE_MODULE_CALLBACK module_func, void *user_data)
- typedef SAVAPI_STATUS(* SAVAPI_global_set_t) (SAVAPI_GLOBAL_OPTION option_id, SAVAPI_TCHAR *buffer)
- typedef SAVAPI_STATUS(* SAVAPI_APC_initialize_t) (SAVAPI_APC_GLOBAL_INIT *savapi_apc_init)
- typedef SAVAPI_STATUS(* SAVAPI_APC_uninitialize_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_initialize_t) (SAVAPI_OA_GLOBAL_INIT *savapi_oa_init)
- typedef SAVAPI_STATUS(* SAVAPI_OA_uninitialize_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_create_instances_t) (SAVAPI_OA_INSTANCE_CALLBACK init↵ func, SAVAPI_OA_INSTANCE_CALLBACK uninit_func)
- typedef SAVAPI_STATUS(* SAVAPI_OA_start_scan_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_stop_scan_t) ()

6.24.1 Detailed Description

Handle types for exported SAVAPI main functions.

6.24.2 Typedef Documentation

6.24.2.1 SAVAPI_APC_get_version_t

```
typedef SAVAPI_STATUS(* SAVAPI_APC_get_version_t) (SAVAPI_VERSION *apc_version)
```

6.24.2.2 SAVAPI_APC_initialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_APC_initialize_t) (SAVAPI_APC_GLOBAL_INIT *savapi_apc_init)
```

6.24.2.3 SAVAPI_APC_uninitialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_APC_uninitialize_t) ()
```

6.24.2.4 SAVAPI_create_instance_t

```
typedef SAVAPI_STATUS(* SAVAPI_create_instance_t) (SAVAPI_INSTANCE_INIT *init, SAVAPI_FD *savapi←  
_fd)
```

6.24.2.5 SAVAPI_engine_modules_get_t

```
typedef SAVAPI_STATUS(* SAVAPI_engine_modules_get_t) (const SAVAPI_GLOBAL_INIT *init, SAVAPI_ENGINE_MODULE_CAL  
module_func, void *user_data)
```

6.24.2.6 SAVAPI_engine_versions_get_t

```
typedef SAVAPI_STATUS(* SAVAPI_engine_versions_get_t) (SAVAPI_VERSION *ave_version, SAVAPI_VERSION  
*avpack_version, SAVAPI_VERSION *vdf_version)
```

6.24.2.7 SAVAPI_extract_malware_names_t

```
typedef SAVAPI_STATUS(* SAVAPI_extract_malware_names_t) (const SAVAPI_TCHAR *dir_path)
```

6.24.2.8 SAVAPI_free_t

```
typedef void(* SAVAPI_free_t) (void **ptr)
```

6.24.2.9 SAVAPI_get_fops_t

```
typedef SAVAPI_STATUS(* SAVAPI_get_fops_t) (SAVAPI_FD savapi_fd, void **fops_pointer, void **fops_context)
```

6.24.2.10 SAVAPI_get_t

```
typedef SAVAPI_STATUS(* SAVAPI_get_t) (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer, SAVAPI_SIZE_T *buffer_size)
```

6.24.2.11 SAVAPI_get_user_data_t

```
typedef SAVAPI_STATUS(* SAVAPI_get_user_data_t) (SAVAPI_FD savapi_fd, void **user_data)
```

6.24.2.12 SAVAPI_get_version_t

```
typedef SAVAPI_STATUS(* SAVAPI_get_version_t) (SAVAPI_VERSION *version)
```

6.24.2.13 SAVAPI_global_set_t

```
typedef SAVAPI_STATUS(* SAVAPI_global_set_t) (SAVAPI_GLOBAL_OPTION option_id, SAVAPI_TCHAR *buffer)
```


6.24.2.14 SAVAPI_initialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_initialize_t) (SAVAPI_GLOBAL_INIT *savapi_init)
```

6.24.2.15 SAVAPI_is_running_ex_t

```
typedef SAVAPI_STATUS(* SAVAPI_is_running_ex_t) (const SAVAPI_TCHAR *hostname, unsigned int port)
```

6.24.2.16 SAVAPI_OA_create_instances_t

```
typedef SAVAPI_STATUS(* SAVAPI_OA_create_instances_t) (SAVAPI_OA_INSTANCE_CALLBACK init_func, SAVAPI_OA_INSTANCE_CALLBACK uninit_func)
```

6.24.2.17 SAVAPI_OA_initialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_OA_initialize_t) (SAVAPI_OA_GLOBAL_INIT *savapi_oa_init)
```

6.24.2.18 SAVAPI_OA_start_scan_t

```
typedef SAVAPI_STATUS(* SAVAPI_OA_start_scan_t) ()
```

6.24.2.19 SAVAPI_OA_stop_scan_t

```
typedef SAVAPI_STATUS(* SAVAPI_OA_stop_scan_t) ()
```

6.24.2.20 SAVAPI_OA_uninitialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_OA_uninitialize_t) ()
```

6.24.2.21 SAVAPI_register_callback_t

```
typedef SAVAPI_STATUS(* SAVAPI_register_callback_t) (SAVAPI_FD savapi_fd, unsigned int callback_id, SAVAPI_CALLBACK callback)
```

6.24.2.22 SAVAPI_release_instance_t

```
typedef SAVAPI_STATUS(* SAVAPI_release_instance_t) (SAVAPI_FD *savapi_fd)
```

6.24.2.23 SAVAPI_reload_engine_ex_t

```
typedef SAVAPI_STATUS(* SAVAPI_reload_engine_ex_t) (const SAVAPI_GLOBAL_INIT *global_init)
```

6.24.2.24 SAVAPI_scan_t

```
typedef SAVAPI_STATUS(* SAVAPI_scan_t) (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name)
```

6.24.2.25 SAVAPI_send_signal_t

```
typedef SAVAPI_STATUS(* SAVAPI_send_signal_t) (SAVAPI_FD savapi_fd, unsigned int signal_id, SAVAPI_SIGNAL_DATA *data)
```

6.24.2.26 SAVAPI_set_fops_t

```
typedef SAVAPI_STATUS(* SAVAPI_set_fops_t) (SAVAPI_FD savapi_fd, void *fops_pointer, void *fops_context)
```

6.24.2.27 SAVAPI_set_log_callback_t

```
typedef SAVAPI_STATUS(* SAVAPI_set_log_callback_t) (SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL min_level, void *user_data)
```

6.24.2.28 SAVAPI_set_quickload_init_t

```
typedef void(* SAVAPI_set_quickload_init_t) ()
```

6.24.2.29 SAVAPI_set_t

```
typedef SAVAPI_STATUS(* SAVAPI_set_t) (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer)
```

6.24.2.30 SAVAPI_set_user_data_t

```
typedef SAVAPI_STATUS(* SAVAPI_set_user_data_t) (SAVAPI_FD savapi_fd, void *user_data)
```

6.24.2.31 SAVAPI_simple_scan_t

```
typedef SAVAPI_STATUS(* SAVAPI_simple_scan_t) (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name, SAVAPI_SIMPLE_SCAN_OUTPUT *output)
```

6.24.2.32 SAVAPI_uninitialize_t

```
typedef SAVAPI_STATUS(* SAVAPI_uninitialize_t) ()
```

6.24.2.33 SAVAPI_unregister_callback_t

```
typedef SAVAPI_STATUS(* SAVAPI_unregister_callback_t) (SAVAPI_FD savapi_fd, unsigned int callback←_id, SAVAPI_CALLBACK callback)
```

6.25 SAVAPI functions

Functions

- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_log_callback** (SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL min_level, void *user_data)
Sets the SAVAPI logging function.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_initialize** (SAVAPI_GLOBAL_INIT *savapi_init)
SAVAPI initialization function Initializes the SAVAPI library, according to the parameters specified in the initialization structure. It should be called once per process, but it may be called several times per process only if SAVAPI_uninitialize() has been called in between. The latter case is useful when initializing SAVAPI in 'quick load' mode (SAVAPI_set_quickload_init()), and then uninitializing and reinitializing SAVAPI in 'normal mode' for scanning purposes.
- **void SAVAPI_EXP SAVAPI_set_quickload_init** ()
Sets the SAVAPI initialization mode to 'quick load' Recommended if only component versions are needed.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_uninitialize** ()
SAVAPI uninitialization function Uninitializes the SAVAPI library, cleaning up all used resources. Once called, all subsequent SAVAPI calls will fail with SAVAPI_E_NOT_INITIALIZED error code.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_initialize** (SAVAPI_APC_GLOBAL_INIT *savapi_apc_init)
SAVAPI initialization function for APC component Initializes the APC library, according to the parameters specified in the initialization structure.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_uninitialize** ()
SAVAPI uninitialization function for APC component Uninitializes the APC library, cleaning up all used resources.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_get_version** (SAVAPI_VERSION *version)
Returns the APC version.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_version** (SAVAPI_VERSION *version)
Returns SAVAPI library version.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_create_instance** (SAVAPI_INSTANCE_INIT *init, SAVAPI_FD *savapi_fd)
SAVAPI factory function The function opens a connection to the SAVAPI daemon for client-mode, or, for library mode, it creates a new SAVAPI instance.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_release_instance** (SAVAPI_FD *savapi_fd)
Destroys a SAVAPI handler, previously created with SAVAPI_create_instance. The function closes the connection to the SAVAPI daemon for client-mode, or, for library mode, it releases the SAVAPI instance.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_user_data** (SAVAPI_FD savapi_fd, void *user_data)
Sets user specific data. This functions sets user data that will be returned untouched as user_data member of SAVAPI_CALLBACK_DATA structure.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_user_data** (SAVAPI_FD savapi_fd, void **user_data)
Gets user specific data. This functions gets the user data set by SAVAPI_set_user_data.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running** ()
Determines if the SAVAPI daemon is running The library must be initialized with the proper daemon connection parameters for this function to run correctly.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running_ex** (const SAVAPI_TCHAR *hostname, unsigned int port)
Determines if the SAVAPI daemon is running on the specified interface (hostname and port)
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_register_callback** (SAVAPI_FD savapi_fd, unsigned int callback_id, SAVAPI_CALLBACK callback)
Registers a client defined callback.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_unregister_callback** (SAVAPI_FD savapi_fd, unsigned int callback_id, SAVAPI_CALLBACK callback)
Unregisters a previously registered client defined callback.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_scan** (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name)
Starts a scanning process. During the scan operation the registered callbacks may be triggered.

- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_simple_scan** (SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name, SAVAPI_SIMPLE_SCAN_OUTPUT *output)
Starts a scanning process during which no callbacks will be triggered.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_set** (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer)
Sets SAVAPI individual settings.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_get** (SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR *buffer, SAVAPI_SIZE_T *buffer_size)
Reads SAVAPI settings.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_dynamic_detect** (SAVAPI_TCHAR *type, int *id)
Retrieve the various types that can be detected (and dynamically turned on/off).
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_send_signal** (SAVAPI_FD savapi_fd, unsigned int signal_id, SAVAPI_SIGNAL_DATA *data)
Sends a signal to a specific SAVAPI instance The SAVAPI_scan may take a long amount of time to finish scanning its target and in some situations a forced abort would be desirable. In these kind of situations, SAVAPI_send_signal may help by sending signals to a running SAVAPI instance (SAVAPI_SIGNAL_SCAN_ABORT for instance).
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_fops** (SAVAPI_FD savapi_fd, void *fops_pointer, void *fops ↵ context)
Specify the new fops who will be used by the engine.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_fops** (SAVAPI_FD savapi_fd, void **fops_pointer, void **fops_context)
Get the fops which is currently used by the engine.
- **void SAVAPI_EXP SAVAPI_free** (void **ptr)
Frees the memory space pointed to by ptr.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine** ()
Reloads the engine from the location given at global initialization.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_versions_get** (SAVAPI_VERSION *ave_version, SAVAPI_VERSION *avpack_version, SAVAPI_VERSION *vdf_version)
Retrieves the engine versions: ave, avpack and vdf.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine_ex** (const SAVAPI_GLOBAL_INIT *global_init)
Reloads the engine from the specified location.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_extract_malware_names** (const SAVAPI_TCHAR *dir_path)
Extracts the malware names from memory to disk. The purpose of the function is to reduce the amount of memory (RAM) used by the SAVAPI Library. The basic idea is to unload the malware names (which are only needed in case of alerts) from memory and dump them to disk.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_modules_get** (const SAVAPI_GLOBAL_INIT *init, SAVAPI_ENGINE_MODULE_CALLBACK module_func, void *user_data)
Retrieves the engine components through the provided callback.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_global_set** (SAVAPI_GLOBAL_OPTION optionId, SAVAPI_TCHAR *buffer)
Sets SAVAPI global settings.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_initialize** (SAVAPI_OA_GLOBAL_INIT *oa_global_init)
SAVAPI initialization function for OnAccess component Initializes the OnAccess library, according to the parameters specified in the initialization structure.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_uninitialize** ()
SAVAPI uninitialization function for OnAccess component Uninitializes the OnAccess library, cleaning up all used resources.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_create_instances** (SAVAPI_OA_INSTANCE_CALLBACK init↵_func, SAVAPI_OA_INSTANCE_CALLBACK uninit_func)
Creates automatically the number of SAVAPI instances specified in SAVAPI_OA_initialize. For each instance created, the callback_func function will be called with the instance to be configured.
- **SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_start_scan** ()
Starts the real-time on-access scanning process. During the scan operation the instance registered callbacks may be triggered.

- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_stop_scan \(\)](#)
Stops the real-time on-access scanning process.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_FPC_disable_preinit \(\)](#)
Disables the pre-initialization of FPC inside the [SAVAPI_initialize\(\)](#) function.

6.25.1 Detailed Description

6.25.2 Function Documentation

6.25.2.1 SAVAPI_APC_get_version()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_get_version (
    SAVAPI_VERSION * version )
```

Returns the APC version.

Parameters

<i>version</i>	[OUT]: Pointer to the structure where to store the result
----------------	---

Returns

SAVAPI_S_OK for success or an error code otherwise

Note

Function returns [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

6.25.2.2 SAVAPI_APC_initialize()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_initialize (
    SAVAPI_APC_GLOBAL_INIT * savapi_apc_init )
```

SAVAPI initialization function for APC component Initializes the APC library, according to the parameters specified in the initialization structure.

Parameters

<i>savapi_apc_init</i>	[IN]: A pointer to the initialization structure, which must be filled with the proper values for initialization
------------------------	---

Returns

SAVAPI_S_OK on success or an error otherwise

Note

APC is an optional component of SAVAPI which allows scanning in the cloud. By calling this function you enable the APC functionality

This function must be called a single time per process

This function must be called after [SAVAPI_initialize\(\)](#), before creating any instance

The [SAVAPI_APC_GLOBAL_INIT](#) structure is copied internally and it is not longer needed by SAVAPI after calling this function.

This function returns [SAVAPI_E_NOT_SUPPORTED](#) in client mode.

6.25.2.3 SAVAPI_APC_uninitialize()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_uninitialize ( )
```

SAVAPI uninitialization function for APC component Unitializes the APC library, cleaning up all used resources.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

APC is an optional component of SAVAPI which allows scanning in the cloud. By calling this function you disable the APC functionality

This function must be called a single time per process

This function must be called after all SAVAPI instances are released and before [SAVAPI_uninitialize\(\)](#)

This function returns [SAVAPI_E_NOT_SUPPORTED](#) in client mode.

6.25.2.4 SAVAPI_create_instance()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_create_instance (
    SAVAPI_INSTANCE_INIT * init,
    SAVAPI_FD * savapi_fd )
```

SAVAPI factory function The function opens a connection to the SAVAPI daemon for client-mode, or, for library mode, it creates a new SAVAPI instance.

Parameters

<i>init</i>	[IN]: Pointer to a structure containing all the initialization data needed to create a SAVAPI instance.
<i>savapi↔ _fd</i>	[OUT]: Handle to the SAVAPI instance. To be used in all the subsequent SAVAPI calls

Returns

SAVAPI_S_OK on success or an error otherwise

6.25.2.5 SAVAPI_engine_modules_get()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_modules_get (
    const SAVAPI_GLOBAL_INIT * init,
    SAVAPI_ENGINE_MODULE_CALLBACK module_func,
    void * user_data )
```

Retrieves the engine components through the provided callback.

Parameters

<i>init</i>	[IN]: A pointer to the initialization structure
<i>module_func</i>	[IN]: Pointer to the function to be called for every engine module
<i>user_data</i>	[IN]: Pointer to the user-defined data

Returns

SAVAPI_S_OK on success or an error otherwise

Note

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode.

6.25.2.6 SAVAPI_engine_versions_get()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_versions_get (
    SAVAPI_VERSION * ave_version,
    SAVAPI_VERSION * avpack_version,
    SAVAPI_VERSION * vdf_version )
```

Retrieves the engine versions: ave, avpack and vdf.

Parameters

<i>ave_version</i>	This structure will contain the AVE version numbers
<i>avpack_version</i>	This structure will contain the AV-PACK version numbers
<i>vdf_version</i>	This structure will contain the VDF version numbers

Returns

SAVAPI_S_OK or an error code otherwise

Note

Function returns [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

6.25.2.7 SAVAPI_extract_malware_names()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_extract_malware_names (
    const SAVAPI_TCHAR * dir_path )
```

Extracts the malware names from memory to disk. The purpose of the function is to reduce the amount of memory (RAM) used by the SAVAPI Library. The basic idea is to unload the malware names (which are only needed in case of alerts) from memory and dump them to disk.

Parameters

<i>dir_path</i>	[IN]: The directory where the file containing the malware names information will be created. If set to NULL, the function will use the system temporary folder.
-----------------	---

Returns

SAVAPI_S_OK on success or an error otherwise

Note

The function will create, in the chosen folder (the *dir_path* or the system temporary folder), a file with the following naming scheme: 'AV-malware-names-<process-PID>-<6 random chars>'. In order to retrieve the path to the file, the [SAVAPI_get](#) function along with the [SAVAPI_OPTION_MALWARE_NAMES_FILE](#) option should be used. The file will be kept opened by the SAVAPI Library as long as the current engine is in use and will be closed and removed afterwards (i.e. after a successfully call to one of [SAVAPI_reload_engine](#), [SAVAPI_reload_engine_ex](#) and [SAVAPI_uninitialize](#) functions). Thus after a call to the [SAVAPI_reload_engine](#) or [SAVAPI_reload_engine_ex](#), this function should be called again in order to extract the new engine's malware names.

It has to be ensured that the Library has the appropriate permissions to create the file and that there is enough free disk space - around 100 Mb.

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode.

Warning

Calling this function more than once for a single engine is forbidden and the [SAVAPI_E_BUSY](#) error will be returned instead!

6.25.2.8 SAVAPI_FPC_disable_preinit()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_FPC_disable_preinit ( )
```

Disables the pre-initialization of FPC inside the [SAVAPI_initialize\(\)](#) function.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

This function must be called before [SAVAPI_initialize\(\)](#)

If this function is called, FPC cannot be enabled afterwards using [SAVAPI_OPTION_FPC](#) option

This function returns [SAVAPI_E_NOT_SUPPORTED](#) in client mode

6.25.2.9 SAVAPI_free()

```
void SAVAPI_EXP SAVAPI_free (
    void ** ptr )
```

Frees the memory space pointed to by ptr.

Parameters

<i>ptr</i>	[IN/OUT]: Pointer who will become free and null
------------	---

Returns

Nothing

6.25.2.10 SAVAPI_get()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_get (
    SAVAPI_FD savapi_fd,
    SAVAPI_OPTION option_id,
    SAVAPI_TCHAR * buffer,
    SAVAPI_SIZE_T * buffer_size )
```

Reads SAVAPI settings.

Parameters

<i>savapi_fd</i>	[IN]: Handle of the SAVAPI instance
<i>option_id</i>	[IN]: The id of the option to be retrieved
<i>buffer</i>	[OUT]: Buffer allocated by caller which will store the result of a successful get as a NULL terminated string. If the buffer is NULL and the other parameters are valid, the function will set the needed buffer-size and return SAVAPI_S_OK
<i>buffer_size</i>	[IN/OUT]: Specifies the size, given in SAVAPI_TCHAR characters, of the buffer argument. If the buffer is not large enough, upon return it will contain the needed size (including the terminator). If it's equal or larger than the needed size, it will remain unchanged.

Returns

SAVAPI_S_OK on success or an error otherwise

6.25.2.11 SAVAPI_get_dynamic_detect()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_dynamic_detect (
    SAVAPI_TCHAR * type,
    int * id )
```

Retrieve the various types that can be detected (and dynamically turned on/off).

Parameters

<i>type</i>	[IN]: The type that should be detected (current values: AD-WARE,ADSPY,APPL,BDC,DIAL,GAME,HIDDENEXT,JOKE,PCK,PFS,PHISH,PUA,SPR)
<i>id</i>	[OUT]: Stores the type id

Returns

SAVAPI_S_OK on success or an error otherwise

Warning

Not implemented yet.

6.25.2.12 SAVAPI_get_fops()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_fops (
    SAVAPI_FD savapi_fd,
    void ** fops_pointer,
    void ** fops_context )
```

Get the fops which is currently used by the engine.

Parameters

<i>savapi_fd</i>	[IN]: Handle of the SAVAPI instance
<i>fops_pointer</i>	[OUT]: The fops used in the current savapi session
<i>fops_context</i>	[OUT]: The fops context used in the current savapi session

Returns

SAVAPI_S_OK for success or an error code otherwise.

Note

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

6.25.2.13 SAVAPI_get_user_data()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_user_data (
    SAVAPI_FD savapi_fd,
    void ** user_data )
```

Gets user specific data. This functions gets the user data set by [SAVAPI_set_user_data](#).

Parameters

<i>savapi_fd</i>	[IN]: Handle to the SAVAPI instance.
<i>user_data</i>	[OUT]: User specific data

Returns

SAVAPI_S_OK in case of success or an error code otherwise

6.25.2.14 SAVAPI_get_version()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_version (
    SAVAPI_VERSION * version )
```

Returns SAVAPI library version.

Parameters

<i>version</i>	[OUT]: Pointer to the structure where to store the result
----------------	---

Returns

SAVAPI_S_OK on success or an error otherwise

6.25.2.15 SAVAPI_global_set()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_global_set (
    SAVAPI_GLOBAL_OPTION optionId,
    SAVAPI_TCHAR * buffer )
```

Sets SAVAPI global settings.

Parameters

<i>option↔ Id</i>	[IN]: The id of the option to be set
<i>buffer</i>	[IN]: Pointer to the value to be set

Returns

SAVAPI_S_OK on success or an error otherwise

Note

All options related to SAVAPI OnAccess must be set after [SAVAPI_OA_create_instances\(\)](#) has been called.

6.25.2.16 SAVAPI_initialize()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_initialize (
    SAVAPI_GLOBAL_INIT * savapi_init )
```

SAVAPI initialization function Initializes the SAVAPI library, according to the parameters specified in the initialization structure. It should be called once per process, but it may be called several times per process only if [SAVAPI_uninitialize\(\)](#) has been called in between. The latter case is useful when initializing SAVAPI in 'quick load' mode ([SAVAPI_set_quickload_init\(\)](#)), and then uninitializing and reinitializing SAVAPI in 'normal mode' for scanning purposes.

Parameters

<i>savapi_init</i>	[IN]: A pointer to the initialization structure, which must be filled with the proper values for initialization
--------------------	---

Returns

SAVAPI_S_OK on success or an error otherwise

Note

The initialization function must be called before calling any other SAVAPI function (except the [SAVAPI_set_log_callback\(\)](#) and [SAVAPI_set_quickload_init\(\)](#) functions).

The [SAVAPI_GLOBAL_INIT](#) structure is copied internally and it is not longer needed by SAVAPI after calling this function.

6.25.2.17 SAVAPI_is_running()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running ( )
```

Determines if the SAVAPI daemon is running The library must be initialized with the proper daemon connection parameters for this function to run correctly.

Returns

- 0 If the daemon is stopped
- 1 If the daemon is running
- SAVAPI_E_NOT_INITIALIZED If the SAVAPI library was not properly initialized.

Note

This function is deprecated, use [SAVAPI_is_running_ex\(\)](#) instead.

6.25.2.18 SAVAPI_is_running_ex()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running_ex (
    const SAVAPI_TCHAR * hostname,
    unsigned int port )
```

Determines if the SAVAPI daemon is running on the specified interface (hostname and port)

Parameters

<i>hostname</i>	[IN]: Specifies the host on which the SAVAPI daemon is located.
<i>port</i>	[IN]: Specifies the port on which to connect to the daemon.

Returns

SAVAPI_S_OK if the SAVAPI daemon is running on the given interface or an error code otherwise.

Note

For local sockets (see [SAVAPI_FLAG_USE_LOCAL_SOCKET](#)) the

Parameters

<i>hostname</i>	must be a path to a file and the
<i>port</i>	must be 0.

Note

This function returns [SAVAPI_E_NOT_SUPPORTED](#) in library mode.

6.25.2.19 SAVAPI_OA_create_instances()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_create_instances (
    SAVAPI_OA_INSTANCE_CALLBACK init_func,
    SAVAPI_OA_INSTANCE_CALLBACK uninit_func )
```

Creates automatically the number of SAVAPI instances specified in `SAVAPI_OA_initialize`. For each instance created, the `callback_func` function will be called with the instance to be configured.

Parameters

<i>init_func</i>	[IN]: Callback function to be called after each OnAccess instance is created
<i>uninit_func</i>	[IN]: Callback function to be called before each OnAccess instance is destroyed

Returns

`SAVAPI_S_OK` on success or an error otherwise

Note

This function must be called after [SAVAPI_OA_initialize\(\)](#)

If `init_func` and `uninit_func` are NULL, OnAccess will have the default behavior, meaning that the infected files will be blocked and the clean files will be allowed

If one of the parameters is NULL, the other must be NULL, also. Otherwise, an error will be returned

The instances creation is asynchronous, they might be created after the function returns

Only supported on Windows.

6.25.2.20 SAVAPI_OA_initialize()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_initialize (
    SAVAPI_OA_GLOBAL_INIT * oa_global_init )
```

SAVAPI initialization function for OnAccess component Initializes the OnAccess library, according to the parameters specified in the initialization structure.

Parameters

<i>oa_global_init</i>	[IN]: A pointer to the initialization structure, which must be filled with the proper values for initialization
-----------------------	---

Returns

`SAVAPI_S_OK` on success or an error otherwise

Note

OnAccess is an optional component of SAVAPI which allows real-time on-access scanning. By calling this function you enable the OA functionality

This function must be called a single time per process

This function must be called after [SAVAPI_initialize\(\)](#)

The [SAVAPI_OA_GLOBAL_INIT](#) structure is copied internally and it is not longer needed by SAVAPI after calling this function.

Only supported on Windows.

6.25.2.21 SAVAPI_OA_start_scan()

`SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_start_scan ()`

Starts the real-time on-access scanning process. During the scan operation the instance registered callbacks may be triggered.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

Only supported on Windows.

6.25.2.22 SAVAPI_OA_stop_scan()

`SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_stop_scan ()`

Stops the real-time on-access scanning process.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

Only supported on Windows.

6.25.2.23 SAVAPI_OA_uninitialize()

`SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_uninitialize ()`

SAVAPI uninitialization function for OnAccess component Unitializes the OnAccess library, cleaning up all used resources.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

OnAccess is an optional component of SAVAPI which allows real-time on-access scanning. By calling this function you disable the OnAccess functionality

This function must be called a single time per process

This function must be called before `SAVAPI_uninitialize()`

Only supported on Windows.

6.25.2.24 SAVAPI_register_callback()

`SAVAPI_EXP SAVAPI_STATUS SAVAPI_register_callback (`
 `SAVAPI_FD savapi_fd,`
 `unsigned int callback_id,`
 `SAVAPI_CALLBACK callback)`

Registers a client defined callback.

Parameters

<i>savapi_fd</i>	[IN]: Handle to the SAVAPI instance on which the callback will be available.
<i>callback↔ _id</i>	[IN]: The callback type (e.g. SAVAPI_CALLBACK_REPORT_FILE_STATUS, SAVAPI_CALLBACK_ARCHIVE_OPEN etc.)
<i>callback</i>	[IN]: Pointer to a callback function

Returns

SAVAPI_S_OK if everything went OK or an error code otherwise

Note

Callback registering is not allowed during scanning operations, otherwise the [SAVAPI_E_BUSY](#) will be returned.

Only one callback function is allowed to be registered per callback type/id. If for the given type/id a callback function was already registered then this function will return [SAVAPI_E_INVALID_PARAMETER](#)

See also

[Callbacks' ids](#)

6.25.2.25 SAVAPI_release_instance()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_release_instance (
    SAVAPI_FD * savapi_fd )
```

Destroys a SAVAPI handler, previously created with [SAVAPI_create_instance](#). The function closes the connection to the SAVAPI daemon for client-mode, or, for library mode, it releases the SAVAPI instance.

Parameters

<i>savapi↔ _fd</i>	[IN/OUT]: SAVAPI instance to be released. As a precaution, the pointer will be nulled in order to become very clear that pointer will be unusable for now on
------------------------	--

Returns

SAVAPI_S_OK on success or an error otherwise

Note

- For each handler created with [SAVAPI_create_instance](#) function, the correspondent **SAVAPI↔
release_instance** must be called!
- After calling the function the **savapi_fd** pointer will be invalid and must not be used anymore

6.25.2.26 SAVAPI_reload_engine()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine ( )
```

Reloads the engine from the location given at global initialization.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

This function will simply call the [SAVAPI_uninitialize](#) routine followed by the [SAVAPI_initialize](#) having as parameter the same data provided at global initialization (aka the call to [SAVAPI_initialize](#)) in order to (re)load the engine files from the initial location(s).

All constraints that apply to the [SAVAPI_uninitialize](#) and [SAVAPI_initialize](#) are also available for this function. Therefore, in order to call the function, all instances must be released otherwise the [SAVAPI_E_BUSY](#) will be returned.

This function is deprecated, use [SAVAPI_reload_engine_ex](#) instead, which allows loading a new engine without interrupt of service.

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

Warning

If something wrong goes when the new engine is loaded (e.g. engine is corrupted, some engine files are missing, etc), the library will not be usable anymore (i.e. all functions will return [SAVAPI_E_NOT_INITIALIZED](#)) so the user should only call [SAVAPI_uninitialize](#) and abort the execution.

This function might fail if called after a [SAVAPI_reload_engine_ex](#) call.

6.25.2.27 SAVAPI_reload_engine_ex()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine_ex (
    const SAVAPI_GLOBAL_INIT * global_init )
```

Reloads the engine from the specified location.

Parameters

<i>global_init</i>	[IN]: A pointer to the initialization structure containing the paths to the new engine and vdf files
--------------------	--

Returns

SAVAPI_S_OK on success or an error otherwise

Note

When this function is called, the engine will be (re)loaded from the specified path. The old engine's instances will be kept until their reference counter will reach 0. Calling this function, affects only the new SAVAPI instances (obtained by calling [SAVAPI_create_instance\(\)](#)). They will use the new loaded engine. The SAVAPI instances that are already started won't be affected by this function, they will continue to use the engine that they were started with.

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

6.25.2.28 SAVAPI_scan()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_scan (
    SAVAPI_FD savapi_fd,
    SAVAPI_TCHAR * file_name )
```

Starts a scanning process. During the scan operation the registered callbacks may be triggered.

Parameters

<i>savapi_fd</i>	[IN]: The handle of the SAVAPI instance that will do the scanning
<i>file_name</i>	[IN]: The name of the file to be scanned.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

The execution will not leave [SAVAPI_scan](#) function until scan process is finished.

SAVAPI supports various scan types depending on the file_name format:

- 'path/to/the/file/on/disk' for normal file scanning.
- 'mem://0xAddress,size,name' for scan in memory. The '0xAddress' is the memory area where the buffer with size 'size' is loaded. The 'name' is the display name used when callbacks are triggered. SAVAPI expects that the file from disk is mapped into memory and it will scan that memory address. SAVAPI does not scan processes in memory.
- 'hex_enc://hex_encoded_filename' for scanning files with filename given using hex encoding. This is useful for special encodings (ex. Chinese) in order to avoid conversions. This is not available on Windows platforms: SAVAPI_E_NOT_SUPPORTED will be returned. All filenames returned via callbacks will also be hex-encoded.
- 'apchash://file_hash1,file_hash2' for directly scanning the files' fingerprints (also referred to as hashes) with APC. Users can compute the hashes by using the apchash library and multiple hashes can be verified in a single scan. NOTE: All engine-related scanning options and some APC-related options have no effect in this scanning mode.

6.25.2.29 SAVAPI_send_signal()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_send_signal (
    SAVAPI_FD savapi_fd,
    unsigned int signal_id,
    SAVAPI_SIGNAL_DATA * data )
```

Sends a signal to a specific SAVAPI instance The [SAVAPI_scan](#) may take a long amount of time to finish scanning its target and in some situations a forced abort would be desirable. In these kind of situations, `SAVAPI_send_signal` may help by sending signals to a running SAVAPI instance ([SAVAPI_SIGNAL_SCAN_ABORT](#) for instance).

Parameters

<i>savapi↔ _fd</i>	[IN]: Handle of the SAVAPI instance
<i>signal_id</i>	[IN]: Identifies the signal to be sent. See Signal IDs section
<i>data</i>	[IN]: Specific data to be sent when sending the signal. See SAVAPI_SIGNAL_DATA

Returns

`SAVAPI_S_OK` for success or an error code otherwise.

Note

The SAVAPI signals were designed to be sent asynchronously, when an event arrives (Ctrl+C was issued for instance) and if program execution is within [SAVAPI_scan](#). It makes no sense to send a signal to a SAVAPI instance when we have execution flow control.

See also

Current supported [SAVAPI signals](#)

6.25.2.30 SAVAPI_set()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_set (
    SAVAPI_FD savapi_fd,
    SAVAPI_OPTION option_id,
    SAVAPI_TCHAR * buffer )
```

Sets SAVAPI individual settings.

Parameters

<i>savapi↔ _fd</i>	[IN]: Handle of the SAVAPI instance
<i>option↔ _id</i>	[IN]: The id of the option to be set
<i>buffer</i>	[IN]: NULL-terminated string containing the value of the option to be set

Returns

SAVAPI_S_OK If everything went ok an error code otherwise.

Note

Calling this function during a scanning operation performed by this instance will fail.

6.25.2.31 SAVAPI_set_fops()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_fops (
    SAVAPI_FD savapi_fd,
    void * fops_pointer,
    void * fops_context )
```

Specify the new fops who will be used by the engine.

Parameters

<i>savapi_fd</i>	[IN]: Handle of the SAVAPI instance
<i>fops_pointer</i>	[IN]: Pointer to the fops to use in the current savapi session
<i>fops_context</i>	[IN]: This context will be passed back to the application by each call to the fops used

Returns

SAVAPI_S_OK for success or an error code otherwise.

Note

This function will return [SAVAPI_E_NOT_SUPPORTED](#) in client-mode

6.25.2.32 SAVAPI_set_log_callback()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_log_callback (
    SAVAPI_LOG_CALLBACK log_fct,
    SAVAPI_LOG_LEVEL min_level,
    void * user_data )
```

Sets the SAVAPI logging function.

Parameters

<i>log_fct</i>	[IN]: The function used for logging. If given function is NULL all data set with a previous SAVAPI_set_log_callback call will be cleared so logging will not be performed anymore.
<i>min_level</i>	[IN]: Sets the desired minimum log level. This can be used to filter unwanted log-levels, so that if a message have a lower level, it will be automatically "thrown" by the SAVAPI
<i>user_data</i>	[IN]: The user context

Returns

SAVAPI_S_OK on success or an error otherwise

Note

This function can be called before or/and after global initialization (calling before it's recommended, so that any error messages in the [SAVAPI_initialize\(\)](#) can be logged)

This can be called several times in the same process, so that SAVAPI's user can change the log-level on-the-fly

6.25.2.33 SAVAPI_set_quickload_init()

```
void SAVAPI_EXP SAVAPI_set_quickload_init ( )
```

Sets the SAVAPI initialization mode to 'quick load' Recommended if only component versions are needed.

Returns

nothing

Note

Must be called before any initialization function ([SAVAPI_initialize\(\)](#))

Scanning is prohibited in this mode

Calling [SAVAPI_uninitialize\(\)](#) resets the mode to 'normal load'

Function doesn't perform anything in client-mode

6.25.2.34 SAVAPI_set_user_data()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_user_data (
    SAVAPI_FD savapi_fd,
    void * user_data )
```

Sets user specific data. This functions sets user data that will be returned untouched as *user_data* member of [SAVAPI_CALLBACK_DATA](#) structure.

Parameters

<i>savapi_fd</i>	[IN]: Handle to the SAVAPI instance.
<i>user_data</i>	[IN]: User specific data

Returns

SAVAPI_S_OK in case of success or an error code otherwise

Note

The user is responsible with the memory management. This function will only set the value given in the [SAVAPI_CALLBACK_DATA](#) structure. It will not reserve or free memory for the data.

6.25.2.35 SAVAPI_simple_scan()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_simple_scan (
    SAVAPI_FD savapi_fd,
    SAVAPI_TCHAR * file_name,
    SAVAPI_SIMPLE_SCAN_OUTPUT * output )
```

Starts a scanning process during which no callbacks will be triggered.

Parameters

<i>savapi_fd</i>	[IN]: The handle of the SAVAPI instance that will do the scanning
<i>file_name</i>	[IN]: The name of the file to be scanned
<i>output</i>	[OUT]: The output of the scan, containing a list of the files that were found to be infected, suspicious, erroneous or for which there is additional information (example: office macros)

Returns

SAVAPI_S_OK if no infected or suspicious file was scanned, and if there was no error while scanning any file SAVAPI_S_INFECTED if at least one infected file was found SAVAPI_S_SUSPICIOUS if at least one suspicious file was found, but no infected file SAVAPI_E_SCAN_ERROR if an error was encountered while scanning any of the files, but no infected or suspicious file specific error code if a non scanning error was encountered during the scan process

Note

The output data structure requires no memory management. Memory allocation and deallocation is handled internally by SAVAPI.

More information about the error can be obtained by checking the error code of each file in the list (if it contains any such files).

SAVAPI supports various scan types depending on the file_name format:

- 'path/to/the/file/on/disk' for normal file scanning.
- 'mem://0xAddress,size,name' for scan in memory. The '0xAddress' is the memory area where the buffer with size 'size' is loaded. SAVAPI expects that the file from disk is mapped into memory and it will scan that memory address. SAVAPI does not scan processes in memory.
- 'hex_enc://hex_encoded_filename' for scanning files with filename given using hex encoding. This is useful for special encodings (ex. Chinese) in order to avoid conversions. This is not available on Windows platforms: SAVAPI_E_NOT_SUPPORTED will be returned.
- 'apchash://file_hash1,file_hash2' for directly scanning the files' fingerprints (also referred to as hashes) with APC. Users can compute the hashes by using the apchash library and multiple hashes can be verified in a single scan. NOTE: All engine-related scanning options and some APC-related options have no effect in this scanning mode.

No callbacks and user data set before calling this function will be used. They can still be used with a regular [SAVAPI_scan](#) call

6.25.2.36 SAVAPI_uninitialize()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_uninitialize ( )
```

SAVAPI uninitialization function Uninitializes the SAVAPI library, cleaning up all used resources. Once called, all subsequent SAVAPI calls will fail with SAVAPI_E_NOT_INITIALIZED error code.

Returns

SAVAPI_S_OK on success or an error otherwise

Note

All SAVAPI instances must be released before calling this function, otherwise the SAVAPI_E_BUSY will be returned.

If SAVAPI_set_quickload_init() has been called before, then calling this function will also reset the mode to 'normal load'

If calling this function without prior SAVAPI initialization, an error will be returned

6.25.2.37 SAVAPI_unregister_callback()

```
SAVAPI_EXP SAVAPI_STATUS SAVAPI_unregister_callback (
    SAVAPI_FD savapi_fd,
    unsigned int callback_id,
    SAVAPI_CALLBACK callback )
```

Unregisters a previously registered client defined callback.

Parameters

<i>savapi_fd</i>	[IN]: Handle to the SAVAPI instance.
<i>callback↔ _id</i>	[IN]: The callback type (e.g. SAVAPI_CALLBACK_REPORT_FILE_STATUS, SAVAPI_CALLBACK_ARCHIVE_OPEN etc.)
<i>callback</i>	[IN]: Pointer to a callback function

Returns

SAVAPI_S_OK in case of success or an error code otherwise

Note

Callback unregistering is not allowed during scanning operations, otherwise the SAVAPI_E_BUSY will be returned.

The callback type/id is searched by SAVAPI in its internal callback list and if found then the callback function will be removed, otherwise this function will return SAVAPI_E_INVALID_PARAMETER

6.26 SAVAPI return codes

Typedefs

- typedef enum SAVAPI_status SAVAPI_STATUS

Enumerations

- enum SAVAPI_status {
 SAVAPI_S_OK = 0 , SAVAPI_E_INVALID_PARAMETER = 1 ,
 SAVAPI_E_ALREADY_INITIALIZED = 2 , SAVAPI_E_NOT_INITIALIZED = 3 ,
 SAVAPI_E_BUFFER_TOO_SMALL = 4 , SAVAPI_E_CONNECTION_MODE_NOT_SET = 5 ,
 SAVAPI_E_HOSTNAME_NOT_SET = 6 , SAVAPI_E_NO_MEMORY = 7 ,
 SAVAPI_E_VDF_NOT_FOUND = 8 , SAVAPI_E_VDF_READ = 9 ,
 SAVAPI_E_VDF_CRC = 10 , SAVAPI_E_VDF_VERSION = 11 ,
 SAVAPI_E_WRONG_ENGINE = 12 , SAVAPI_E_ENGINE_NOT_FOUND = 13 ,
 SAVAPI_E_SELFCHK_PATCHED = 14 , SAVAPI_E_SELFCHK_FILE_ERR = 15 ,
 SAVAPI_E_SELFCHK_FILE_CRC = 16 , SAVAPI_E_KEYFILE = 17 ,
 SAVAPI_E_INTERNAL = 18 , SAVAPI_E_NOT_SUPPORTED = 19 ,
 SAVAPI_E_RESULT_FILE_NOT_FOUND = 20 , SAVAPI_E_OPTION_NOT_SUPPORTED = 21 ,
 SAVAPI_E_HIT_MAX_REC = 22 , SAVAPI_E_HIT_MAX_SIZE = 23 ,
 SAVAPI_E_HIT_MAX_RATIO = 24 , SAVAPI_E_ENCRYPTED = 25 ,
 SAVAPI_E_UNSUPPORTED = 26 , SAVAPI_E_PROC_ERROR = 27 ,
 SAVAPI_E_INCOMPLETE = 28 , SAVAPI_E_PARTIAL = 29 ,
 SAVAPI_E_HIT_MAX_COUNT = 30 , SAVAPI_E_ABORTED = 31 ,
 SAVAPI_E_TIMEOUT = 32 , SAVAPI_E_FILE_OPEN = 33 ,
 SAVAPI_E_FILE_READ = 34 , SAVAPI_E_FILE_WRITE = 35 ,
 SAVAPI_E_INVALID_VALUE = 36 , SAVAPI_E_CHDIR_FAILED = 37 ,
 SAVAPI_E_NOT_ABSOLUTE_PATH = 38 , SAVAPI_E_DIR_NOT_EXISTS = 39 ,
 SAVAPI_E_MATCHED = 40 , SAVAPI_E_CONVERSION_FAILED = 41 ,
 SAVAPI_E_CONNECTION_FAILED = 42 , SAVAPI_E_RECEIVE_FAILED = 43 ,
 SAVAPI_E_SEND_FAILED = 44 , SAVAPI_E_OPTION_VALUE_INVALID = 45 ,
 SAVAPI_E_REPAIR_FAILED = 46 , SAVAPI_E_FILE_CREATE = 47 ,
 SAVAPI_E_FILE_DELETE = 48 , SAVAPI_E_FILE_CLOSE = 49 ,
 SAVAPI_E_UNKNOWN = 50 , SAVAPI_E_PREFIX_SET = 51 ,
 SAVAPI_E_PREFIX_GET = 52 , SAVAPI_E_INVALID_QUERY = 53 ,
 SAVAPI_E_KEY_NO_KEYFILE = 54 , SAVAPI_E_KEY_ACCESS_DENIED = 55 ,
 SAVAPI_E_KEY_INVALID_HEADER = 56 , SAVAPI_E_KEY_KEYFILE_VERSION = 57 ,
 SAVAPI_E_KEY_NO_LICENSE = 58 , SAVAPI_E_KEY_FILE_INVALID = 59 ,
 SAVAPI_E_KEY_RECORD_INVALID = 60 , SAVAPI_E_KEY_EVAL_VERSION = 61 ,
 SAVAPI_E_KEY_DEMO_VERSION = 62 , SAVAPI_E_KEY_ILLEGAL_LICENSE = 63 ,
 SAVAPI_E_KEY_EXPIRED = 64 , SAVAPI_E_KEY_READ = 65 ,
 SAVAPI_E_LICENSE_RESTRICTION = 66 , SAVAPI_E_LOADING_ENGINE_MODULES = 67 ,
 SAVAPI_E_BUSY = 68 , SAVAPI_E_ENCRYPTED_MIME = 69 ,
 SAVAPI_E_NON_ADDRESSABLE = 70 , SAVAPI_E_MEMORY_LIMIT = 71 ,
 SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ = 72 , SAVAPI_E_PROC_BAD_HEADER = 73 ,
 SAVAPI_E_PROC_INVALID_COMPRESSED_DATA = 74 , SAVAPI_E_PROC_OBSOLETE = 75 ,
 SAVAPI_E_PROC_BAD_FORMAT = 76 , SAVAPI_E_PROC_HEADER_CRC = 77 ,
 SAVAPI_E_PROC_DATA_CRC = 78 , SAVAPI_E_PROC_FILE_CRC = 79 ,
 SAVAPI_E_PROC_BAD_TABLE = 80 , SAVAPI_E_PROC_UNEXPECTED_EOF = 81 ,
 SAVAPI_E_PROC_ARCHIVE_HANDLE = 82 , SAVAPI_E_PROC_NO_FILES_TO_EXTRACT = 83 ,
 SAVAPI_E_PROC_CALLBACK = 84 , SAVAPI_E_PROC_TOTAL_LOSS = 85 ,
 SAVAPI_E_APC_ERROR = 86 , SAVAPI_E_APC_CONNECTION = 87 ,
 SAVAPI_E_APC_NOT_SUPPORTED = 88 , SAVAPI_E_APC_TIMEOUT = 89 ,
 SAVAPI_E_APC_TEMPORARILY_DISABLED = 90 , SAVAPI_E_APC_INCOMPLETE = 91 ,
 SAVAPI_E_APC_NO_LICENSE = 92 , SAVAPI_E_APC_AUTHENTICATION = 93 ,
 }

```

SAVAPI_E_APC_AUTH_RETRY_LATER = 94 , SAVAPI_E_APC_RANDOM_ID = 95 ,
SAVAPI_E_APC_NOT_INITIALIZED = 96 , SAVAPI_E_APC_ALREADY_INITIALIZED = 97 ,
SAVAPI_E_APC_DISABLED = 98 , SAVAPI_E_APC_TIMEOUT_RESTRICTION = 99 ,
SAVAPI_E_APC_UNKNOWN_CATEGORY = 100 , SAVAPI_E_APC_QUOTA = 101 ,
SAVAPI_E_UNSUPPORTED_COMPRESSION = 1000 , SAVAPI_E_OA_NOT_INITIALIZED = 2000 ,
SAVAPI_E_OA_INITIALIZED = 2001 , SAVAPI_E_OA_NO_INSTANCES_CREATED = 2002 ,
SAVAPI_E_OA_INSTANCES_CREATED = 2003 , SAVAPI_E_OA_NO_SCAN_IN_PROGRESS = 2004 ,
SAVAPI_E_OA_SCAN_IN_PROGRESS = 2005 , SAVAPI_E_OA_NO_LICENSE = 2006 ,
SAVAPI_E_OA_ERROR = 2007 , SAVAPI_E_OA_NO_PRIVILEGES = 2008 ,
SAVAPI_E_OA_DRIVERS = 2009 , SAVAPI_E_FPC_TIMEOUT_RESTRICTION = 3000 ,
SAVAPI_S_INFECTED = 4000 , SAVAPI_S_SUSPICIOUS = 4001 ,
SAVAPI_E_SCAN_ERROR = 4002 }

```

6.26.1 Detailed Description

6.26.2 Typedef Documentation

6.26.2.1 SAVAPI_STATUS

```
typedef enum SAVAPI_status SAVAPI_STATUS
```

6.26.3 Enumeration Type Documentation

6.26.3.1 SAVAPI_status

```
enum SAVAPI_status
```

Enumerator

SAVAPI_S_OK	Operation ended with success.
SAVAPI_E_INVALID_PARAMETER	One of supplied parameters is invalid. Note At least one of the function's parameters is invalid (invalid pointers, empty strings, out of range values, etc.).
SAVAPI_E_ALREADY_INITIALIZED	SAVAPI was already initialized. Note Trying to initialize an already initialized SAVAPI library (SAVAPI_initialize was already called successfully).

Enumerator

SAVAPI_E_NOT_INITIALIZED	<p>SAVAPI is not initialized.</p> <p>Note</p> <p>The used functionality requires the SAVAPI library to be initialized first (a successful call of SAVAPI_initialize is needed before).</p>
SAVAPI_E_BUFFER_TOO_SMALL	<p>Supplied buffer is too small.</p> <p>Note</p> <p>An interface function that requires a buffer size as parameter was called with a value smaller than the needed size.</p>
SAVAPI_E_CONNECTION_MODE_NOT_SET	<p>Connection mode flag is not set.</p> <p>Note</p> <p>The SAVAPI_INSTANCE_INIT::flags in the instance creation structure is not set to a known connection mode.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_HOSTNAME_NOT_SET	<p>Host name is not set.</p> <p>Note</p> <p>The SAVAPI_INSTANCE_INIT::host_name field in the instance creation structure was not set.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_NO_MEMORY	<p>Memory allocation failed.</p> <p>Note</p> <p>There is not enough memory available for allocation.</p>
SAVAPI_E_VDF_NOT_FOUND	<p>VDF file(s) not found.</p> <p>Note</p> <p>Path to the VDF files is not correct, files are missing, or there are no access rights to open the files.</p>
SAVAPI_E_VDF_READ	<p>VDF file(s) read failed.</p> <p>Note</p> <p>VDF files are damaged or truncated.</p>
SAVAPI_E_VDF_CRC	<p>VDF file(s) crc check failed.</p> <p>Note</p> <p>One or more VDF files failed checksum check because they were damaged, manipulated or truncated.</p>

Enumerator

SAVAPI_E_VDF_VERSION	<p>Inconsistent versions in VDF files set.</p> <p>Note</p> <p>There are incompatible VDF files within the VDF set. Not all relevant VDF files were downloaded or the engine is too old for the present VDF set.</p>
SAVAPI_E_WRONG_ENGINE	<p>Engine initialization failed.</p> <p>Note</p> <p>Engine is too old for this version of SAVAPI. SAVAPI used a wrong character set when initializing the engine.</p>
SAVAPI_E_ENGINE_NOT_FOUND	<p>Engine file(s) not found.</p> <p>Note</p> <p>One or more engine files are not present in the engine's directory.</p>
SAVAPI_E_SELFCHK_PATCHED	<p>Inconsistent versions in engine files set.</p> <p>Note</p> <p>There are incompatible engine files within the engine set which do not match the expected version. The engine file set was not updated or is too old for the present engine set.</p>
SAVAPI_E_SELFCHK_FILE_ERR	<p>Engine file(s) read failed.</p> <p>Note</p> <p>One or more engine files are damaged or truncated.</p>
SAVAPI_E_SELFCHK_FILE_CRC	<p>Engine file(s) crc check failed.</p> <p>Note</p> <p>One or more engine files failed checksum check because they were manipulated, damaged or truncated.</p>
SAVAPI_E_KEYFILE	<p>Generic key file error.</p>
SAVAPI_E_INTERNAL	<p>SAVAPI internal error.</p> <p>Note</p> <p>An unexpected internal event prevented the normal execution of the library (incorrect pointers, incorrect return values, etc.). Normally this error should never occur. If this error occurs there is a major problem which must be fixed.</p>

Enumerator

SAVAPI_E_NOT_SUPPORTED	<p>Unsupported feature.</p> <p>Note</p> <p>The requested functionality (feature, command, option) may be known but it is not supported by this version of SAVAPI or engine. For instance a called function is not available in the current library mode (SAVAPI_is_running_ex() is not available in SAVAPI Library, or SAVAPI_reload_engine_ex() is not available in SAVAPI Client Library); or the used signal id is unknown (the only known signal for SAVAPI_send_signal is SAVAPI_SIGNAL_SCAN_ABORT); or a new functionality was added but is not yet implemented or not supported yet by the current library version or within the current engine version.</p>
SAVAPI_E_RESULT_FILE_NOT_FOUND	<p>Could not extract file.</p> <p>Note</p> <p>A file to extract during an archive scanning could not be found.</p>
SAVAPI_E_OPTION_NOT_SUPPORTED	<p>Option is not supported.</p> <p>Note</p> <p>Trying to set or retrieve a value for an option with an unknown or obsolete id (for instance, option SAVAPI_OPTION_UPDATE_SERVERS is obsolete).</p>
SAVAPI_E_HIT_MAX_REC	<p>Archive maximum recursion limit reached.</p> <p>Note</p> <p>The limit on the maximum number of archive recursions was exceeded when extracting a file because the file was packed too many times or it contained other deeply nested files. The decompression will be aborted as soon as the limit is exceeded.</p>
SAVAPI_E_HIT_MAX_SIZE	<p>Archive maximum extraction size reached.</p> <p>Note</p> <p>Size of an uncompressed file has exceeded the maximum extraction size. The decompression will be aborted as soon as the limit is exceeded.</p>
SAVAPI_E_HIT_MAX_RATIO	<p>Archive maximum extraction ratio reached.</p> <p>Note</p> <p>Size of an uncompressed file has exceeded the maximum extraction ratio. The decompression will be aborted as soon as the limit is exceeded.</p>

Enumerator

SAVAPI_E_ENCRYPTED	<p>Encrypted contents found.</p> <p>Note</p> <p>One or more files inside the archive are encrypted, but there are also files which are not encrypted and can be extracted; or all files inside the archive are encrypted and it's not possible to extract them.</p>
SAVAPI_E_UNSUPPORTED	<p>Unsupported archive type/format.</p> <p>Note</p> <p>The archive type is not supported. The version of a known archive type is not supported. The archive format is unknown.</p>
SAVAPI_E_PROC_ERROR	<p>Archive generic processing error.</p> <p>Note</p> <p>Any other archive scan processing error which is not covered by SAVAPE_PROC_<name> error codes.</p>
SAVAPI_E_INCOMPLETE	<p>File was not completely scanned.</p> <p>Note</p> <p>Scanning was aborted by user or as result of a terminal warning or error.</p>
SAVAPI_E_PARTIAL	<p>Cannot extract multi-volume archive.</p> <p>Note</p> <p>In case of an archive which is part of a multi-volume archive set, a file could not be fully extracted because is split over several archive parts. Processing the next file may be successful if all information is stored in that part.</p>
SAVAPI_E_HIT_MAX_COUNT	<p>Maximum number of files in archive reached.</p> <p>Note</p> <p>Maximum files count limit was reached while scanning an archive. The scanning will be aborted as soon as the limit is exceeded.</p>
SAVAPI_E_ABORTED	<p>Scan was aborted by signal.</p> <p>Note</p> <p>A scan in progress was aborted by user with SAVAPE_SIGNAL_SCAN_ABORT signal.</p>
SAVAPI_E_TIMEOUT	<p>Scan timed out.</p> <p>Note</p> <p>A scan in progress exceeded the maximum user set scan time-out.</p>

Enumerator

SAVAPI_E_FILE_OPEN	<p>Could not open file.</p> <p>Note</p> <p>File is missing or there are no access rights to open it.</p>
SAVAPI_E_FILE_READ	<p>Could not read file. File read error</p> <p>Note</p> <p>There are no access rights to read file, or the file has been removed, or data from file end is missing, or file is truncated.</p>
SAVAPI_E_FILE_WRITE	<p>Could not write file.</p> <p>Note</p> <p>There are no access rights to write file, or the file has been removed. Disk quota exceeded or disk is damaged.</p>
SAVAPI_E_INVALID_VALUE	<p>Invalid value in configuration or command.</p> <p>Note</p> <p>Failure in SAVAPI Client Library communication with SAVAPI Service resulting in commands with invalid values which cannot be accepted by Service (for instance SET or GET commands with invalid values). The engine path given to the SAVAPI_reload_engine_ex() function collides with previous engine path.</p>
SAVAPI_E_CHDIR_FAILED	<p>Could not change directory.</p> <p>Note</p> <p>Failure in SAVAPI Client Library communication with SAVAPI Service resulting in an unsuccessful SET CWD command.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_NOT_ABSOLUTE_PATH	<p>Path is not absolute.</p> <p>Note</p> <p>Path to a given or required directory is not absolute (for example the path of the temporary scanning directory is not absolute).</p>
SAVAPI_E_DIR_NOT_EXISTS	<p>Directory path does not exist.</p> <p>Note</p> <p>Path to a given directory does not exist (for example the path of the temporary scanning directory does not exist).</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>

Enumerator

SAVAPI_E_MATCHED	<p>File was filtered from scanning.</p> <p>Note</p> <p>File matched a black list rule and was not scanned.</p>
SAVAPI_E_CONVERSION_FAILED	<p>Converting failed.</p> <p>Note</p> <p>A string could not be converted from one encoding to another (for instance a string could not be converted from SAVAPI_TCHAR to char, or in case of SAVAPI Client Library a string could not be converted from SAVAPI_TCHAR to the SAVAPI Service's text mode encoding).</p>
SAVAPI_E_CONNECTION_FAILED	<p>Connection with the SAVAPI Service failed.</p> <p>Note</p> <p>The SAVAPI service is not running on the specified interface.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_RECEIVE_FAILED	<p>Failed to receive data from the SAVAPI Service.</p> <p>Note</p> <p>The SAVAPI service is not running anymore.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_SEND_FAILED	<p>Failed to send data to the SAVAPI Service.</p> <p>Note</p> <p>The SAVAPI service is not running anymore.</p> <p>This error can only be triggered by the SAVAPI Client Library.</p>
SAVAPI_E_OPTION_VALUE_INVALID	<p>Invalid option value.</p> <p>Note</p> <p>A configuration command received a value buffer which is not acceptable as a value for the associated option id (for instance it is empty).</p>
SAVAPI_E_REPAIR_FAILED	<p>Repair an infected file failed.</p>
SAVAPI_E_FILE_CREATE	<p>Failed to create file.</p> <p>Note</p> <p>Failed to create a temporary file in the temporary scanning directory because there are no access rights, or the file already exists, etc.</p>

Enumerator

SAVAPI_E_FILE_DELETE	Failed to delete file. Note Failed to delete a temporary file in the temporary scanning directory because there are no access rights, file is locked, file does not exist anymore, etc.
SAVAPI_E_FILE_CLOSE	Failed to close file. Note Failed to close a temporary file in the temporary scanning directory because there are no access rights, file was accidentally deleted, etc.
SAVAPI_E_UNKNOWN	Unknown engine error. Note Engine returns an unknown error code.
SAVAPI_E_PREFIX_SET	Failed to set a detect type option. Note SAVAPI failed to set a detect type option (for instance SAVAPI_OPTION_DETECT_ADSPY , SAVAPI_OPTION_DETECT_APPL , others)
SAVAPI_E_PREFIX_GET	Failed to retrieve a detect type option. Note SAVAPI failed to retrieve a detect type option (for instance SAVAPI_OPTION_DETECT_ADSPY , SAVAPI_OPTION_DETECT_APPL , others).
SAVAPI_E_INVALID_QUERY	Invalid query for SAVAPI Service. Note Failure in SAVAPI Client Library communication with SAVAPI Service resulting in an unacceptable command (invalid command, syntax error). This error can only be triggered by the SAVAPI Client Library
SAVAPI_E_KEY_NO_KEYFILE	Keyfile has not been found.
SAVAPI_E_KEY_ACCESS_DENIED	Access to key file has been denied.
SAVAPI_E_KEY_INVALID_HEADER	An invalid header has been found.
SAVAPI_E_KEY_KEYFILE_VERSION	Invalid keyfile version number.
SAVAPI_E_KEY_NO_LICENSE	No valid license found.
SAVAPI_E_KEY_FILE_INVALID	Key file is invalid (invalid CRC)
SAVAPI_E_KEY_RECORD_INVALID	Invalid license record detected.
SAVAPI_E_KEY_EVAL_VERSION	Application is evaluation version.
SAVAPI_E_KEY_DEMO_VERSION	Application is demo version.
SAVAPI_E_KEY_ILLEGAL_LICENSE	Illegal (cracked) license in keyfile.
SAVAPI_E_KEY_EXPIRED	This key has expired.

Enumerator

SAVAPI_E_KEY_READ	Error reading from key file.
SAVAPI_E_LICENSE_RESTRICTION	<p>Operation not allowed (license restriction)</p> <p>Note</p> <p>Scan command was issued without setting a valid product id.</p>
SAVAPI_E_LOADING_ENGINE_MODULES	<p>Error loading engine modules.</p> <p>Note</p> <p>SAVAPI could not load engine modules because they are not available or there are no access rights.</p>
SAVAPI_E_BUSY	<p>SAVAPI is busy.</p> <p>Note</p> <p>A configuration request was given during scanning a file (for instance SET/GET command or callback register/unregister command). SAVAPI_uninitialize was called without releasing all SAVAPI instances before.</p>
SAVAPI_E_ENCRYPTED_MIME	<p>Encrypted mail found.</p> <p>Note</p> <p>While scanning an archive an encrypted mail was found.</p>
SAVAPI_E_NON_ADDRESSABLE	<p>Non addressable memory location.</p> <p>Note</p> <p>A scan request was issued for an address that is not in the available address space for the current platform. For example, on a 64 bit machine the available address space is [0..MAX_INT_64].</p>
SAVAPI_E_MEMORY_LIMIT	<p>Internal memory limit reached.</p> <p>Note</p> <p>An engine-internal safety limit regarding memory usage of a subroutine has been reached (this can i.e. be caused by excessively large dictionaries in archives).</p>
SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ	<p>Incomplete archive block read.</p> <p>Note</p> <p>An archive block is damaged and could not be read.</p>
SAVAPI_E_PROC_BAD_HEADER	<p>Bad archive header.</p> <p>Note</p> <p>The archive header is invalid.</p>

Enumerator

SAVAPI_E_PROC_INVALID_COMPRESSED_DATA	<p>Bad compressed data.</p> <p>Note</p> <p>The compressed data from the archive is invalid. Some files could not be extracted and scanned.</p>
SAVAPI_E_PROC_OBSOLETE	<p>Obsolete archive information.</p> <p>Note</p> <p>Archive is packed with a very old or a developer version of a packer application and contains obsolete information and unsupported entries.</p>
SAVAPI_E_PROC_BAD_FORMAT	<p>Bad header format.</p> <p>Note</p> <p>The archive header has been changed with a newer (unsupported) version of a packer application. The archive header is damaged.</p>
SAVAPI_E_PROC_HEADER_CRC	<p>Bad header crc.</p> <p>Note</p> <p>An archive header failed checksum check.</p>
SAVAPI_E_PROC_DATA_CRC	<p>Bad data crc.</p> <p>Note</p> <p>Checksum of compressed data does not match.</p>
SAVAPI_E_PROC_FILE_CRC	<p>Bad crc for extracted file.</p> <p>Note</p> <p>Checksum of a decompressed file does not match.</p>
SAVAPI_E_PROC_BAD_TABLE	<p>Invalid decompression table.</p> <p>Note</p> <p>Archive contains an invalid decompression table.</p>
SAVAPI_E_PROC_UNEXPECTED_EOF	<p>Unexpected end of file.</p> <p>Note</p> <p>Decompression aborted because of unexpected end of file in archive.</p>
SAVAPI_E_PROC_ARCHIVE_HANDLE	<p>Archive internal handle error.</p> <p>Note</p> <p>An internal handle related to archive processing is invalid or not initialized.</p>

Enumerator

SAVAPI_E_PROC_NO_FILES_TO_EXTRACT	<p>No files could be extracted.</p> <p>Note</p> <p>Archive is invalid, corrupt or damaged.</p>
SAVAPI_E_PROC_CALLBACK	<p>Archive internal callback error.</p> <p>Note</p> <p>Decompression aborted because an internal archive callback is invalid or caused an error.</p>
SAVAPI_E_PROC_TOTAL_LOSS	<p>File extraction failed.</p> <p>Note</p> <p>Not all archive contents could be extracted.</p>
SAVAPI_E_APC_ERROR	Generic APC-related error.
SAVAPI_E_APC_CONNECTION	<p>APC connection error.</p> <p>Note</p> <p>An error occurred while communicating with the cloud server.</p>
SAVAPI_E_APC_NOT_SUPPORTED	<p>APC protocol is not supported.</p> <p>Note</p> <p>The APC protocol is no longer supported and must be updated.</p>
SAVAPI_E_APC_TIMEOUT	<p>APC operation timed out.</p> <p>Note</p> <p>Either the APC connection timeout, APC scan timeout or global scan timeout was reached while an APC operation was taking place.</p>
SAVAPI_E_APC_TEMPORARILY_DISABLED	APC is currently disabled due to too many failed APC scans.
SAVAPI_E_APC_INCOMPLETE	<p>File could not be scanned with APC.</p> <p>Note</p> <p>APC scanning was aborted by user or as result of an error.</p>
SAVAPI_E_APC_NO_LICENSE	No valid APC license found in the key file.
SAVAPI_E_APC_AUTHENTICATION	APC authentication failed.
SAVAPI_E_APC_AUTH_RETRY_LATER	APC authentication failed, but it should be retried later.

Enumerator

SAVAPI_E_APC_RANDOM_ID	<p>Initially, APC random will be read from "apc_random_id" file. If for some reason the operation fails or the ID is invalid, the ID is computed. If this also fails, the SAVAPI_E_APC_RANDOM_ID will be returned.</p> <p>Note</p> <p>A valid APC random id is a "40 printable characters" string.</p>
SAVAPI_E_APC_NOT_INITIALIZED	<p>APC has not been initialized.</p> <p>Note</p> <p>The used functionality requires the APC library to be initialized first (a successful call of SAVAPI_APC_initialize is needed before).</p>
SAVAPI_E_APC_ALREADY_INITIALIZED	<p>APC was already initialized.</p> <p>Note</p> <p>Trying to initialize an already initialized APC library (SAVAPI_APC_initialize was already called successfully).</p>
SAVAPI_E_APC_DISABLED	<p>APC permanently disabled.</p> <p>Note</p> <p>This error is given when APC is disabled</p>
SAVAPI_E_APC_TIMEOUT_RESTRICTION	<p>APC timeout restrictions not met.</p> <p>Note</p> <p>This error is given when APC was enabled and the user tries to set new APC timeouts that do not comply with the following restriction: $APCConnectionTimeout < APCScanTimeout < ScanTimeout$.</p>
SAVAPI_E_APC_UNKNOWN_CATEGORY	<p>A category could not be determined for the object which was scanned by APC.</p> <p>Note</p> <p>This is generally caused by either:</p> <ul style="list-style-type: none"> • apc_mode being set to SAVAPI_APC_SCAN_MODE_CHECK_ONLY; or • scanning a hash (no file being involved, there is nothing to upload).
SAVAPI_E_APC_QUOTA	<p>APC quota limit reached.</p> <p>Note</p> <p>This error is given when APC quota limit has been reached. Please contact Avira support</p>

Enumerator

SAVAPI_E_UNSUPPORTED_COMPRESSION	<p>Unsupported compression method.</p> <p>Note</p> <p>The compression method of the archive is not supported.</p>
SAVAPI_E_OA_NOT_INITIALIZED	<p>OnAccess has not been initialized.</p> <p>Note</p> <p>The used functionality requires the OnAccess library to be initialized first (a successful call of SAVAPI_OA_initialize is needed before).</p>
SAVAPI_E_OA_INITIALIZED	<p>OnAccess was already initialized.</p> <p>Note</p> <p>Trying to initialize an already initialized OnAccess library (SAVAPI_OA_initialize was already called successfully).</p>
SAVAPI_E_OA_NO_INSTANCES_CREATED	<p>OnAccess instances have not been created yet.</p> <p>Note</p> <p>Trying to do an action (for e.g. calling SAVAPI_OA_start_scan) which requires OA instances to be created</p>
SAVAPI_E_OA_INSTANCES_CREATED	<p>OnAccess instances have already been created.</p> <p>Note</p> <p>Trying to create OnAccess instances when they are already created (SAVAPI_OA_create_instances was already called successfully).</p>
SAVAPI_E_OA_NO_SCAN_IN_PROGRESS	<p>No OnAccess scanning in progress.</p> <p>Note</p> <p>Trying to stop an OnAccess scan which is not in progress (SAVAPI_OA_stop_scan was already called successfully or no SAVAPI_OA_start_scan was called).</p>
SAVAPI_E_OA_SCAN_IN_PROGRESS	<p>OnAccess scanning is already in progress.</p> <p>Note</p> <p>Trying to start an OnAccess scan which is already in progress (SAVAPI_OA_start_scan was already called successfully).</p>
SAVAPI_E_OA_NO_LICENSE	No valid OA license found in the key file.
SAVAPI_E_OA_ERROR	Generic OnAccess-related error.

Enumerator

SAVAPI_E_OA_NO_PRIVILEGES	Not enough privileges when trying to initialize OnAccess module. Note Administrator privileges are needed.
SAVAPI_E_OA_DRIVERS	OnAccess drivers are not installed or not running.
SAVAPI_E_FPC_TIMEOUT_RESTRICTION	FPC timeout restrictions not met. Note This error is given when FPC was enabled and the user tries to set new FPC timeout that do not comply with the following restriction: FPCTimeout < ScanTimeout.
SAVAPI_S_INFECTED	
SAVAPI_S_SUSPICIOUS	
SAVAPI_E_SCAN_ERROR	

6.27 Plugin defines

Macros

- #define SAVAPI_PLG_EXP
- #define SAVAPI_PLG_tchar_t char
- #define _T(x) x
- #define SAVAPI_PLG_MAX_STR 1024
Defines the maximum size for the plugin's information fields.
- #define SAVAPI_PLG_MAX_VER_STR 64
- #define SAVAPI_PLG_VER_MAJ 0 /** Savapi's plugins interface major version */
- #define SAVAPI_PLG_VER_MIN 1 /** Savapi's plugins interface minor version */

6.27.1 Detailed Description

6.27.2 Macro Definition Documentation

6.27.2.1 _T

```
#define _T(  
    x ) x
```

6.27.2.2 SAVAPI_PLG_EXP

```
#define SAVAPI_PLG_EXP
```

6.27.2.3 SAVAPI_PLG_MAX_STR

```
#define SAVAPI_PLG_MAX_STR 1024
```

Defines the maximum size for the plugin's information fields.

6.27.2.4 SAVAPI_PLG_MAX_VER_STR

```
#define SAVAPI_PLG_MAX_VER_STR 64
```

6.27.2.5 SAVAPI_PLG_tchar_t

```
#define SAVAPI_PLG_tchar_t char
```

6.27.2.6 SAVAPI_PLG_VER_MAJ

```
#define SAVAPI_PLG_VER_MAJ 0 /** Savapi's plugins interface major version */
```

6.27.2.7 SAVAPI_PLG_VER_MIN

```
#define SAVAPI_PLG_VER_MIN 1 /** Savapi's plugins interface minor version */
```

6.28 Plugin return statuses

This statuses are returned by the plugin routines.

Macros

- #define [SAVAPI_EPLG_SUCCESS](#) 0 /** Operation ended with success */
- #define [SAVAPI_EPLG_UKNW_INTERFACE](#) 1 /** The requested interface is not supported */
- #define [SAVAPI_EPLG_INVAL](#) 2 /** An invalid parameter has been provided */
- #define [SAVAPI_EPLG_INIT](#) 3 /** Plugin or instance already initialized */
- #define [SAVAPI_EPLG_NO_MEM](#) 4 /** Memory allocation error */
- #define [SAVAPI_EPLG_INTERNAL](#) 5 /** An internal error occurred */

6.28.1 Detailed Description

This statuses are returned by the plugin routines.

6.28.2 Macro Definition Documentation

6.28.2.1 SAVAPI_EPLG_INIT

```
#define SAVAPI_EPLG_INIT 3 /** Plugin or instance already initialized */
```

6.28.2.2 SAVAPI_EPLG_INTERNAL

```
#define SAVAPI_EPLG_INTERNAL 5 /** An internal error occurred */
```

6.28.2.3 SAVAPI_EPLG_INVALID

```
#define SAVAPI_EPLG_INVALID 2 /** An invalid parameter has been provided */
```

6.28.2.4 SAVAPI_EPLG_NO_MEM

```
#define SAVAPI_EPLG_NO_MEM 4 /** Memory allocation error */
```

6.28.2.5 SAVAPI_EPLG_SUCCESS

```
#define SAVAPI_EPLG_SUCCESS 0 /** Operation ended with success */
```

6.28.2.6 SAVAPI_EPLG_UKNW_INTERFACE

```
#define SAVAPI_EPLG_UKNW_INTERFACE 1 /** The requested interface is not supported */
```

6.29 Plugin's specific typedefs

Typedefs

- typedef void [SAVAPI_PLG_options_t](#)
- typedef void [SAVAPI_PLG_inst_options_t](#)
- typedef void [SAVAPI_PLG_instance_t](#)

6.29.1 Detailed Description

Remarks

Specific definition of this types are found in the specific header of each plugin.

6.29.2 Typedef Documentation

6.29.2.1 SAVAPI_PLG_inst_options_t

```
typedef void SAVAPI\_PLG\_inst\_options\_t
```

plugin's global init data

6.29.2.2 SAVAPI_PLG_instance_t

```
typedef void SAVAPI\_PLG\_instance\_t
```

plugin's instance init data

6.29.2.3 SAVAPI_PLG_options_t

```
typedef void SAVAPI\_PLG\_options\_t
```

6.30 Plugin's general typedefs

Typedefs

- typedef int [SAVAPI_PLG_status_t](#)
Plugin's return codes type.
- typedef [SAVAPI_PLG_status_t](#)(* [SAVAPI_PLG_init_t](#)) (const [SAVAPI_PLG_options_t](#) *options)
Prototype for a plugin's initialize function.
- typedef void(* [SAVAPI_PLG_uninit_t](#)) ()
Prototype for a plugin's uninitialize function.
- typedef [SAVAPI_PLG_status_t](#)(* [SAVAPI_PLG_instance_create_t](#)) ([SAVAPI_PLG_instance_t](#) **instance, const [SAVAPI_PLG_tchar_t](#) *interface_id, const [SAVAPI_PLG_inst_options_t](#) *options)
Prototype for a plugin's instance create function.
- typedef void(* [SAVAPI_PLG_instance_release_t](#)) ([SAVAPI_PLG_instance_t](#) **instance)
Prototype for a plugin's instance destroy function.

6.30.1 Detailed Description

6.30.2 Typedef Documentation

6.30.2.1 SAVAPI_PLG_init_t

```
typedef SAVAPI_PLG_status_t(* SAVAPI_PLG_init_t) (const SAVAPI_PLG_options_t *options)
```

Prototype for a plugin's initialize function.

6.30.2.2 SAVAPI_PLG_instance_create_t

```
typedef SAVAPI_PLG_status_t(* SAVAPI_PLG_instance_create_t) (SAVAPI_PLG_instance_t **instance,  
const SAVAPI_PLG_tchar_t *interface_id, const SAVAPI_PLG_inst_options_t *options)
```

Prototype for a plugin's instance create function.

6.30.2.3 SAVAPI_PLG_instance_release_t

```
typedef void(* SAVAPI_PLG_instance_release_t) (SAVAPI_PLG_instance_t **instance)
```

Prototype for a plugin's instance destroy function.

6.30.2.4 SAVAPI_PLG_status_t

```
typedef int SAVAPI_PLG_status_t
```

Plugin's return codes type.

6.30.2.5 SAVAPI_PLG_uninit_t

```
typedef void(* SAVAPI_PLG_uninit_t) ()
```

Prototype for a plugin's uninitialize function.

6.31 Plugin's structures definitions

Data Structures

- struct `_SAVAPI_PLG_info_t`
Structure containing the plugin's information (name, version etc.)

Macros

- #define `SAVAPI_PLG_MAIN_FUNC` `savapi_plg_main`

Typedefs

- typedef struct `_SAVAPI_PLG_info_t` `SAVAPI_PLG_info_t`
Structure containing the plugin's information (name, version etc.)
- typedef `SAVAPI_PLG_status_t`(* `SAVAPI_PLG_main_t`) (const `SAVAPI_PLG_info_t` **`savapi_plg_info`)
Main function that must be exported by the plugin.

Functions

- `SAVAPI_PLG_EXP SAVAPI_PLG_status_t SAVAPI_PLG_MAIN_FUNC` (const `SAVAPI_PLG_info_t` **`plg_info`)

6.31.1 Detailed Description

6.31.2 Macro Definition Documentation

6.31.2.1 SAVAPI_PLG_MAIN_FUNC

```
#define SAVAPI_PLG_MAIN_FUNC savapi_plg_main
```

6.31.3 Typedef Documentation

6.31.3.1 SAVAPI_PLG_info_t

```
typedef struct _SAVAPI_PLG_info_t SAVAPI_PLG_info_t
```

Structure containing the plugin's information (name, version etc.)

6.31.3.2 SAVAPI_PLG_main_t

```
typedef SAVAPI_PLG_status_t (* SAVAPI_PLG_main_t) (const SAVAPI_PLG_info_t **savapi_plg_info)
```

Main function that must be exported by the plugin.

6.31.4 Function Documentation

6.31.4.1 SAVAPI_PLG_MAIN_FUNC()

```
SAVAPI_PLG_EXP SAVAPI_PLG_status_t SAVAPI_PLG_MAIN_FUNC (
    const SAVAPI_PLG_info_t ** plg_info )
```

6.32 types for a SAVAPI FOPS plugin

Data Structures

- struct [_SAVAPI_PLG_fops_instance_t](#)

Macros

- `#define` [SAVAPI_PLG_AVE_FOPS_T](#)("SAVAPI_AVE_FOPS")

Typedefs

- typedef const char ** [SAVAPI_PLG_fops_options_t](#)
- typedef void [SAVAPI_PLG_fops_inst_options_t](#)
- typedef struct [_SAVAPI_PLG_fops_instance_t](#) [SAVAPI_PLG_fops_instance_t](#)

6.32.1 Detailed Description

6.32.2 Macro Definition Documentation

6.32.2.1 SAVAPI_PLG_AVE_FOPS

```
#define SAVAPI_PLG_AVE_FOPS_T("SAVAPI_AVE_FOPS")
```

6.32.3 Typedef Documentation

6.32.3.1 SAVAPI_PLG_fops_inst_options_t

```
typedef void SAVAPI_PLG_fops_inst_options_t
```

plugin's global init options

6.32.3.2 SAVAPI_PLG_fops_instance_t

```
typedef struct _SAVAPI_PLG_fops_instance_t SAVAPI_PLG_fops_instance_t
```

plugin's instance init options

6.32.3.3 SAVAPI_PLG_fops_options_t

```
typedef const char** SAVAPI_PLG_fops_options_t
```

6.33 SAVAPI STCHAR function pointers

Handle types for exported SAVAPI STCHAR functions.

Typedefs

- typedef SAVAPI_STATUS(* STCHARToChar_t) (char **pDest, const SAVAPI_TCHAR *pSrc)
- typedef SAVAPI_STATUS(* CharToSTCHAR_t) (SAVAPI_TCHAR **pDest, const char *pSrc)

6.33.1 Detailed Description

Handle types for exported SAVAPI STCHAR functions.

6.33.2 Typedef Documentation

6.33.2.1 CharToSTCHAR_t

```
typedef SAVAPI_STATUS(* CharToSTCHAR_t) (SAVAPI_TCHAR **pDest, const char *pSrc)
```

6.33.2.2 STCHARToChar_t

```
typedef SAVAPI_STATUS(* STCHARToChar_t) (char **pDest, const SAVAPI_TCHAR *pSrc)
```

Chapter 7

Data Structure Documentation

7.1 _AVE_STRUCT_FILE_OPERATIONS Struct Reference

The File OPerationS (FOPS) structure. Here are defined all the function prototypes which must be implemented in order to allow the scanning engine to process a special type of object.

```
#include <fops.h>
```

Data Fields

- `int(* fops_open)(FOPS_HANDLE *fh, void *filename, int mode, void *file_context, void *fops_context)`
Opens / creates a file.
- `int(* fops_close)(FOPS_HANDLE *fh, void *fops_context)`
Closes a file handle.
- `int(* fops_read)(FOPS_HANDLE fh, void *buffer, UINT64 count, UINT64 *nread, void *fops_context)`
Reads 'count' bytes from file 'fh' into 'buffer'.
- `int(* fops_write)(FOPS_HANDLE fh, void *buffer, UINT64 count, UINT64 *nwritten, void *fops_context)`
Writes 'count' bytes from 'buffer' into file 'fh'.
- `int(* fops_tell)(FOPS_HANDLE fh, INT64 *curpos, void *fops_context)`
Gets the current position in the file.
- `int(* fops_seek)(FOPS_HANDLE fh, INT64 offset, int wherefrom, void *fops_context)`
Sets the current file position to 'offset' (relative to 'wherefrom')
- `int(* fops_getfattr)(void *filename, _fattr_t *attr, void *file_context, void *fops_context)`
Gets the file's attributes.
- `int(* fops_setfattr)(void *filename, _fattr_t attr, void *file_context, void *fops_context)`
Sets the file's attributes.
- `int(* fops_getfsize)(FOPS_HANDLE fh, INT64 *fsize, void *fops_context)`
Gets the file's size.
- `int(* fops_unlink)(void *filename, void *file_context, void *fops_context)`
Deletes a file.
- `int(* fops_rename)(void *oldname, void *newname, void *file_context, void *fops_context)`
Renames a file.
- `int(* fops_access)(void *filename, int amode, void *file_context, void *fops_context)`
tries to access a file with a specific access
- `void *(* fops_malloc)(UINT64 size, void *fops_context)`

- allocates memory*
- void(* [fops_free](#))(void *ptr, void *fops_context)
- frees the memory allocated by fops_malloc*
- char *([fops_gets](#))(FOPS_HANDLE fh, char *str, int num, void *fops_context)
- Read a complete string from an opened file.*
- int(* [fops_puts](#))(FOPS_HANDLE fh, char *str, void *fops_context)
- Write a complete string to an opened file.*
- int(* [fops_getc](#))(FOPS_HANDLE fh, void *fops_context)
- Gets a character from an opened file.*
- int(* [fops_putc](#))(FOPS_HANDLE fh, int character, void *fops_context)
- Writes a character to a stream.*
- int(* [fops_ungetc](#))(FOPS_HANDLE fh, int character, void *fops_context)
- Ungets a character from the stream.*
- int(* [fops_flush](#))(FOPS_HANDLE fh, void *fops_context)
- Flushes pending file operations.*
- int(* [fops_get_last_error](#))(void *fops_context)
- Gets the last known error from the fops.*

7.1.1 Detailed Description

The File OPerationS (FOPS) structure. Here are defined all the function prototypes which must be implemented in order to allow the scanning engine to process a special type of object.

7.1.2 Field Documentation

7.1.2.1 fops_access

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_access) (void *filename, int amode, void *file_context, void *fops_context)
```

tries to access a file with a specific access

Parameters

<i>amode</i>	The mode to access the file. It can be one of 0 Existence only 2 Write-only 4 Read-only 6 Read and write
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

Returns 0 if the file is accessible with the given mode, 1 if it is not accessible

7.1.2.2 fops_close

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_close) (FOPS_HANDLE *fh, void *fops_context)
```

Closes a file handle.

Parameters

<i>fh</i>	The file handle to be closed
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful closed or non-zero if it failed.

7.1.2.3 fops_flush

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_flush) (FOPS_HANDLE fh, void *fops_context)
```

Flushes pending file operations.

Note

Flushes potentially pending operations, so parallel calls to fops_open/fops_read will result in up-to-date data to be read

Parameters

<i>fops_context</i>	The context of the scan. It can be anything..
---------------------	---

Returns

Returns 0 on success. On failure, -1 is returned and the stream remains unchanged.

7.1.2.4 fops_free

```
void (* _AVE_STRUCT_FILE_OPERATIONS::fops_free) (void *ptr, void *fops_context)
```

frees the memory allocated by fops_malloc

See also

[fops_malloc](#)

Parameters

<i>ptr</i>	the pointer to be released
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

Nothing

7.1.2.5 fops_get_last_error

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_get_last_error) (void *fops_context)
```

Gets the last known error from the fops.

Parameters

<i>fops_context</i>	The context of the scan. It It can be anything..
---------------------	--

Returns

The error code as defined by this fops (platform/fops-specific) 0 is no error

7.1.2.6 fops_getc

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_getc) (FOPS_HANDLE fh, void *fops_context)
```

Gets a character from an opened file.

Parameters

<i>fh</i>	The file handle to write to Returns the character currently pointed by the internal file position indicator of the specified stream. The internal file position indicator is then advanced by one character to point to the next character. fgetc and getc are equivalent, except that the latter one may be implemented as a macro.
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

The character read is returned as an int value. If the End-of-File is reached or a reading error occurs, the function returns -1

7.1.2.7 fops_getfattr

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_getfattr) (void *filename, _fattr_t *attr, void *file↵
_context, void *fops_context)
```

Gets the file's attributes.

Parameters

<i>filename</i>	The name of the file
<i>attr</i>	The actual file attributes, *attr must be NULL on error
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful or non-zero on error

7.1.2.8 fops_getfsize

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_getfsize) (FOPS_HANDLE fh, INT64 *fsize, void *fops_↵
context)
```

Gets the file's size.

Parameters

<i>fh</i>	The file handle
<i>fsize</i>	The file size returned
<i>file_context</i>	The file context, usually a handler(HANDLE, int, FILE*, etc.)
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

Zero if successful or non-zero on error

7.1.2.9 fops_gets

```
char * (* _AVE_STRUCT_FILE_OPERATIONS::fops_gets) (FOPS_HANDLE fh, char *str, int num, void ↵
*fops_context)
```

Read a complete string from an opened file.

Note

The string will end either when a 0 termination character or a carriage return is found.
See also fgets ANSI function. Note that there is no text mode behavior, only binary.

Parameters

<i>fh</i>	The file handle to read from
<i>str</i>	The buffer where to store the chars
<i>num</i>	The maximum length to read
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

The string (same as in str) or NULL if an error occurs

7.1.2.10 fops_malloc

```
void *(* _AVE_STRUCT_FILE_OPERATIONS::fops_malloc) (UINT64 size, void *fops_context)
```

allocates memory

Parameters

<i>size</i>	amount of bytes to be allocated
<i>file_context</i>	the file context, usually a handler(HANDLE, int, FILE*, etc.)
<i>fops_context</i>	The context of the scan. It can be anything..

Returns

Zero (NULL) if there is not enough memory is available or a pointer to the allocated memory if the memory was allocated

7.1.2.11 fops_open

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_open) (FOPS_HANDLE *fh, void *filename, int mode, void *file_context, void *fops_context)
```

Opens / creates a file.

Parameters

<i>fh</i>	Will be set to the handle of the file just opened
<i>filename</i>	The name of the file to be opened
<i>mode</i>	The file open mode. It can be one of OPEN_RO - open for read access OPEN_RW - open for read and write access (default implementation locks file) OPEN_CR - create or truncate the file (with read + write access, default implementation locks file)
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It can be anything. API version 5.5

Returns

Zero if successfully opened / created or non-zero if failed

7.1.2.12 fops_putc

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_putc) (FOPS_HANDLE fh, int character, void *fops_↵  
context)
```

Writes a character to a stream.

Note

Writes a single byte to the file at the current offset and advances the position indicator.

Parameters

<i>fh</i>	The file handle to write
<i>character</i>	The character to be written.
<i>fops_context</i>	The context of the scan. It can be anything..

Returns

If there are no errors, the same character that has been written is returned. If an error occurs, -1 is returned.

7.1.2.13 fops_puts

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_puts) (FOPS_HANDLE fh, char *str, void *fops_context)
```

Write a complete string to an opened file.

Note

The string will end when a 0 termination character is found.

See also fputs ANSI function. Note that there is no text mode behavior, only binary.

Parameters

<i>fh</i>	The file handle to write to
<i>str</i>	The buffer which contains the chars
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Returns on success, a non-negative value. On error, the function returns -1.

7.1.2.14 fops_read

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_read) (FOPS_HANDLE fh, void *buffer, UINT64 count,
UINT64 *nread, void *fops_context)
```

Reads 'count' bytes from file 'fh' into 'buffer'.

Parameters

<i>fh</i>	The file handle to read from
<i>buffer</i>	The buffer to write the bytes read from the handle
<i>count</i>	The amount of bytes to be read
<i>nread</i>	The number of bytes actually read, even in error case
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful or non-zero on error

Warning

*nread may be less than count on EOF condition

7.1.2.15 fops_rename

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_rename) (void *oldname, void *newname, void *file_↵
context, void *fops_context)
```

Renames a file.

Parameters

<i>oldname</i>	The old file name to be renamed
<i>newname</i>	The new file name to be set
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It can be anything..

Returns

Zero if successful or non-zero on error

7.1.2.16 fops_seek

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_seek) (FOPS_HANDLE fh, INT64 offset, int wherefrom, void *fops_context)
```

Sets the current file position to 'offset' (relative to 'wherefrom')

Parameters

<i>fh</i>	The file handle
<i>offset</i>	The position in the file (relative to 'wherefrom')
<i>wherefrom</i>	The position in the file. It may be one of SEEK_SET - set absolute file position SEEK_CUR - add offset to current position SEEK_END - seek offset bytes from end of file
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful or non-zero on error

7.1.2.17 fops_setfattr

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_setfattr) (void *filename, _fattr_t attr, void *file_↵ context, void *fops_context)
```

Sets the file's attributes.

Parameters

<i>filename</i>	The name of the file
<i>attr</i>	The attribute to be set. It may be S_IREAD S_IWRITE S_IREAD S_IWRITE
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful or non-zero on error

7.1.2.18 fops_tell

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_tell) (FOPS_HANDLE fh, INT64 *curpos, void *fops_↵  
context)
```

Gets the current position in the file.

Parameters

<i>fh</i>	The file handle
<i>curpos</i>	The current file position; it is set to -1 on error
<i>fops_context</i>	The context of the scan. It can be anything.

Returns

Zero if successful or non-zero on error

7.1.2.19 fops_ungetc

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_ungetc) (FOPS_HANDLE fh, int character, void *fops_↵  
context)
```

Ungets a character from the stream.

Note

A character is virtually put back into an input stream at the same position the last character was read and the internal file position indicator is decreased back to that previous position, so that this character is returned by the next call to a reading operation on that stream. This character **MUST** be the same character as the one last read from the stream in a previous operation. If it differs the resulting behaviour is **UNDEFINED**. The default fops as implemented by the engine supports submitting a different character just like the ansi ungetc, however this should not be relied upon in all circumstances since this provides a big obstacle to implementations that don't have access to equivalents of the ansi streaming file functions. If the End-Of-File internal indicator was set, it is cleared after a call to this function. If the argument passed for the character parameter is EOF, the operation fails.

Parameters

<i>fh</i>	The file handle to write to
<i>character</i>	The character to be put back. The character is passed as its int promotion.
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

If successful, the character that was pushed back is returned. On failure, -1 is returned and the stream remains unchanged.

7.1.2.20 fops_unlink

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_unlink) (void *filename, void *file_context, void *fops↵_context)
```

Deletes a file.

Parameters

<i>file_name</i>	The file name to be deleted
<i>file_context</i>	File-specific context, it's internally used by the SAVAPI Library and should not be modified by the apps.
<i>fops_context</i>	The context of the scan. It It can be anything..

Returns

Zero if successful or non-zero on error

7.1.2.21 fops_write

```
int (* _AVE_STRUCT_FILE_OPERATIONS::fops_write) (FOPS_HANDLE fh, void *buffer, UINT64 count,↵UINT64 *nwritten, void *fops_context)
```

Writes 'count' bytes from 'buffer' into file 'fh'.

Parameters

<i>fh</i>	The file handle to write to
<i>buffer</i>	The buffer which will be written into the handle
<i>count</i>	The amount of bytes to write
<i>nwritten</i>	The number of bytes actually written, even in error case
<i>fops_context</i>	The context of the scan. It can be anything.

Warning

*nwritten may be less than count if the disk is full

Returns

Zero if successful or non-zero on error

The documentation for this struct was generated from the following file:

- [fops.h](#)

7.2 SAVAPI_report_content_data::_content_data Union Reference

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_IFRAME_URL_DATA](#) * [iframeurl_data](#)

7.2.1 Detailed Description

Union used to switch the content data depending on the received 'type'

7.2.2 Field Documentation

7.2.2.1 [iframeurl_data](#)

```
SAVAPI\_IFRAME\_URL\_DATA* SAVAPI_report_content_data::_content_data::iframeurl_data
```

Information about the iframe report (url, attribute)

The documentation for this union was generated from the following file:

- [savapi.h](#)

7.3 _SAVAPI_PLG_fops_instance_t Struct Reference

```
#include <savapi_plg_fops.h>
```

Data Fields

- [AVE_FOPS](#) [fops](#)

7.3.1 Detailed Description

plugin's instance init options

7.3.2 Field Documentation

7.3.2.1 fops

`AVE_FOPS _SAVAPI_PLG_fops_instance_t::fops`

The documentation for this struct was generated from the following file:

- [savapi_plg_fops.h](#)

7.4 _SAVAPI_PLG_info_t Struct Reference

Structure containing the plugin's information (name, version etc.)

```
#include <savapi_plg.h>
```

Data Fields

- [SAVAPI_PLG_init_t](#) init
- [SAVAPI_PLG_uninit_t](#) uninit
- [SAVAPI_PLG_instance_create_t](#) instance_create
- [SAVAPI_PLG_instance_release_t](#) instance_release
- int [interface_ver_maj](#)
- int [interface_ver_min](#)
- [SAVAPI_PLG_tchar_t](#) name [[SAVAPI_PLG_MAX_STR](#)]
- [SAVAPI_PLG_tchar_t](#) version [[SAVAPI_PLG_MAX_VER_STR](#)]
- void * [res](#)

7.4.1 Detailed Description

Structure containing the plugin's information (name, version etc.)

7.4.2 Field Documentation

7.4.2.1 init

[SAVAPI_PLG_init_t](#) [_SAVAPI_PLG_info_t::init](#)

Global related routines

7.4.2.2 instance_create

[SAVAPI_PLG_instance_create_t](#) [_SAVAPI_PLG_info_t::instance_create](#)

global uninit

Instance related routines

7.4.2.3 instance_release

[SAVAPI_PLG_instance_release_t](#) [_SAVAPI_PLG_info_t::instance_release](#)

instance create

7.4.2.4 interface_ver_maj

[int](#) [_SAVAPI_PLG_info_t::interface_ver_maj](#)

instance release

Version related fields

7.4.2.5 interface_ver_min

[int](#) [_SAVAPI_PLG_info_t::interface_ver_min](#)

interface major version. Should be set to [SAVAPI_PLG_VER_MAJ](#)

7.4.2.6 name

[SAVAPI_PLG_tchar_t](#) [_SAVAPI_PLG_info_t::name](#)[[SAVAPI_PLG_MAX_STR](#)]

interface minor version. Should be set to [SAVAPI_PLG_VER_MIN](#)

7.4.2.7 res

[void*](#) [_SAVAPI_PLG_info_t::res](#)

plugin's version

7.4.2.8 uninit

[SAVAPI_PLG_uninit_t](#) [_SAVAPI_PLG_info_t::uninit](#)

global init

7.4.2.9 version

[SAVAPI_PLG_tchar_t](#) [_SAVAPI_PLG_info_t::version](#)[[SAVAPI_PLG_MAX_VER_STR](#)]

friendly name

The documentation for this struct was generated from the following file:

- [savapi_plg.h](#)

7.5 SAVAPI_report_scan_details_data::_scan_details_data Union Reference

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_ALERT_URL_DATA](#) * [alert_url_data](#)
- [SAVAPI_REPAIRABLE_DATA](#) * [repairable_data](#)

7.5.1 Detailed Description

Union used to switch the scan details data depending on the received 'type'

7.5.2 Field Documentation

7.5.2.1 alert_url_data

[SAVAPI_ALERT_URL_DATA](#)* [SAVAPI_report_scan_details_data::_scan_details_data::alert_url_data](#)

Contains the alert URL report

7.5.2.2 repairable_data

[SAVAPI_REPAIRABLE_DATA](#)* SAVAPI_report_scan_details_data::_scan_details_data::repairable_data

Contains the details about the reparable data

The documentation for this union was generated from the following file:

- [savapi.h](#)

7.6 SAVAPI_alert_url_data Struct Reference

Structure associated with the ALERTURL report.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * alert_url
- [SAVAPI_FILE_INFO](#) file_info

7.6.1 Detailed Description

Structure associated with the ALERTURL report.

7.6.2 Field Documentation

7.6.2.1 alert_url

[SAVAPI_TCHAR](#)* SAVAPI_alert_url_data::alert_url

Pointer to the string containing the alert URL

7.6.2.2 file_info

[SAVAPI_FILE_INFO](#) SAVAPI_alert_url_data::file_info

Information (name, type, level) about the scanned file

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.7 SAVAPI_APC_global_init Struct Reference

The structure used for initializing APC.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * [cert_dir](#)
- [SAVAPI_TCHAR](#) * [temp_dir](#)
- [SAVAPI_TCHAR](#) * [lib_dir](#)
- [SAVAPI_APC_SCAN_MODE](#) [apc_mode](#)
- [SAVAPI_SIZE_T](#) [cache_size](#)
- unsigned int [dump_cache_file](#)
- [SAVAPI_TCHAR](#) * [cache_file_path](#)
- unsigned int [blackout_retries](#)
- unsigned int [blackout_timeout](#)
- char * [proxy](#)

7.7.1 Detailed Description

The structure used for initializing APC.

7.7.2 Field Documentation

7.7.2.1 [apc_mode](#)

```
SAVAPI\_APC\_SCAN\_MODE SAVAPI_APC_global_init::apc_mode
```

APC scan mode

7.7.2.2 [blackout_retries](#)

```
unsigned int SAVAPI_APC_global_init::blackout_retries
```

The maximum number of timeouts allowed before declaring APC unreachable.

Note

If 0, then SAVAPI will always try to reach APC.

7.7.2.3 blackout_timeout

`unsigned int SAVAPI_APC_global_init::blackout_timeout`

Number of seconds after which SAVAPI will try another connection to APC.

Note

Accepted range is 1 - INT16_MAX

7.7.2.4 cache_file_path

`SAVAPI_TCHAR* SAVAPI_APC_global_init::cache_file_path`

Path for cache file dump (optional)

Note

It must be unique for each library implementation, otherwise conflicts may appear

If `dump_cache_file` is set to disabled, this parameter is ignored

If this variable is set to NULL or empty string, cache file (`savapi_apc_cache.dat`) will be saved in default temporary directory

7.7.2.5 cache_size

`SAVAPI_SIZE_T SAVAPI_APC_global_init::cache_size`

APC cache size

Note

If 0, the cache will be disabled

The size of the cache will greatly affect the time needed by the APC to finish processing the requests. The more size available, the more data can be stored and used later, in order to save bandwidth. For high-intensive applications, a bigger value is recommended.

Recommended size is 5242880 (5MB)

7.7.2.6 cert_dir

`SAVAPI_TCHAR* SAVAPI_APC_global_init::cert_dir`

Path to the APC certificate directory (optional)

Note

If this variable is set to NULL or empty string, the current working directory will be used instead

7.7.2.7 dump_cache_file

unsigned int SAVAPI_APC_global_init::dump_cache_file

Enable/disable cache file dump (1 = enabled, 0 = disabled)

7.7.2.8 lib_dir

SAVAPI_TCHAR* SAVAPI_APC_global_init::lib_dir

Path to the directory containing the APC library (optional)

Note

If this variable is set to NULL or empty string, the current working directory will be used instead

7.7.2.9 proxy

char* SAVAPI_APC_global_init::proxy

APC connection proxy server

Note

The parameter holds the host name or IP address. To specify a port number in this string, append :[port] to the end of the host name. If not specified, SAVAPI will use as default the port 1080.

The proxy string may be prefixed with [scheme]:// to specify the kind of proxy to be used. Supported schemes are: [http://](#), [https://](#), [socks4://](#), [socks4a://](#) and [socks5://](#).

If no protocol is specified, the proxy will be treated as a HTTP proxy server.

Only ASCII characters are supported.

If the proxy requires authentication, the credentials can be specified by adding username:password before the host name, as shown in one of the examples below.

If no proxy is provided, SAVAPI will try to read it from other sources: environment variables in the following order: https_proxy, HTTPS_PROXY, http_proxy, HTTP_PROXY, all_proxy, ALL_PROXY.

Examples: Proxy=10.0.0.1:3128 Proxy= [http://proxy-server:3128](#) Proxy=socks4://socks-proxy-server Proxy= [http://username:password@proxy-server:3128](#)

7.7.2.10 temp_dir

SAVAPI_TCHAR* SAVAPI_APC_global_init::temp_dir

Path to the temporary directory to be used by APC (optional)

Note

If this variable is set to NULL or empty string, the default system temporary directory will be used instead

It is recommended for the location to not be a directory that contains sensitive files, such as SAVAPI binaries or configuration files.

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.8 SAVAPI_APC_REPORT_DATA Struct Reference

Information to report when calling a [APC_set_report_info_t](#) function.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_APC_SCAN_ANSWER](#) `scan_answer`
- [SAVAPI_MALWARE_INFO](#) `malware_info`
- unsigned int `store_cache`
- [SAVAPI_SIZE_T](#) `ttl`

7.8.1 Detailed Description

Information to report when calling a [APC_set_report_info_t](#) function.

7.8.2 Field Documentation

7.8.2.1 `malware_info`

[SAVAPI_MALWARE_INFO](#) `SAVAPI_APC_REPORT_DATA::malware_info`

Information about the reported malware

Note

This structure is read only if `scan_answer` is [SAVAPI_APC_ANSWER_INFECTED](#)

7.8.2.2 `scan_answer`

[SAVAPI_APC_SCAN_ANSWER](#) `SAVAPI_APC_REPORT_DATA::scan_answer`

File scan answer

7.8.2.3 `store_cache`

unsigned int `SAVAPI_APC_REPORT_DATA::store_cache`

Store the result in cache or not (1 = enabled, 0 = disabled)

Note

The result cannot be stored in cache if the hash received on the [SAVAPI_APC_SCAN_DATA](#) structure is unavailable

7.8.2.4 ttl

[SAVAPI_SIZE_T](#) SAVAPI_APC_REPORT_DATA::ttl

For how many seconds should this detection remain valid

Note

This field is read only if [store_cache](#) is enabled

If 0, a default value of 600 (10 minutes) will be applied.

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.9 SAVAPI_apc_scan_data Struct Reference

The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_FILE_INFO](#) file_info
- [SAVAPI_APC_SCAN_STAGE](#) stage
- char * hash
- int risk_rating_level
- void * fops_handle
- void * fops_context
- unsigned int flags
- [SAVAPI_FD](#) savapi_fd
- [APC_set_report_info_t](#) set_report_info

7.9.1 Detailed Description

The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.

7.9.2 Field Documentation

7.9.2.1 file_info

[SAVAPI_FILE_INFO](#) SAVAPI_apc_scan_data::file_info

Information (name, type, level) about the scanned file

7.9.2.2 flags

```
unsigned int SAVAPI_apc_scan_data::flags
```

General purpose flags field

7.9.2.3 fops_context

```
void* SAVAPI_apc_scan_data::fops_context
```

FOPS context for the current file

7.9.2.4 fops_handle

```
void* SAVAPI_apc_scan_data::fops_handle
```

FOPS handle for the current file (of type [FOPS_HANDLE](#))

7.9.2.5 hash

```
char* SAVAPI_apc_scan_data::hash
```

The hash of the scanned file

Note

This field is unavailable if stage is [SAVAPI_APC_STAGE_PRE_FILTER](#) or if stage is [SAVAPI_APC_STAGE_POST_SCAN](#) and the file was filtered

7.9.2.6 risk_rating_level

```
int SAVAPI_apc_scan_data::risk_rating_level
```

A value between 0 and 7 which states the risk of the current file containing malware:

- 0 -> file has very low risk;
- 7 -> file has very high risk.

Note

A value of -1 means that no rating level is provided for this file.

7.9.2.7 savapi_fd

`SAVAPI_FD SAVAPI_apc_scan_data::savapi_fd`

Handle to the SAVAPI instance, needed for [set_report_info](#)

7.9.2.8 set_report_info

`APC_set_report_info_t SAVAPI_apc_scan_data::set_report_info`

Function to be called when new information is found about the current scanned file

Note

The information given by calling this function will be used only if [SAVAPI_APC_SCAN_REPORT](#) is returned in the [SAVAPI_CALLBACK_APC_SCAN](#) callback

7.9.2.9 stage

`SAVAPI_APC_SCAN_STAGE SAVAPI_apc_scan_data::stage`

The stage of the APC scan in which the callback was called

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.10 SAVAPI_archive_open_data Struct Reference

Contains the data sent to a `archive_open` callback.

```
#include <savapi.h>
```

Data Fields

- unsigned int [flags](#)
- [SAVAPI_FILE_INFO](#) `file_info`

7.10.1 Detailed Description

Contains the data sent to a `archive_open` callback.

7.10.2 Field Documentation

7.10.2.1 file_info

[SAVAPI_FILE_INFO](#) SAVAPI_archive_open_data::file_info

Information (name, type, level) about the archive to be opened

7.10.2.2 flags

unsigned int SAVAPI_archive_open_data::flags

General purpose flags field.

Note

Currently defined flags: see [Filename flags](#) group.

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.11 SAVAPI_callback_data Struct Reference

Structure passed by SAVAPI to a user defined callback, containing all the necessary data.

```
#include <savapi.h>
```

Data Structures

- union [specific_data](#)
Callbacks specific data.

Data Fields

- unsigned int [type](#)
- unsigned int [version](#)
- unsigned int [flags](#)
- void * [user_data](#)
User custom data.
- union [SAVAPI_callback_data::specific_data](#) [callback_data](#)

7.11.1 Detailed Description

Structure passed by SAVAPI to a user defined callback, containing all the necessary data.

7.11.2 Field Documentation

7.11.2.1 callback_data

```
union SAVAPI_callback_data::specific_data SAVAPI_callback_data::callback_data
```

7.11.2.2 flags

```
unsigned int SAVAPI_callback_data::flags
```

Reserved

7.11.2.3 type

```
unsigned int SAVAPI_callback_data::type
```

The callback id. See [Callbacks' ids](#)

7.11.2.4 user_data

```
void* SAVAPI_callback_data::user_data
```

User custom data.

Note

SAVAPI will not make any assumption regarding this field. It will just be passed back to callback function

7.11.2.5 version

```
unsigned int SAVAPI_callback_data::version
```

The callback version

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.12 SAVAPI_command_data Struct Reference

The structure to be passed when sending a command.

```
#include <savapi.h>
```

Data Fields

- unsigned int [signal_id](#)
- void * [command_data](#)

Signal specific data.

7.12.1 Detailed Description

The structure to be passed when sending a command.

7.12.2 Field Documentation

7.12.2.1 command_data

```
void* SAVAPI_command_data::command_data
```

Signal specific data.

Note

Currently SAVAPI has defined only [SAVAPI_SIGNAL_SCAN_ABORT](#) signal which doesn't require any data. Thus, "specific_data" field is currently empty.

Todo Add specific date as soon as new signals, which require data will be defined.

7.12.2.2 signal_id

```
unsigned int SAVAPI_command_data::signal_id
```

signal id. See [SAVAPI signals](#)

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.13 SAVAPI_error_data Struct Reference

The structure associated with report error callback.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_FILE_INFO](#) `file_info`
- unsigned int `category`
- unsigned int `level`
- int `code`
- [SAVAPI_KEY_VALUE](#) * `options`

7.13.1 Detailed Description

The structure associated with report error callback.

The callback is triggered each time an error occurred on scanning process (an I/O error for instance). Also the callback can be called if warnings or infos reports during scanning are activated.

Note

See [SAVAPI_CALLBACK_REPORT_ERROR](#)

7.13.2 Field Documentation

7.13.2.1 category

```
unsigned int SAVAPI_error_data::category
```

error category see [Error or information categories](#)

7.13.2.2 code

```
int SAVAPI_error_data::code
```

error code. See [SAVAPI return codes](#)

Note

If error level is not `SAVAPI_ELEVEL_ERROR` this field contains flags. See [Scan warnings](#) and [Scan information](#)

7.13.2.3 file_info

`SAVAPI_FILE_INFO SAVAPI_error_data::file_info`

Information (name, type, level) about the file where the error occurred

7.13.2.4 level

`unsigned int SAVAPI_error_data::level`

error level see [Error levels](#)

7.13.2.5 options

`SAVAPI_KEY_VALUE* SAVAPI_error_data::options`

The container contain currently only the error code string.

Note

The [code](#) is the id within container

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.14 SAVAPI_file_info Struct Reference

Contains data about the scanned file.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * [name](#)
- unsigned int [type](#)
- unsigned int [level](#)

7.14.1 Detailed Description

Contains data about the scanned file.

7.14.2 Field Documentation

7.14.2.1 level

```
unsigned int SAVAPI_file_info::level
```

The file recursion level (0 for regular files, +1 for each level in an archive)

7.14.2.2 name

```
SAVAPI_TCHAR* SAVAPI_file_info::name
```

file name

7.14.2.3 type

```
unsigned int SAVAPI_file_info::type
```

The file type. See [File types](#).

Note

Can be one or more types (ex: file is an archive and is found in an archive).

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.15 SAVAPI_file_status_data Struct Reference

Contains the data sent to a report file status callback.

```
#include <savapi.h>
```

Data Fields

- unsigned int [flags](#)
- unsigned int [scan_answer](#)
- [SAVAPI_FILE_INFO](#) [file_info](#)
- [SAVAPI_MALWARE_INFO](#) [malware_info](#)
Malware information (name, type, etc). See [SAVAPI_malware_info](#) for more details.
- unsigned int [warning](#)
- unsigned int [info](#)

7.15.1 Detailed Description

Contains the data sent to a report file status callback.

Note

See [SAVAPI_CALLBACK_REPORT_FILE_STATUS](#)

7.15.2 Field Documentation

7.15.2.1 file_info

[SAVAPI_FILE_INFO](#) SAVAPI_file_status_data::file_info

File information (name, type, level). See [SAVAPI_file_info](#) for more details

7.15.2.2 flags

unsigned int SAVAPI_file_status_data::flags

General purpose flags field.

Note

Currently defined flags: check [Filename flags](#) group

7.15.2.3 info

unsigned int SAVAPI_file_status_data::info

additional info to report See [Scan information](#)

7.15.2.4 malware_info

[SAVAPI_MALWARE_INFO](#) SAVAPI_file_status_data::malware_info

Malware information (name, type, etc). See [SAVAPI_malware_info](#) for more details.

Note

Contains data only if the object processed is not clean.

7.15.2.5 scan_answer

unsigned int SAVAPI_file_status_data::scan_answer

File scan answer. See [SAVAPI scan statuses](#) for available values

7.15.2.6 warning

```
unsigned int SAVAPI_file_status_data::warning
```

warning value to report See [Scan warnings](#)

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.16 SAVAPI_global_init Struct Reference

The structure used at SAVAPI initialization.

```
#include <savapi.h>
```

Data Fields

- unsigned int [api_major_version](#)
- unsigned int [api_minor_version](#)
- unsigned int [program_type](#)
- [SAVAPI_TCHAR](#) * [engine_dirpath](#)
- [SAVAPI_TCHAR](#) * [vdfs_dirpath](#)
- [SAVAPI_TCHAR](#) * [avll_dirpath](#)
- [SAVAPI_TCHAR](#) * [key_file_name](#)

7.16.1 Detailed Description

The structure used at SAVAPI initialization.

7.16.2 Field Documentation

7.16.2.1 api_major_version

```
unsigned int SAVAPI_global_init::api_major_version
```

The expected API major version ([SAVAPI_API_MAJOR_VERSION](#))

7.16.2.2 api_minor_version

```
unsigned int SAVAPI_global_init::api_minor_version
```

The expected API minor version ([SAVAPI_API_MINOR_VERSION](#))

7.16.2.3 avll_dirpath

[SAVAPI_TCHAR*](#) SAVAPI_global_init::avll_dirpath

IGNORED OPTION - Path to a directory containing avll license library

7.16.2.4 engine_dirpath

[SAVAPI_TCHAR*](#) SAVAPI_global_init::engine_dirpath

Path to a directory containing engine modules (optional)

Note

If this variable is set to NULL or empty string, the current working directory will be used instead

7.16.2.5 key_file_name

[SAVAPI_TCHAR*](#) SAVAPI_global_init::key_file_name

The path to the license file (optional)

Note

If this field is set to NULL or empty string, the default values of current_dir/HBEDV.KEY or current_dir/hbedv.↔ key will be used instead, if any of them exists

If the path is not absolute, it will be considered relative to the current working directory

7.16.2.6 program_type

unsigned int SAVAPI_global_init::program_type

The unique program number which identifies the 3rd party application for the license checking function. This has to be requested from Avira.

7.16.2.7 vdfs_dirpath

[SAVAPI_TCHAR*](#) SAVAPI_global_init::vdfs_dirpath

Path to a directory containing the signature files (optional)

Note

If this variable is set to NULL or empty string, the same directory as [engine_dirpath](#) will be used

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.17 SAVAPI_iframe_url_data Struct Reference

Structure associated with the iframe report.

```
#include <savapi.h>
```

Data Fields

- unsigned int [attribute](#)
- [SAVAPI_TCHAR](#) * [url](#)

7.17.1 Detailed Description

Structure associated with the iframe report.

Note

This structure is deprecated.

7.17.2 Field Documentation

7.17.2.1 attribute

```
unsigned int SAVAPI_iframe_url_data::attribute
```

iframe attribute. See [Iframes informations](#)

7.17.2.2 url

```
SAVAPI\_TCHAR* SAVAPI_iframe_url_data::url
```

iframe url.

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.18 SAVAPI_instance_init Struct Reference

The structure used at SAVAPI instance creation.

```
#include <savapi.h>
```

Data Fields

- unsigned int [flags](#)
- unsigned int [connection_timeout](#)
- [SAVAPI_TCHAR](#) * [host_name](#)
- unsigned int [port](#)
- unsigned int [scan_timeout](#)
- unsigned int [get_timeout](#)
- unsigned int [set_timeout](#)
- [SAVAPI_TCHAR](#) * [username](#)
- [SAVAPI_TCHAR](#) * [password](#)

7.18.1 Detailed Description

The structure used at SAVAPI instance creation.

7.18.2 Field Documentation

7.18.2.1 `connection_timeout`

```
unsigned int SAVAPI_instance_init::connection_timeout
```

Specified the connection timeout in milliseconds.

Note

Available only in client-mode.

7.18.2.2 `flags`

```
unsigned int SAVAPI_instance_init::flags
```

Initialization flags, right now only the flag deciding the connection type is defined. See [Initialization flags](#)

7.18.2.3 `get_timeout`

```
unsigned int SAVAPI_instance_init::get_timeout
```

Specifies the timeout (in milliseconds) of the get options operation.

Note

Used only in client-mode.

7.18.2.4 host_name

`SAVAPI_TCHAR* SAVAPI_instance_init::host_name`

Specifies the machine on which the SAVAPI daemon is located.

Note

Used only in client-mode.

7.18.2.5 password

`SAVAPI_TCHAR* SAVAPI_instance_init::password`

This field is deprecated.

Note

Any data contained by this field will be ignored.

7.18.2.6 port

`unsigned int SAVAPI_instance_init::port`

Specifies the port on which to connect to the daemon.

Note

Used in the same conditions as [host_name](#).

7.18.2.7 scan_timeout

`unsigned int SAVAPI_instance_init::scan_timeout`

This field is deprecated.

Note

Any data contained by this field will be ignored.

7.18.2.8 set_timeout

```
unsigned int SAVAPI_instance_init::set_timeout
```

Specifies the timeout (in milliseconds) of the set options operation.

Note

Used only in client-mode.

7.18.2.9 username

```
SAVAPI_TCHAR* SAVAPI_instance_init::username
```

This field is deprecated.

Note

Any data contained by this field will be ignored.

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.19 SAVAPI_key_value Struct Reference

Generic container.

```
#include <savapi.h>
```

Data Fields

- unsigned int [id](#)
- unsigned int [type](#)
- char * [value](#)

7.19.1 Detailed Description

Generic container.

This kind of container is very useful in case of need to store many options. It offers a very elegant encapsulation and a very high flexibility (the user will not know how data will be stored).

The elements from container are accessed using a key ([SAVAPI_key_value::id](#) member). The element is accessed through [SAVAPI_key_value::value](#) member and its type through [SAVAPI_key_value::type](#) member

7.19.2 Field Documentation

7.19.2.1 id

```
unsigned int SAVAPI_key_value::id
```

The element associated id

7.19.2.2 type

```
unsigned int SAVAPI_key_value::type
```

The element type

7.19.2.3 value

```
char* SAVAPI_key_value::value
```

The element value

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.20 SAVAPI_malware_info Struct Reference

Contains data about the found malware in an infected/suspicious file.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * [name](#)
- [SAVAPI_TCHAR](#) * [type](#)
- [SAVAPI_TCHAR](#) * [message](#)
- [SAVAPI_TCHAR](#) * [app_flags](#)
- unsigned int [removable](#)
- unsigned short [strict](#)

7.20.1 Detailed Description

Contains data about the found malware in an infected/suspicious file.

7.20.2 Field Documentation

7.20.2.1 app_flags

`SAVAPI_TCHAR* SAVAPI_malware_info::app_flags`

This field is deprecated and should be ignored

7.20.2.2 message

`SAVAPI_TCHAR* SAVAPI_malware_info::message`

Additional information about found malware

7.20.2.3 name

`SAVAPI_TCHAR* SAVAPI_malware_info::name`

The malware name or null if file is clean

7.20.2.4 removable

`unsigned int SAVAPI_malware_info::removable`

Set to 1 if the scanned file is infected by a file infector which can be repaired / Set to 0 otherwise

7.20.2.5 strict

`unsigned short SAVAPI_malware_info::strict`

This field is deprecated and should be ignored

7.20.2.6 type

`SAVAPI_TCHAR* SAVAPI_malware_info::type`

The malware type. Can have the following values: adware, backdoor, constructor, dialer, dropper, exploit, game, heuristic, joke, macro, packer, phishing, program, riskware, script, trash, trojan, virus, worm. Additionally, there is a dynamic list of types from APC, which start with "APC/" prefix

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.21 SAVAPI_OA_file_result_data Struct Reference

Contains the data sent to an OnAccess file status callback.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * [filename](#)
- unsigned int [pid](#)
- unsigned char * [sid](#)
- unsigned long [sid_len](#)

7.21.1 Detailed Description

Contains the data sent to an OnAccess file status callback.

Note

See [SAVAPI_CALLBACK_OA_FILE_RESULT](#)

7.21.2 Field Documentation

7.21.2.1 filename

```
SAVAPI\_TCHAR* SAVAPI_OA_file_result_data::filename
```

name of the file for which the allow/block decision needs to be made

7.21.2.2 pid

```
unsigned int SAVAPI_OA_file_result_data::pid
```

process ID of the process which opened the file

7.21.2.3 sid

```
unsigned char* SAVAPI_OA_file_result_data::sid
```

security ID of the user who opened the file

7.21.2.4 sid_len

```
unsigned long SAVAPI_OA_file_result_data::sid_len
```

the length of the buffer containing the security ID

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.22 SAVAPI_OA_global_init Struct Reference

The structure used for initializing OnAccess.

```
#include <savapi.h>
```

Data Fields

- unsigned int [threads_number](#)
Number of threads to be used by OnAccess scanner.
- unsigned int [scm_pending_time](#)
SCM pending time.

7.22.1 Detailed Description

The structure used for initializing OnAccess.

Note

Only supported on Windows.

7.22.2 Field Documentation

7.22.2.1 scm_pending_time

```
unsigned int SAVAPI_OA_global_init::scm_pending_time
```

SCM pending time.

Note

In case of a service, the estimated time required, in milliseconds, for a pending start, stop, pause, or continue operation. Set it to 0 if the library will not run inside a service.

Available values: 0 or any value bigger than or equal to 5000

Default value: 0

7.22.2.2 threads_number

```
unsigned int SAVAPI_OA_global_init::threads_number
```

Number of threads to be used by OnAccess scanner.

Note

Available values: 3 - 16

If this value is set to 0, the default value will be used

Default value: 16

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.23 SAVAPI_pre_scan_data Struct Reference

Contains the data sent to a prescan callback.

```
#include <savapi.h>
```

Data Fields

- unsigned int [flags](#)
- [SAVAPI_FILE_INFO](#) [file_info](#)

7.23.1 Detailed Description

Contains the data sent to a prescan callback.

7.23.2 Field Documentation

7.23.2.1 file_info

```
SAVAPI\_FILE\_INFO SAVAPI_pre_scan_data::file_info
```

Information (name, type, level) about the scanned file

7.23.2.2 flags

```
unsigned int SAVAPI_pre_scan_data::flags
```

General purpose flags field.

Note

Currently defined flags: see [Filename flags](#) group

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.24 SAVAPI_repairable_data Struct Reference

Structure associated with the REPAIRABLE report.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_FILE_INFO](#) `file_info`
- [SAVAPI_MALWARE_INFO](#) `malware_info`

7.24.1 Detailed Description

Structure associated with the REPAIRABLE report.

7.24.2 Field Documentation

7.24.2.1 file_info

```
SAVAPI\_FILE\_INFO SAVAPI_repairable_data::file_info
```

Information (name, type, level) about the scanned file

7.24.2.2 malware_info

```
SAVAPI\_MALWARE\_INFO SAVAPI_repairable_data::malware_info
```

Malware information (name, type, etc).

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.25 SAVAPI_report_content_data Struct Reference

The structure associated with report content callback.

```
#include <savapi.h>
```

Data Structures

- union [_content_data](#)

Data Fields

- unsigned int [flags](#)
- unsigned int [type](#)
- [SAVAPI_FILE_INFO](#) [file_info](#)
- union [SAVAPI_report_content_data::_content_data](#) [content_data](#)

7.25.1 Detailed Description

The structure associated with report content callback.

Note

This structure is deprecated.

7.25.2 Field Documentation

7.25.2.1 content_data

```
union SAVAPI\_report\_content\_data::\_content\_data SAVAPI\_report\_content\_data::content\_data
```

7.25.2.2 file_info

```
SAVAPI\_FILE\_INFO SAVAPI\_report\_content\_data::file\_info
```

Information (name, type, level) about the scanned file

7.25.2.3 flags

```
unsigned int SAVAPI\_report\_content\_data::flags
```

Reserved

7.25.2.4 type

```
unsigned int SAVAPI_report_content_data::type
```

report type ([SAVAPI report content types](#))

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.26 SAVAPI_report_progress_data Struct Reference

The structure associated with report progress callback.

```
#include <savapi.h>
```

Data Fields

- unsigned int [flags](#)
- [SAVAPI_TCHAR](#) * [message](#)

7.26.1 Detailed Description

The structure associated with report progress callback.

7.26.2 Field Documentation

7.26.2.1 flags

```
unsigned int SAVAPI_report_progress_data::flags
```

Reserved

7.26.2.2 message

```
SAVAPI\_TCHAR* SAVAPI_report_progress_data::message
```

the progress message

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.27 SAVAPI_report_scan_details_data Struct Reference

The structure associated with report scan details callback.

```
#include <savapi.h>
```

Data Structures

- union [_scan_details_data](#)

Data Fields

- unsigned int [flags](#)
- unsigned int [type](#)
- union [SAVAPI_report_scan_details_data::_scan_details_data](#) [scan_details_data](#)

7.27.1 Detailed Description

The structure associated with report scan details callback.

7.27.2 Field Documentation

7.27.2.1 flags

```
unsigned int SAVAPI_report_scan_details_data::flags
```

Reserved

7.27.2.2 scan_details_data

```
union SAVAPI_report_scan_details_data::_scan_details_data SAVAPI_report_scan_details_data↔  
::scan_details_data
```

7.27.2.3 type

```
unsigned int SAVAPI_report_scan_details_data::type
```

report type(see [SAVAPI report scan details types](#))

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.28 SAVAPI_signal_data Struct Reference

The structure to be passed when sending a signal.

```
#include <savapi.h>
```

Data Fields

- unsigned int [signal_id](#)
- void * [signal_data](#)

Signal specific data.

7.28.1 Detailed Description

The structure to be passed when sending a signal.

7.28.2 Field Documentation

7.28.2.1 signal_data

```
void* SAVAPI_signal_data::signal_data
```

Signal specific data.

Note

Currently SAVAPI has defined only [SAVAPI_SIGNAL_SCAN_ABORT](#) signal which doesn't require any data. Thus, "specific_data" field is currently empty.

Todo Add specific date as soon as new signals, which require data will be defined.

7.28.2.2 signal_id

```
unsigned int SAVAPI_signal_data::signal_id
```

signal id. See [SAVAPI signals](#)

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.29 SAVAPI_simple_scan_file_data Struct Reference

The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_TCHAR](#) * [name](#)
- unsigned int [scan_answer](#)
- unsigned int [error_level](#)
- unsigned int [error_code](#)
- [SAVAPI_TCHAR](#) * [malware_name](#)
- [SAVAPI_TCHAR](#) * [malware_type](#)

7.29.1 Detailed Description

The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).

7.29.2 Field Documentation

7.29.2.1 error_code

```
unsigned int SAVAPI_simple_scan_file_data::error_code
```

error code. See [SAVAPI return codes](#)

Note

If error level is not [SAVAPI_ELEVEL_ERROR](#) this field contains flags. See [Scan warnings](#) and [Scan information](#)

7.29.2.2 error_level

```
unsigned int SAVAPI_simple_scan_file_data::error_level
```

error level see [Error levels](#)

Note

when [scan_answer](#) is [SAVAPI_SCAN_STATUS_ERROR](#) this should be checked together with the error code

7.29.2.3 malware_name

`SAVAPI_TCHAR* SAVAPI_simple_scan_file_data::malware_name`

The malware name or null if file is clean

7.29.2.4 malware_type

`SAVAPI_TCHAR* SAVAPI_simple_scan_file_data::malware_type`

The malware type. Can have the following values: adware, backdoor, constructor, dialer, dropper, exploit, game, heuristic, joke, macro, packer, phishing, program, riskware, script, trash, trojan, virus, worm. Additionally, there is a dynamic list of types from APC, which start with "APC/" prefix

7.29.2.5 name

`SAVAPI_TCHAR* SAVAPI_simple_scan_file_data::name`

file name

7.29.2.6 scan_answer

`unsigned int SAVAPI_simple_scan_file_data::scan_answer`

File scan answer. See [SAVAPI scan statuses](#) for available values

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.30 SAVAPI_simple_scan_output Struct Reference

The structure containing the output for [SAVAPI_simple_scan](#).

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_SIMPLE_SCAN_FILE_DATA](#) * files
- unsigned int count
- [SAVAPI_SIMPLE_SCAN_STATISTICS](#) stats

7.30.1 Detailed Description

The structure containing the output for [SAVAPI_simple_scan](#).

Note

Memory management is done by SAVAPI so there is no need to allocate or free the structure.

7.30.2 Field Documentation

7.30.2.1 count

```
unsigned int SAVAPI_simple_scan_output::count
```

number of items in the array containing files that are infected, suspicious or encountered errors

7.30.2.2 files

```
SAVAPI_SIMPLE_SCAN_FILE_DATA* SAVAPI_simple_scan_output::files
```

array of files that were found to be infected, suspicious, erroneous or for which there is additional information (example: office macros)

7.30.2.3 stats

```
SAVAPI_SIMPLE_SCAN_STATISTICS SAVAPI_simple_scan_output::stats
```

simple scan statistics

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.31 SAVAPI_simple_scan_statistics Struct Reference

The structure contains statistics for the simple scan.

```
#include <savapi.h>
```

Data Fields

- unsigned int [total_files](#)
- unsigned int [infections](#)
- unsigned int [suspicious](#)
- unsigned int [errors](#)

7.31.1 Detailed Description

The structure contains statistics for the simple scan.

7.31.2 Field Documentation

7.31.2.1 errors

```
unsigned int SAVAPI_simple_scan_statistics::errors
```

number of errors, warnings, or additional information (can be more than one per file)

7.31.2.2 infections

```
unsigned int SAVAPI_simple_scan_statistics::infections
```

number of infections detected (can be more than one per file)

7.31.2.3 suspicions

```
unsigned int SAVAPI_simple_scan_statistics::suspicions
```

number of suspicions detected (can be more than one per file)

7.31.2.4 total_files

```
unsigned int SAVAPI_simple_scan_statistics::total_files
```

Total number of files that were scanned

Note

For client mode this statistic parameter is not available and will always be 0

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.32 SAVAPI_version Struct Reference

The structure used to retrieve SAVAPI version.

```
#include <savapi.h>
```


Data Fields

- unsigned int [major](#)
- unsigned int [minor](#)
- unsigned int [build_major](#)
- unsigned int [build_minor](#)

7.32.1 Detailed Description

The structure used to retrieve SAVAPI version.

7.32.2 Field Documentation

7.32.2.1 build_major

```
unsigned int SAVAPI_version::build_major
```

Major version of the build

7.32.2.2 build_minor

```
unsigned int SAVAPI_version::build_minor
```

Minor version of the build

7.32.2.3 major

```
unsigned int SAVAPI_version::major
```

Major version of the product

7.32.2.4 minor

```
unsigned int SAVAPI_version::minor
```

Minor version of the product

The documentation for this struct was generated from the following file:

- [savapi.h](#)

7.33 SAVAPI_callback_data::specific_data Union Reference

Callbacks specific data.

```
#include <savapi.h>
```

Data Fields

- [SAVAPI_PRESCAN_DATA](#) * [pre_scan_data](#)
- [SAVAPI_ARCHIVE_OPEN_DATA](#) * [archive_open_data](#)
- [SAVAPI_FILE_STATUS_DATA](#) * [file_status_data](#)
- [SAVAPI_ERROR_DATA](#) * [error_data](#)
- [SAVAPI_REPORT_PROGRESS_DATA](#) * [report_progress_data](#)
- [SAVAPI_REPORT_CONTENT_DATA](#) * [report_content_data](#)
- [SAVAPI_REPORT_SCAN_DETAILS_DATA](#) * [report_scan_details_data](#)
- [SAVAPI_OA_FILE_RESULT_DATA](#) * [oa_file_result_data](#)
- [SAVAPI_APC_SCAN_DATA](#) * [apc_scan_data](#)
- void * [private_data](#)

7.33.1 Detailed Description

Callbacks specific data.

7.33.2 Field Documentation

7.33.2.1 [apc_scan_data](#)

[SAVAPI_APC_SCAN_DATA](#)* [SAVAPI_callback_data::specific_data::apc_scan_data](#)

specific data for the APC scan callback. See [SAVAPI_apc_scan_data](#)

7.33.2.2 [archive_open_data](#)

[SAVAPI_ARCHIVE_OPEN_DATA](#)* [SAVAPI_callback_data::specific_data::archive_open_data](#)

specific data for archive open callback

See [SAVAPI_archive_open_data](#)

7.33.2.3 [error_data](#)

[SAVAPI_ERROR_DATA](#)* [SAVAPI_callback_data::specific_data::error_data](#)

specific data for error report callback

See [SAVAPI_error_data](#)

7.33.2.4 file_status_data

[SAVAPI_FILE_STATUS_DATA*](#) SAVAPI_callback_data::specific_data::file_status_data

specific data for file status callback

See [SAVAPI_file_status_data](#)

7.33.2.5 oa_file_result_data

[SAVAPI_OA_FILE_RESULT_DATA*](#) SAVAPI_callback_data::specific_data::oa_file_result_data

specific data for OnAccess file result callback. See [SAVAPI_OA_file_result_data](#)

7.33.2.6 pre_scan_data

[SAVAPI_PRESCAN_DATA*](#) SAVAPI_callback_data::specific_data::pre_scan_data

specific data for pre scan callback

See [SAVAPI_pre_scan_data](#)

7.33.2.7 private_data

void* SAVAPI_callback_data::specific_data::private_data

private data. Reserved for internal use

7.33.2.8 report_content_data

[SAVAPI_REPORT_CONTENT_DATA*](#) SAVAPI_callback_data::specific_data::report_content_data

specific data for report content callback. See [SAVAPI_report_content_data](#)

7.33.2.9 report_progress_data

[SAVAPI_REPORT_PROGRESS_DATA*](#) SAVAPI_callback_data::specific_data::report_progress_data

specific data for report progress callback

See [SAVAPI_report_progress_data](#)

7.33.2.10 report_scan_details_data

[SAVAPI_REPORT_SCAN_DETAILS_DATA*](#) SAVAPI_callback_data::specific_data::report_scan_details_data

specific data for report scan details callback. See [SAVAPI_report_scan_details_data](#)

The documentation for this union was generated from the following file:

- [savapi.h](#)

Chapter 8

File Documentation

8.1 asc_bin.h File Reference

Typedefs

- typedef int(* [bin2hex_t](#)) (const char *binblock, [SAVAPI_SIZE_T](#) binblock_size, char *hexblock, [SAVAPI_SIZE_T](#) *hexblock_size)
- typedef int(* [hex2bin_t](#)) (const char *hexblock, [SAVAPI_SIZE_T](#) hexblock_size, char *binblock, [SAVAPI_SIZE_T](#) *binblock_size)

Functions

- [SAVAPI_EXP](#) int [bin2asc](#) (const char *binblock, char **ascblock)
Convert the file name to ASCII hex representation.
- [SAVAPI_EXP](#) int [asc2bin](#) (const char *ascblock, [SAVAPI_SIZE_T](#) ascblock_size, char **binblock)
Convert ASCII hex representation to file name.
- [SAVAPI_EXP](#) int [bin2hex](#) (const char *binblock, [SAVAPI_SIZE_T](#) binblock_size, char *hexblock, [SAVAPI_SIZE_T](#) *hexblock_size)
Transform the string given in binblock in the equivalent encoded string in hexblock.
- [SAVAPI_EXP](#) int [hex2bin](#) (const char *hexblock, [SAVAPI_SIZE_T](#) hexblock_size, char *binblock, [SAVAPI_SIZE_T](#) *binblock_size)
Converts a series of hex-encoded characters to normal binary encoding.

8.1.1 Function Documentation

8.1.1.1 asc2bin()

```
SAVAPI\_EXP int asc2bin (  
    const char * ascblock,  
    SAVAPI\_SIZE\_T ascblock_size,  
    char ** binblock )
```

Convert ASCII hex representation to file name.

Parameters

<i>ascblock</i>	[IN]: The ASCII hex block
<i>ascblock_size</i>	[IN]: The ASCII hex block size
<i>binblock</i>	[OUT]: The converted file name

Returns

SAVAPI_S_OK for success or an error code

Note

The binblock parameter will be internally allocated. The caller is responsible to release the memory (ie: calling [SAVAPI_free\(\)](#) on binblock).

This function is obsolete. Instead, please use the new, [hex2bin\(\)](#) function.

8.1.1.2 bin2asc()

```
SAVAPI_EXP int bin2asc (  
    const char * binblock,  
    char ** ascblock )
```

Convert the file name to ASCII hex representation.

Parameters

<i>binblock</i>	[IN]: The file name to convert
<i>ascblock</i>	[OUT]: The ASCII hex file name representation

Returns

SAVAPI_S_OK for success or an error code

Note

The ascblock parameter will be internally allocated. The caller is responsible to release the memory (ie: calling [SAVAPI_free\(\)](#) on ascblock).

This function is obsolete. Instead, please use the new, [bin2hex\(\)](#) function.

8.1.1.3 bin2hex()

```
SAVAPI_EXP int bin2hex (  
    const char * binblock,  
    SAVAPI_SIZE_T binblock_size,  
    char * hexblock,  
    SAVAPI_SIZE_T * hexblock_size )
```

Transform the string given in binblock in the equivalent encoded string in hexblock.

Parameters

<i>binblock</i>	[IN]: Buffer containing the string to be transformed.
<i>binblock_size</i>	[IN]: Number of characters to encode from the binblock
<i>hexblock</i>	[OUT]: Will contain the transformed buffer. Must be allocated by caller
<i>hexblock_size</i>	[IN/OUT]: On input it contains the available size of the hexblock, on output will contain needed size for the resulted hexblock It must be at least two times the binblock_size (each character will be displayed as 2 hexadecimal digits)

Returns

- SAVAPI_S_OK for success,
- SAVAPI_E_INVALID_PARAMETER if one of the input parameters is NULL, or the binblock length exceeds the maximum allowed buffer size
- SAVAPI_E_BUFFER_TOO_SMALL if the given size is too small,
- SAVAPI_E_CONVERSION_FAILED if the conversion could not be performed

Note

If the hexblock buffer is not big enough it will return an error and will put the needed size in the hexblock_size parameter on output.

Unlike the bin2asc function, this functions does not internally allocate anything. It will use buffers allocated by the caller according to the given size.

8.1.1.4 hex2bin()

```
SAVAPI_EXP int hex2bin (
    const char * hexblock,
    SAVAPI_SIZE_T hexblock_size,
    char * binblock,
    SAVAPI_SIZE_T * binblock_size )
```

Converts a series of hex-encoded characters to normal binary encoding.

Parameters

<i>hexblock</i>	[IN]: Buffer containing an already encoded sequence of characters
<i>hexblock_size</i>	[IN]: Number of characters to decode from the hexblock
<i>binblock</i>	[OUT]: Contains the original string. Must be allocated by caller
<i>binblock_size</i>	[IN/OUT]: On input contains the available size of the binblock, on output will contain needed size for the resulted binblock It must be at least half the binblock_size (each character is displayed as 2 hexadecimal digits)

Returns

- SAVAPI_S_OK for success,

- SAVAPI_E_INVALID_PARAMETER if one of the input parameters is NULL, or the binblock length exceeds the maximum allowed buffer size
- SAVAPI_E_BUFFER_TOO_SMALL if the given size is too small,
- SAVAPI_E_CONVERSION_FAILED if the conversion could not be performed

Note

Unlike the asc2bin function, this functions does not internally allocate anything. It will use buffers allocated by the caller according to the given size.

8.2 asc_bin.h

[Go to the documentation of this file.](#)

```

00001 #ifndef ASC_BIN_H_
00002 #define ASC_BIN_H_
00003
00004 #include "savapi.h"
00005
00006 #ifdef __cplusplus
00007 extern "C" {
00008 #endif
00009
00017 typedef int (CC *bin2hex_t)(const char *binblock, SAVAPI_SIZE_T binblock_size, char *hexblock,
SAVAPI_SIZE_T *hexblock_size);
00018 typedef int (CC *hex2bin_t)(const char *hexblock, SAVAPI_SIZE_T hexblock_size, char *binblock,
SAVAPI_SIZE_T *binblock_size);
00019
00032 SAVAPI_EXP int CC bin2asc(const char *binblock, char **ascblock);
00033
00046 SAVAPI_EXP int CC asc2bin(const char *ascblock, SAVAPI_SIZE_T ascblock_size, char **binblock);
00047
00068 SAVAPI_EXP int CC bin2hex(const char *binblock, SAVAPI_SIZE_T binblock_size, char *hexblock,
SAVAPI_SIZE_T *hexblock_size);
00069
00086 SAVAPI_EXP int CC hex2bin(const char *hexblock, SAVAPI_SIZE_T hexblock_size, char *binblock,
SAVAPI_SIZE_T *binblock_size);
00087
00088 #ifdef __cplusplus
00089 }
00090 #endif
00091 #endif /*ASC_BIN_H_*/

```

8.3 fops.h File Reference

This file defines the interface to allow the engine to scan any type of object.

Data Structures

- [struct _AVE_STRUCT_FILE_OPERATIONS](#)

The File OperationS (FOPS) structure. Here are defined all the function prototypes which must be implemented in order to allow the scanning engine to process a special type of object.

Macros

- #define [FOPS_INVALID_HANDLE](#) NULL
- #define [FOPS_ERROR](#) 1

File Open modes

- #define [OPEN_RO](#) 0
- #define [OPEN_RW](#) 1
- #define [OPEN_CR](#) 2

SEEK defines

- #define [SEEK_SET](#) 0
- #define [SEEK_CUR](#) 1
- #define [SEEK_END](#) 2

Typedefs

- typedef void * [FOPS_HANDLE](#)
- typedef long [_fpos_t](#)
- typedef int [_fattr_t](#)
- typedef struct [_AVE_STRUCT_FILE_OPERATIONS](#) [AVE_FOPS](#)

Functions

- void [check_free_mem](#) (void)
- int [e_tempname](#) (void *dir)

Variables

- int [f_check_mem](#)
- [AVE_FOPS_user_fops](#)

8.3.1 Detailed Description

This file defines the interface to allow the engine to scan any type of object.

Warning

Please take care to respect the following constraints when implementing the following functions:

open

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_open](#)

The implementation needs to set the passed handle to invalid [FOPS_INVALID_HANDLE](#) in case of an error.

read

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_read](#)

set *nread on error as well reading 0 bytes is not an error case.

write

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_write](#)

Set *nwritten in all error cases

tell

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_tell](#)

*curpos must be -1 in all error cases

getfattr

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_getfattr](#)

*attr must be set NULL in all error cases

getfsize

See also

[_AVE_STRUCT_FILE_OPERATIONS::fops_getfsize](#)

*fsize must be 0 in all error cases

8.3.2 Macro Definition Documentation

8.3.2.1 FOPS_ERROR

```
#define FOPS_ERROR 1
```

Return this in the functions which return an error

8.3.2.2 FOPS_INVALID_HANDLE

```
#define FOPS_INVALID_HANDLE NULL
```

Invalid handle. Set this when the file can not be created/accessed

8.3.2.3 OPEN_CR

```
#define OPEN_CR 2
```

Create file

8.3.2.4 OPEN_RO

```
#define OPEN_RO 0
```

Open Read Only

8.3.2.5 OPEN_RW

```
#define OPEN_RW 1
```

Open Read/Write

8.3.2.6 SEEK_CUR

```
#define SEEK_CUR 1
```

Start from the current position

8.3.2.7 SEEK_END

```
#define SEEK_END 2
```

Start from the end of the file

8.3.2.8 SEEK_SET

```
#define SEEK_SET 0
```

Start from the beginning

8.3.3 Typedef Documentation

8.3.3.1 __fattr_t

```
typedef int __fattr_t
```

Special attribute type for our FOPS

8.3.3.2 `_fpos_t`

```
typedef long _fpos_t
```

Special data type for our FOPS

8.3.3.3 `AVE_FOPS`

```
typedef struct _AVE_STRUCT_FILE_OPERATIONS AVE_FOPS
```

8.3.3.4 `FOPS_HANDLE`

```
typedef void* FOPS_HANDLE
```

A generic definition of the `FOPS_HANDLE`. You may have anything you want here.

8.3.4 Function Documentation

8.3.4.1 `check_free_mem()`

```
void check_free_mem (
    void )
```

8.3.4.2 `e_tempname()`

```
int e_tempname (
    void * dir )
```

8.3.5 Variable Documentation

8.3.5.1 `_user_fops`

```
AVE_FOPS _user_fops [extern]
```

8.3.5.2 f_check_mem

```
int f_check_mem [extern]
```

8.4 fops.h

[Go to the documentation of this file.](#)

```
00001
00027 /**
00028  * Usage of the 'fops_context' and 'file_context' parameters
00029  *
00030  * In general, the application can either use the standard FOPS provided by the
00031  * SAVAPI Library _OR_ it can implement its own FOPS functions in order to prevent
00032  * the SAVAPI Library from using the standard memory and file I/O functions, and
00033  * to use the application-defined functions instead.
00034  *
00035  * If an application decides to provide its own FOPS functions, it has to
00036  * implement _ALL_ memory and file I/O functions that are part of the
00037  * AVE_FOPS structure (defined in this header file).
00038  *
00039  * The 'fops_context' parameter is an opaque data set by the applications through
00040  * @ref SAVAPI_set_fops.
00041  * It is reserved for the application in order to pass user-defined data associated
00042  * with the FOPS instance to each FOPS function. SAVAPI Library passes it to all
00043  * FOPS functions without modifying its value.
00044  *
00045  * The 'file_context' parameter is internally used by the SAVAPI Library and it
00046  * should not be touched by the applications!
00047  */
00048
00049
00050 /*
00051  Doxygen has a problem parsing the function definitions below
00052  when the calling convention is used. This is why I have defined CC as _cdecl.
00053  If you want to generate the docs, just remove the string CC.
00054  */
00055
00056 #ifndef __FOPS_H
00057 #define __FOPS_H
00058
00059
00060 #include <stdio.h>
00061 #include <sys/stat.h>
00062
00063 /* all functions are extern "C" */
00064 #ifdef __cplusplus
00065 extern "C"
00066 {
00067 #endif
00068
00069 #ifndef AVCORE
00070 #include "fopstypes.h"
00071 #else
00072 #include "system.h"
00073 #endif
00074
00084 #define OPEN_RO 0
00085
00087 #define OPEN_RW 1
00088
00090 #define OPEN_CR 2
00091
00097 #define FOPS_INVALID_HANDLE NULL
00098
00100 #define FOPS_ERROR 1
00101
00105 typedef void *FOPS_HANDLE;
00106
00110 #ifndef SEEK_SET
00112 # define SEEK_SET 0
00114 # define SEEK_CUR 1
00116 # define SEEK_END 2
00117 #endif
00123 typedef long __fpos_t;
00124
00126 typedef int __fattr_t; /* S_IREAD, S_IWRITE */
00127
00132 #ifndef CC
00133 # define CC _cdecl
```

```

00134 #endif
00135
00141 typedef struct _AVE_STRUCT_FILE_OPERATIONS
00142 {
00155 int (CC *fops_open)(FOPS_HANDLE *fh, void *filename, int mode, void *file_context, void
    *fops_context);
00156
00163 int (CC *fops_close)(FOPS_HANDLE *fh, void *fops_context);
00164
00177 int (CC *fops_read)(FOPS_HANDLE fh, void *buffer, UINT64 count, UINT64 *nread, void *fops_context);
00178
00190 int (CC *fops_write)(FOPS_HANDLE fh, void *buffer, UINT64 count, UINT64 *nwritten, void
    *fops_context);
00191
00199 int (CC *fops_tell)(FOPS_HANDLE fh, INT64 *curpos, void *fops_context);
00200
00213 int (CC *fops_seek)(FOPS_HANDLE fh, INT64 offset, int wherefrom, void *fops_context);
00214
00223 int (CC *fops_getfattr)(void *filename, _fattr_t *attr, void *file_context, void *fops_context);
00224
00237 int (CC *fops_setfattr)(void *filename, _fattr_t attr, void *file_context, void *fops_context);
00238
00247 int (CC *fops_getfsize)(FOPS_HANDLE fh, INT64 *fsize, void *fops_context);
00248
00256 int (CC *fops_unlink)(void *filename, void *file_context, void *fops_context);
00257
00266 int (CC *fops_rename)(void *oldname, void *newname, void *file_context, void *fops_context);
00267
00279 int (CC *fops_access)(void *filename, int amode, void *file_context, void *fops_context);
00280
00288 void *(CC *fops_malloc)(UINT64 size, void *fops_context);
00289
00296 void (CC *fops_free)(void *ptr, void *fops_context);
00297
00309 char *(CC *fops_gets)(FOPS_HANDLE fh, char *str, int num, void *fops_context);
00310
00322 int (CC *fops_puts)(FOPS_HANDLE fh, char *str, void *fops_context);
00323
00335 int (CC *fops_getc)(FOPS_HANDLE fh, void *fops_context);
00336
00345 int (CC *fops_putc)(FOPS_HANDLE fh, int character, void *fops_context);
00346
00366 int (CC *fops_ungetc)(FOPS_HANDLE fh, int character, void *fops_context);
00367
00376 int (CC *fops_flush)(FOPS_HANDLE fh, void *fops_context);
00377
00383 int (CC *fops_get_last_error)(void *fops_context);
00384
00385 } AVE_FOPS;
00386
00387 extern int f_check_mem;
00388 void check_free_mem(void);
00389 int e_tempname(void *dir);
00390
00391 extern AVE_FOPS _user_fops;
00392
00393 #ifdef __cplusplus
00394 }
00395 #endif
00396
00397 #endif /* __FILEIO_H */
00398
00399

```

8.5 fopstypes.h File Reference

Macros

- #define `_cdecl`

Typedefs

- typedef u_int64_t `UINT64`
- typedef int64_t `INT64`
- typedef u_int32_t `UINT32`

- typedef int32_t [INT32](#)
- typedef u_int16_t [UINT16](#)
- typedef int16_t [INT16](#)
- typedef u_int8_t [UINT8](#)
- typedef int8_t [INT8](#)

8.5.1 Macro Definition Documentation

8.5.1.1 _cdecl

```
#define _cdecl
```

8.5.2 Typedef Documentation

8.5.2.1 INT16

```
typedef int16_t INT16
```

8.5.2.2 INT32

```
typedef int32_t INT32
```

8.5.2.3 INT64

```
typedef int64_t INT64
```

8.5.2.4 INT8

```
typedef int8_t INT8
```

8.5.2.5 UINT16

```
typedef u_int16_t  UINT16
```

8.5.2.6 UINT32

```
typedef u_int32_t  UINT32
```

8.5.2.7 UINT64

```
typedef u_int64_t  UINT64
```

8.5.2.8 UINT8

```
typedef u_int8_t   UINT8
```

8.6 fopstypes.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __FOPSTYPES_H
00002 #define __FOPSTYPES_H
00003
00004 #ifndef DISABLE_AVETYPES
00005
00006 #if defined(__WIN32__) || defined(_WIN32) || defined(WIN32)
00007     #include <BaseTsd.h>
00008 #else
00009     #include <sys/types.h>
00010     #ifndef _cdecl
00011         #define _cdecl
00012     #endif
00013     #if defined(__sun__)
00014         typedef uint64_t      UINT64;
00015         typedef int64_t       INT64;
00016         typedef uint32_t      UINT32;
00017         typedef int32_t       INT32;
00018         typedef uint16_t      UINT16;
00019         typedef int16_t       INT16;
00020         typedef uint8_t       UINT8;
00021         typedef int8_t        INT8;
00022     #else
00023         typedef u_int64_t     UINT64;
00024         typedef int64_t       INT64;
00025         typedef u_int32_t     UINT32;
00026         typedef int32_t       INT32;
00027         typedef u_int16_t     UINT16;
00028         typedef int16_t       INT16;
00029         typedef u_int8_t      UINT8;
00030         typedef int8_t        INT8;
00031     #endif
00032 #endif
00033
00034 #endif // __AVETYPES_H
00037
```


8.7 savapi.h File Reference

Data Structures

- struct [SAVAPI_global_init](#)
The structure used at SAVAPI initialization.
- struct [SAVAPI_APC_global_init](#)
The structure used for initializing APC.
- struct [SAVAPI_instance_init](#)
The structure used at SAVAPI instance creation.
- struct [SAVAPI_file_info](#)
Contains data about the scanned file.
- struct [SAVAPI_malware_info](#)
Contains data about the found malware in an infected/suspicious file.
- struct [SAVAPI_pre_scan_data](#)
Contains the data sent to a prescan callback.
- struct [SAVAPI_archive_open_data](#)
Contains the data sent to a archive_open callback.
- struct [SAVAPI_key_value](#)
Generic container.
- struct [SAVAPI_file_status_data](#)
Contains the data sent to a report file status callback.
- struct [SAVAPI_OA_file_result_data](#)
Contains the data sent to an OnAccess file status callback.
- struct [SAVAPI_error_data](#)
The structure associated with report error callback.
- struct [SAVAPI_APC_REPORT_DATA](#)
Information to report when calling a [APC_set_report_info_t](#) function.
- struct [SAVAPI_apc_scan_data](#)
The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.
- struct [SAVAPI_report_progress_data](#)
The structure associated with report progress callback.
- struct [SAVAPI_iframe_url_data](#)
Structure associated with the iframe report.
- struct [SAVAPI_report_content_data](#)
The structure associated with report content callback.
- union [SAVAPI_report_content_data::_content_data](#)
- struct [SAVAPI_alert_url_data](#)
Structure associated with the ALERTURL report.
- struct [SAVAPI_repairable_data](#)
Structure associated with the REPAIRABLE report.
- struct [SAVAPI_report_scan_details_data](#)
The structure associated with report scan details callback.
- union [SAVAPI_report_scan_details_data::_scan_details_data](#)
- struct [SAVAPI_callback_data](#)
Structure passed by SAVAPI to a user defined callback, containing all the necessary data.
- union [SAVAPI_callback_data::_specific_data](#)
Callbacks specific data.
- struct [SAVAPI_simple_scan_file_data](#)
The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).

- struct [SAVAPI_simple_scan_statistics](#)
The structure contains statistics for the simple scan.
- struct [SAVAPI_simple_scan_output](#)
The structure containing the output for [SAVAPI_simple_scan](#).
- struct [SAVAPI_signal_data](#)
The structure to be passed when sending a signal.
- struct [SAVAPI_command_data](#)
The structure to be passed when sending a command.
- struct [SAVAPI_version](#)
The structure used to retrieve SAVAPI version.
- struct [SAVAPI_OA_global_init](#)
The structure used for initializing OnAccess.

Macros

- #define [SAVAPI_API_MAJOR_VERSION](#) 5
Major API version.
- #define [SAVAPI_API_MINOR_VERSION](#) 5
Minor API version.
- #define [SAVAPI_SYMBOL](#)(s) [STR](#)(s)
Macro to be used when loading a SAVAPI symbol while using dynamic linking.
- #define [STR](#)(s) #s
- #define [SAVAPI_ECAT_ERROR_IO](#) 0
- #define [SAVAPI_ECAT_ERROR_SCAN](#) 1
- #define [SAVAPI_ECAT_ERROR_UNPACK](#) 2
- #define [SAVAPI_ECAT_ERROR_GENERIC](#) 3
- #define [SAVAPI_ECAT_APC_REPORT_TTL](#) 4
- #define [SAVAPI_ELEVEL_ERROR](#) 0
- #define [SAVAPI_ELEVEL_WARNING](#) 1
- #define [SAVAPI_ELEVEL_INFO](#) 2
- #define [SAVAPI_FLAG_USE_TCP](#) 1
- #define [SAVAPI_FLAG_USE_LOCAL_SOCKET](#) 2
- #define [SAVAPI_W_DAMAGED](#) 1
- #define [SAVAPI_W_OLE_DAMAGED](#) 2
- #define [SAVAPI_W_SUSPICIOUS](#) 4
- #define [SAVAPI_W_PROGRESS_ABORT](#) 8
- #define [SAVAPI_W_HEADER_MALFORMED](#) 16
- #define [SAVAPI_W_POTENTIAL_ARCH_BOMB](#) 32
- #define [SAVAPI_W_RATIO_EXCEEDED](#) 64
- #define [SAVAPI_W_MAX_EXTRACTED](#) 128
- #define [SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE](#) 1
- #define [SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL](#) 2
- #define [SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS](#) 4
- #define [SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS](#) 8
- #define [SAVAPI_I_OLEFILE](#) 1
- #define [SAVAPI_I_TEMPLATE](#) 2
- #define [SAVAPI_I_MACROS_PRESENT](#) 4
- #define [SAVAPI_I_ALL_MACROS_DELETED](#) 8
- #define [SAVAPI_I_OLE_ENCRYPTED](#) 16
- #define [SAVAPI_I_ACTIVE_CONTENT_PRESENT](#) 32
- #define [SAVAPI_I_MAILBOX](#) 64
- #define [SAVAPI_I_MACRO_AUTOSTART](#) 128

- #define [SAVAPI_I_APC_SCAN_DURATION](#) 256
- #define [SAVAPI_I_APC_RESULT_EXPIRY](#) 512
- #define [SAVAPI_CALLBACK_REPORT_FILE_STATUS](#) 0

Triggered after a file is scanned. The callback data contains the status of the last scanned file.
- #define [SAVAPI_CALLBACK_REPORT_ERROR](#) 3

Triggered to report an error or a warning.
- #define [SAVAPI_CALLBACK_PRE_SCAN](#) 4

Triggered before the scanning begins. Can be used to create filters. For example, if we want to scan only .exe files, we install a PRE_SCAN callback. Before each file is scanned, the PRE_SCAN callback will be called. Inside our implementation of the callback, we implement the filter. If the returned code is success, the file will be scanned, otherwise it will be skipped.
- #define [SAVAPI_CALLBACK_ARCHIVE_OPEN](#) 5

Triggered before opening an archive. If the returned code is success, the archive will be opened, otherwise it will be skipped from opening.
- #define [SAVAPI_CALLBACK_PROGRESS_REPORT](#) 6

Triggered when messages related to scan progress are available.
- #define [SAVAPI_CALLBACK_CONTENT_REPORT](#) 7

Triggered when messages related to scan (progress, warnings or infos that are not error_callback related) are available. IFRAME detection ([SAVAPI_OPTION_IFRAMES_URL](#)) will be reported through this callback.
- #define [SAVAPI_CALLBACK_SCAN_DETAILS_REPORT](#) 8

Triggered when messages related to scan process details are available. The virus description url ([SAVAPI_OPTION_NOTIFY_ALERTURL](#)) will be reported through this callback.
- #define [SAVAPI_CALLBACK_OA_FILE_RESULT](#) 9

Triggered after a file was scanned with OnAccess, in order to decide what action to take The action is taken depending on the return code of the callback (see [SAVAPI_OA_SCAN_RESULT](#))
- #define [SAVAPI_CALLBACK_APC_SCAN](#) 10
- #define [SAVAPI_REPORT_ALERTURL](#) 1
- #define [SAVAPI_REPORT_REPAIRABLE](#) 2
- #define [SAVAPI_REPORT_CONTENT_IFRAME](#) 0
- #define [SAVAPI_SIGNAL_SCAN_ABORT](#) 1

Will cause the SAVAPI instance to abort scanning process as soon as possible.
- #define [SAVAPI_SCAN_STATUS_CLEAN](#) 0
- #define [SAVAPI_SCAN_STATUS_INFECTED](#) 1
- #define [SAVAPI_SCAN_STATUS_SUSPICIOUS](#) 2
- #define [SAVAPI_SCAN_STATUS_ERROR](#) 3
- #define [SAVAPI_SCAN_STATUS_FINISHED](#) 4
- #define [SAVAPI_FTYPE_REGULAR](#) 4
- #define [SAVAPI_FTYPE_ARCHIVE](#) 1
- #define [SAVAPI_FTYPE_IN_ARCHIVE](#) 2
- #define [SAVAPI_FLAG_LAST_FILENAME_DEFAULT](#) 1 << 0

The last filename is not the one reported by the engine, but a default one set by the lib.

Typedefs

- typedef enum [SAVAPI_option](#) [SAVAPI_OPTION](#)
- typedef enum [SAVAPI_global_option](#) [SAVAPI_GLOBAL_OPTION](#)
- typedef struct [SAVAPI_global_init](#) [SAVAPI_GLOBAL_INIT](#)

The structure used at SAVAPI initialization.
- typedef enum [SAVAPI_apc_scan_mode](#) [SAVAPI_APC_SCAN_MODE](#)

Defines the APC scan mode.
- typedef struct [SAVAPI_apc_global_init](#) [SAVAPI_APC_GLOBAL_INIT](#)

The structure used for initializing APC.
- typedef struct [SAVAPI_instance_init](#) [SAVAPI_INSTANCE_INIT](#)

- The structure used at SAVAPI instance creation.*

 - typedef struct [SAVAPI_file_info SAVAPI_FILE_INFO](#)

Contains data about the scanned file.
 - typedef struct [SAVAPI_malware_info SAVAPI_MALWARE_INFO](#)

Contains data about the found malware in an infected/suspicious file.
 - typedef struct [SAVAPI_pre_scan_data SAVAPI_PRESCAN_DATA](#)

Contains the data sent to a prescan callback.
 - typedef struct [SAVAPI_archive_open_data SAVAPI_ARCHIVE_OPEN_DATA](#)

Contains the data sent to a archive_open callback.
 - typedef struct [SAVAPI_key_value SAVAPI_KEY_VALUE](#)

Generic container.
 - typedef struct [SAVAPI_file_status_data SAVAPI_FILE_STATUS_DATA](#)

Contains the data sent to a report file status callback.
 - typedef struct [SAVAPI_OA_file_result_data SAVAPI_OA_FILE_RESULT_DATA](#)

Contains the data sent to an OnAccess file status callback.
 - typedef struct [SAVAPI_error_data SAVAPI_ERROR_DATA](#)

The structure associated with report error callback.
 - typedef void * [SAVAPI_FD](#)

SAVAPI instance handle.
 - typedef [SAVAPI_STATUS](#)(* [APC_set_report_info_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_APC_REPORT_DATA](#) *data)

Type of function that is called by the user to report findings regarding a scanned file.
 - typedef struct [SAVAPI_apc_scan_data SAVAPI_APC_SCAN_DATA](#)

The structure associated with the [SAVAPI_CALLBACK_APC_SCAN](#) callback.
 - typedef struct [SAVAPI_report_progress_data SAVAPI_REPORT_PROGRESS_DATA](#)

The structure associated with report progress callback.
 - typedef struct [SAVAPI_iframe_url_data SAVAPI_IFRAME_URL_DATA](#)

Structure associated with the iframe report.
 - typedef struct [SAVAPI_report_content_data SAVAPI_REPORT_CONTENT_DATA](#)

The structure associated with report content callback.
 - typedef struct [SAVAPI_alert_url_data SAVAPI_ALERT_URL_DATA](#)

Structure associated with the ALERTURL report.
 - typedef struct [SAVAPI_repairable_data SAVAPI_REPAIRABLE_DATA](#)

Structure associated with the REPAIRABLE report.
 - typedef struct [SAVAPI_report_scan_details_data SAVAPI_REPORT_SCAN_DETAILS_DATA](#)

The structure associated with report scan details callback.
 - typedef struct [SAVAPI_callback_data SAVAPI_CALLBACK_DATA](#)

Structure passed by SAVAPI to a user defined callback, containing all the necessary data.
 - typedef struct [SAVAPI_simple_scan_file_data SAVAPI_SIMPLE_SCAN_FILE_DATA](#)

The structure contains information about each infected, suspicious, or erroneous file scanned by [SAVAPI_simple_scan](#).
 - typedef struct [SAVAPI_simple_scan_statistics SAVAPI_SIMPLE_SCAN_STATISTICS](#)

The structure contains statistics for the simple scan.
 - typedef struct [SAVAPI_simple_scan_output SAVAPI_SIMPLE_SCAN_OUTPUT](#)

The structure containing the output for [SAVAPI_simple_scan](#).
 - typedef struct [SAVAPI_signal_data SAVAPI_SIGNAL_DATA](#)

The structure to be passed when sending a signal.
 - typedef struct [SAVAPI_command_data SAVAPI_COMMAND_DATA](#)

The structure to be passed when sending a command.
 - typedef struct [SAVAPI_version SAVAPI_VERSION](#)

The structure used to retrieve SAVAPI version.
 - typedef enum [_SAVAPI_log_level SAVAPI_LOG_LEVEL](#)

The enumeration used to specify the SAVAPI's logging levels.

- typedef enum [SAVAPI_engine_module_type](#) [SAVAPI_ENGINE_MODULE_TYPE](#)

Defines the type of an engine module.

- typedef struct [SAVAPI_OA_global_init](#) [SAVAPI_OA_GLOBAL_INIT](#)

The structure used for initializing OnAccess.

- typedef int(* [SAVAPI_CALLBACK](#)) ([SAVAPI_CALLBACK_DATA](#) *data)

SAVAPI callback function pointer definition.

- typedef void(* [SAVAPI_LOG_CALLBACK](#)) ([SAVAPI_LOG_LEVEL](#) log_level, const [SAVAPI_TCHAR](#) *message, void *user_data)

SAVAPI callback for logging.

- typedef int(* [SAVAPI_ENGINE_MODULE_CALLBACK](#)) (const [SAVAPI_TCHAR](#) *name, [SAVAPI_ENGINE_MODULE_TYPE](#) type, void *user_data)

Callback function used to return one engine single module.

- typedef int(* [SAVAPI_OA_INSTANCE_CALLBACK](#)) ([SAVAPI_FD](#) savapi_fd)

Callback function used to configure the instances created by OnAccess.

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_set_log_callback_t](#)) ([SAVAPI_LOG_CALLBACK](#) log_fct, [SAVAPI_LOG_LEVEL](#) min_level, void *user_data)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_initialize_t](#)) ([SAVAPI_GLOBAL_INIT](#) *savapi_init)

- typedef void(* [SAVAPI_set_quickload_init_t](#)) ()

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_uninitialize_t](#)) ()

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_get_version_t](#)) ([SAVAPI_VERSION](#) *version)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_engine_versions_get_t](#)) ([SAVAPI_VERSION](#) *ave_version, [SAVAPI_VERSION](#) *avpack_version, [SAVAPI_VERSION](#) *vdf_version)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_APC_get_version_t](#)) ([SAVAPI_VERSION](#) *apc_version)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_create_instance_t](#)) ([SAVAPI_INSTANCE_INIT](#) *init, [SAVAPI_FD](#) *savapi_fd)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_release_instance_t](#)) ([SAVAPI_FD](#) *savapi_fd)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_set_user_data_t](#)) ([SAVAPI_FD](#) savapi_fd, void *user_data)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_get_user_data_t](#)) ([SAVAPI_FD](#) savapi_fd, void **user_data)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_is_running_ex_t](#)) (const [SAVAPI_TCHAR](#) *hostname, unsigned int port)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_register_callback_t](#)) ([SAVAPI_FD](#) savapi_fd, unsigned int callback_id, [SAVAPI_CALLBACK](#) callback)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_unregister_callback_t](#)) ([SAVAPI_FD](#) savapi_fd, unsigned int callback_id, [SAVAPI_CALLBACK](#) callback)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_scan_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_TCHAR](#) *file_name)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_simple_scan_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_TCHAR](#) *file_name, [SAVAPI_SIMPLE_SCAN_OUTPUT](#) *output)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_set_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_OPTION](#) option_id, [SAVAPI_TCHAR](#) *buffer)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_get_t](#)) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_OPTION](#) option_id, [SAVAPI_TCHAR](#) *buffer, [SAVAPI_SIZE_T](#) *buffer_size)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_send_signal_t](#)) ([SAVAPI_FD](#) savapi_fd, unsigned int signal_id, [SAVAPI_SIGNAL_DATA](#) *data)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_set_fops_t](#)) ([SAVAPI_FD](#) savapi_fd, void *fops_pointer, void *fops_↔ context)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_get_fops_t](#)) ([SAVAPI_FD](#) savapi_fd, void **fops_pointer, void **fops_↔ context)

- typedef void(* [SAVAPI_free_t](#)) (void **ptr)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_reload_engine_ex_t](#)) (const [SAVAPI_GLOBAL_INIT](#) *global_init)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_extract_malware_names_t](#)) (const [SAVAPI_TCHAR](#) *dir_path)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_engine_modules_get_t](#)) (const [SAVAPI_GLOBAL_INIT](#) *init, [SAVAPI_ENGINE_MODULE_CALLBACK](#) module_func, void *user_data)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_global_set_t](#)) ([SAVAPI_GLOBAL_OPTION](#) option_id, [SAVAPI_TCHAR](#) *buffer)

- typedef [SAVAPI_STATUS](#)(* [SAVAPI_APC_initialize_t](#)) ([SAVAPI_APC_GLOBAL_INIT](#) *savapi_apc_init)

- typedef SAVAPI_STATUS(* SAVAPI_APC_uninitialize_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_initialize_t) (SAVAPI_OA_GLOBAL_INIT *savapi_oa_init)
- typedef SAVAPI_STATUS(* SAVAPI_OA_uninitialize_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_create_instances_t) (SAVAPI_OA_INSTANCE_CALLBACK init_↵
func, SAVAPI_OA_INSTANCE_CALLBACK uninit_func)
- typedef SAVAPI_STATUS(* SAVAPI_OA_start_scan_t) ()
- typedef SAVAPI_STATUS(* SAVAPI_OA_stop_scan_t) ()

Enumerations

- enum SAVAPI_option {
 SAVAPI_OPTION_CWD = 1, SAVAPI_OPTION_CONF,
 SAVAPI_OPTION_ARCHIVE_SCAN, SAVAPI_OPTION_ARCHIVE_MAX_SIZE,
 SAVAPI_OPTION_ARCHIVE_MAX_REC, SAVAPI_OPTION_ARCHIVE_MAX_RATIO,
 SAVAPI_OPTION_ARCHIVE_MAX_COUNT, SAVAPI_OPTION_MAILBOX_SCAN,
 SAVAPI_OPTION_HEUR_MACRO, SAVAPI_OPTION_HEUR_LEVEL,
 SAVAPI_OPTION_SCAN_TEMP, SAVAPI_OPTION_SCAN_TIMEOUT,
 SAVAPI_OPTION_REPAIR, SAVAPI_OPTION_NOTIFY_REPAIR,
 SAVAPI_OPTION_NOTIFY_OFFICE, SAVAPI_OPTION_NOTIFY_OFFICE_MACRO,
 SAVAPI_OPTION_NOTIFY_ALERTURL, SAVAPI_OPTION_DETECT_ADSPY,
 SAVAPI_OPTION_DETECT_APPL, SAVAPI_OPTION_DETECT_BDC,
 SAVAPI_OPTION_DETECT_DIAL, SAVAPI_OPTION_DETECT_GAME,
 SAVAPI_OPTION_DETECT_HIDDENEXT, SAVAPI_OPTION_DETECT_JOKE,
 SAVAPI_OPTION_DETECT_PCK, SAVAPI_OPTION_DETECT_PHISH,
 SAVAPI_OPTION_DETECT_SPR, SAVAPI_OPTION_IFRAMES_URL,
 SAVAPI_OPTION_REPORT_ENCRYPTED_MIME, SAVAPI_OPTION_SCAN_MODE,
 SAVAPI_OPTION_MIME_SCAN, SAVAPI_OPTION_PGP_SCAN,
 SAVAPI_OPTION_SCAN_PROGRESS, SAVAPI_OPTION_DETECT_ADWARE,
 SAVAPI_OPTION_DETECT_PFS, SAVAPI_OPTION_RESERVED1,
 SAVAPI_OPTION_RESERVED2, SAVAPI_OPTION_APC_CONNECTION_TIMEOUT,
 SAVAPI_OPTION_APC_SCAN_TIMEOUT, SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL,
 SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL, SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART
 ,
 SAVAPI_OPTION_DETECT_PUA, SAVAPI_OPTION_APC_REPORT_SCAN_TTL,
 SAVAPI_OPTION_FPC, SAVAPI_OPTION_FPC_TIMEOUT,
 SAVAPI_OPTION_APC_PE_MODE, SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY,
 SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED, SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY
 ,
 SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL, SAVAPI_OPTION_APC_ELF_MODE,
 SAVAPI_OPTION_APC_MACH_O_MODE, SAVAPI_OPTION_PRODUCT = 1000,
 SAVAPI_OPTION_DETECT_ALLTYPES, SAVAPI_OPTION_SCAN_TIMEOUTS,
 SAVAPI_OPTION_SAVAPI = 2000, SAVAPI_OPTION_AVE_VERSION,
 SAVAPI_OPTION_VDF_VERSION, SAVAPI_OPTION_PID,
 SAVAPI_OPTION_EXPIRE, SAVAPI_OPTION_VDFSIGCOUNT,
 SAVAPI_OPTION_SELECTABLE_DETECT, SAVAPI_OPTION_DESCR_ADSPY,
 SAVAPI_OPTION_DESCR_APPL, SAVAPI_OPTION_DESCR_BDC,
 SAVAPI_OPTION_DESCR_DIAL, SAVAPI_OPTION_DESCR_GAME,
 SAVAPI_OPTION_DESCR_HIDDENEXT, SAVAPI_OPTION_DESCR_JOKE,
 SAVAPI_OPTION_DESCR_PCK, SAVAPI_OPTION_DESCR_PHISH,
 SAVAPI_OPTION_DESCR_SPR, SAVAPI_OPTION_VDF_DATE,
 SAVAPI_OPTION_MALWARE_NAMES_FILE, SAVAPI_OPTION_DESCR_ADWARE,
 SAVAPI_OPTION_DESCR_PFS, SAVAPI_OPTION_DESCR_PUA }
- enum SAVAPI_global_option {
 SAVAPI_OPTION_G_OA_EXTENSIONS_LIST = 3000, SAVAPI_OPTION_G_OA_EXCEPTED_FILES,
 SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES, SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES
 ,
 SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES, SAVAPI_OPTION_G_OA_CACHE_SCAN_NETWORK_DRIVES

- ,
SAVAPI_OPTION_G_OA_SCAN_TIMEOUT , SAVAPI_OPTION_G_OA_MALWARE_RESPONSE_TTL ,
SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT = 4000 , SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES
- ,
SAVAPI_OPTION_G_PROXY }
- enum SAVAPI_OA_SCAN_RESULT {
SAVAPI_OA_RESULT_ALLOW = 0 , SAVAPI_OA_RESULT_DENY ,
SAVAPI_OA_RESULT_DELETE , SAVAPI_OA_RESULT_RENAME ,
SAVAPI_OA_RESULT_WIPE }
- enum SAVAPI_APC_SCAN_RESULT {
SAVAPI_APC_SCAN_CONTINUE , SAVAPI_APC_SCAN_STOP ,
SAVAPI_APC_SCAN_REPORT }
- enum SAVAPI_APC_scan_mode { SAVAPI_APC_SCAN_MODE_CHECK_ONLY = 1 , SAVAPI_APC_SCAN_MODE_FULL }
- Defines the APC scan mode.*
- enum SAVAPI_APC_SCAN_STAGE {
SAVAPI_APC_STAGE_PRE_FILTER , SAVAPI_APC_STAGE_PRE_HASH_CHECK ,
SAVAPI_APC_STAGE_PRE_UPLOAD , SAVAPI_APC_STAGE_POST_SCAN }
- The stage in APC scanning in which the SAVAPI_CALLBACK_APC_SCAN callback was called.*
- enum SAVAPI_APC_SCAN_ANSWER { SAVAPI_APC_SCAN_ANSWER_CLEAN = 1 , SAVAPI_APC_SCAN_ANSWER_INFECTED }
- Scan answers that can be provided by the user for a reported file.*
- enum _SAVAPI_log_level {
SAVAPI_LOG_DEBUG = 0 , SAVAPI_LOG_INFO ,
SAVAPI_LOG_WARNING , SAVAPI_LOG_ALERT ,
SAVAPI_LOG_ERROR }
- The enumeration used to specify the SAVAPI's logging levels.*
- enum SAVAPI_engine_module_type { SAVAPI_ENGINE_MODULE_AVE = 0 , SAVAPI_ENGINE_MODULE_VDF }
- Defines the type of an engine module.*

Functions

- SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_log_callback (SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL min_level, void *user_data)
Sets the SAVAPI logging function.
- SAVAPI_EXP SAVAPI_STATUS SAVAPI_initialize (SAVAPI_GLOBAL_INIT *savapi_init)
SAVAPI initialization function Initializes the SAVAPI library, according to the parameters specified in the initialization structure. It should be called once per process, but it may be called several times per process only if SAVAPI_uninitialize() has been called in between. The latter case is useful when initializing SAVAPI in 'quick load' mode (SAVAPI_set_quickload_init()), and then uninitializing and reinitializing SAVAPI in 'normal mode' for scanning purposes.
- void SAVAPI_EXP SAVAPI_set_quickload_init ()
Sets the SAVAPI initialization mode to 'quick load' Recommended if only component versions are needed.
- SAVAPI_EXP SAVAPI_STATUS SAVAPI_uninitialize ()
SAVAPI uninitialization function Uninitializes the SAVAPI library, cleaning up all used resources. Once called, all subsequent SAVAPI calls will fail with SAVAPI_E_NOT_INITIALIZED error code.
- SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_initialize (SAVAPI_APC_GLOBAL_INIT *savapi_apc_init)
SAVAPI initialization function for APC component Initializes the APC library, according to the parameters specified in the initialization structure.
- SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_uninitialize ()
SAVAPI uninitialization function for APC component Uninitializes the APC library, cleaning up all used resources.
- SAVAPI_EXP SAVAPI_STATUS SAVAPI_APC_get_version (SAVAPI_VERSION *version)
Returns the APC version.

- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_version](#) ([SAVAPI_VERSION](#) *version)
Returns SAVAPI library version.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_create_instance](#) ([SAVAPI_INSTANCE_INIT](#) *init, [SAVAPI_FD](#) *savapi_fd)
SAVAPI factory function The function opens a connection to the SAVAPI daemon for client-mode, or, for library mode, it creates a new SAVAPI instance.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_release_instance](#) ([SAVAPI_FD](#) *savapi_fd)
Destroys a SAVAPI handler, previously created with [SAVAPI_create_instance](#). The function closes the connection to the SAVAPI daemon for client-mode, or, for library mode, it releases the SAVAPI instance.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_user_data](#) ([SAVAPI_FD](#) savapi_fd, void *user_data)
Sets user specific data. This functions sets user data that will be returned untouched as user_data member of [SAVAPI_CALLBACK_DATA](#) structure.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_user_data](#) ([SAVAPI_FD](#) savapi_fd, void **user_data)
Gets user specific data. This functions gets the user data set by [SAVAPI_set_user_data](#).
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running](#) ()
Determines if the SAVAPI daemon is running The library must be initialized with the proper daemon connection parameters for this function to run correctly.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_is_running_ex](#) (const [SAVAPI_TCHAR](#) *hostname, unsigned int port)
Determines if the SAVAPI daemon is running on the specified interface (hostname and port)
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_register_callback](#) ([SAVAPI_FD](#) savapi_fd, unsigned int callback_id, [SAVAPI_CALLBACK](#) callback)
Registers a client defined callback.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_unregister_callback](#) ([SAVAPI_FD](#) savapi_fd, unsigned int callback_id, [SAVAPI_CALLBACK](#) callback)
Unregisters a previously registered client defined callback.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_scan](#) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_TCHAR](#) *file_name)
Starts a scanning process. During the scan operation the registered callbacks may be triggered.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_simple_scan](#) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_TCHAR](#) *file_name, [SAVAPI_SIMPLE_SCAN_OUTPUT](#) *output)
Starts a scanning process during which no callbacks will be triggered.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_set](#) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_OPTION](#) option_id, [SAVAPI_TCHAR](#) *buffer)
Sets SAVAPI individual settings.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_get](#) ([SAVAPI_FD](#) savapi_fd, [SAVAPI_OPTION](#) option_id, [SAVAPI_TCHAR](#) *buffer, [SAVAPI_SIZE_T](#) *buffer_size)
Reads SAVAPI settings.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_dynamic_detect](#) ([SAVAPI_TCHAR](#) *type, int *id)
Retrieve the various types that can be detected (and dynamically turned on/off).
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_send_signal](#) ([SAVAPI_FD](#) savapi_fd, unsigned int signal_id, [SAVAPI_SIGNAL_DATA](#) *data)
Sends a signal to a specific SAVAPI instance The [SAVAPI_scan](#) may take a long amount of time to finish scanning its target and in some situations a forced abort would be desirable. In these kind of situations, [SAVAPI_send_signal](#) may help by sending signals to a running SAVAPI instance ([SAVAPI_SIGNAL_SCAN_ABORT](#) for instance).
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_set_fops](#) ([SAVAPI_FD](#) savapi_fd, void *fops_pointer, void *fops_context)
Specify the new fops who will be used by the engine.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_get_fops](#) ([SAVAPI_FD](#) savapi_fd, void **fops_pointer, void **fops_context)
Get the fops which is currently used by the engine.
- void [SAVAPI_EXP SAVAPI_free](#) (void **ptr)
Frees the memory space pointed to by ptr.
- [SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine](#) ()

- Reloads the engine from the location given at global initialization.*
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_versions_get (SAVAPI_VERSION *ave_version, SAVAPI_VERSION *avpack_version, SAVAPI_VERSION *vdf_version)`
Retrieves the engine versions: ave, avpack and vdf.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_reload_engine_ex (const SAVAPI_GLOBAL_INIT *global_init)`
Reloads the engine from the specified location.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_extract_malware_names (const SAVAPI_TCHAR *dir_path)`
Extracts the malware names from memory to disk. The purpose of the function is to reduce the amount of memory (RAM) used by the SAVAPI Library. The basic idea is to unload the malware names (which are only needed in case of alerts) from memory and dump them to disk.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_engine_modules_get (const SAVAPI_GLOBAL_INIT *init, SAVAPI_ENGINE_MODULE_CALLBACK module_func, void *user_data)`
Retrieves the engine components through the provided callback.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_global_set (SAVAPI_GLOBAL_OPTION optionId, SAVAPI_TCHAR *buffer)`
Sets SAVAPI global settings.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_initialize (SAVAPI_OA_GLOBAL_INIT *oa_global_init)`
SAVAPI initialization function for OnAccess component Initializes the OnAccess library, according to the parameters specified in the initialization structure.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_uninitialize ()`
SAVAPI uninitialization function for OnAccess component Uninitializes the OnAccess library, cleaning up all used resources.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_create_instances (SAVAPI_OA_INSTANCE_CALLBACK init_func, SAVAPI_OA_INSTANCE_CALLBACK uninit_func)`
Creates automatically the number of SAVAPI instances specified in SAVAPI_OA_initialize. For each instance created, the callback_func function will be called with the instance to be configured.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_start_scan ()`
Starts the real-time on-access scanning process. During the scan operation the instance registered callbacks may be triggered.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_OA_stop_scan ()`
Stops the real-time on-access scanning process.
- `SAVAPI_EXP SAVAPI_STATUS SAVAPI_FPC_disable_preinit ()`
Disables the pre-initialization of FPC inside the SAVAPI_initialize() function.

8.8 savapi.h

Go to the documentation of this file.

```

00001 #ifndef SAVAPI_H__
00002 #define SAVAPI_H__
00003
00004 #include "savapi_errors.h"
00005 #include "stchar.h"
00006
00055 #define SAVAPI_API_MAJOR_VERSION    5
00061 #define SAVAPI_API_MINOR_VERSION    5
00062
00070 #define SAVAPI_SYMBOL(s) STR(s)
00071 #define STR(s) #s
00072
00081 #define SAVAPI_ECAT_ERROR_IO          0
00083 #define SAVAPI_ECAT_ERROR_SCAN        1
00085 #define SAVAPI_ECAT_ERROR_UNPACK      2
00087 #define SAVAPI_ECAT_ERROR_GENERIC     3
00089 #define SAVAPI_ECAT_APC_REPORT_TTL    4
00090
00099 #define SAVAPI_ELEVEL_ERROR           0
00101 #define SAVAPI_ELEVEL_WARNING         1
00103 #define SAVAPI_ELEVEL_INFO            2
00104
00115 #define SAVAPI_FLAG_USE_TCP            1
00117 #define SAVAPI_FLAG_USE_LOCAL_SOCKET  2

```

```

00118
00127 #define SAVAPI_W_DAMAGED 1
00129 #define SAVAPI_W_OLE_DAMAGED 2
00131 #define SAVAPI_W_SUSPICIOUS 4
00133 #define SAVAPI_W_PROGRESS_ABORT 8
00135 #define SAVAPI_W_HEADER_MALFORMED 16
00141 #define SAVAPI_W_POTENTIAL_ARCH_BOMB 32
00143 #define SAVAPI_W_RATIO_EXCEEDED 64
00145 #define SAVAPI_W_MAX_EXTRACTED 128
00146
00156 #define SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE 1
00158 #define SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL 2
00160 #define SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS 4
00162 #define SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS 8
00163
00172 #define SAVAPI_I_OLEFILE 1
00174 #define SAVAPI_I_TEMPLATE 2
00176 #define SAVAPI_I_MACROS_PRESENT 4
00178 #define SAVAPI_I_ALL_MACROS_DELETED 8
00180 #define SAVAPI_I_OLE_ENCRYPTED 16
00182 #define SAVAPI_I_ACTIVE_CONTENT_PRESENT 32
00184 #define SAVAPI_I_MAILBOX 64
00186 #define SAVAPI_I_MACRO_AUTOSTART 128
00190 #define SAVAPI_I_APC_SCAN_DURATION 256
00192 #define SAVAPI_I_APC_RESULT_EXPIRY 512
00193
00211 typedef enum SAVAPI_option
00212 {
00213     /*
00214      * GET/SET options (read/write)
00215      * "SET" requests are available to configure SAVAPI. For the following requests,
00216      * a "GET" counterpart is also available and these are therefore labeled as
00217      * "read/write". Only the "SET" version is listed here although a "GET" version
00218      * also exists. The "GET" response will return the same data that is provided
00219      * with the "SET" request (although the representation of the data may be
00220      * different. For example, a "SET" request with "10K" could lead to a "GET"
00221      * response with "10240".)
00222      */
00223
00230     SAVAPI_OPTION_CWD = 1,
00237     SAVAPI_OPTION_CONF,
00242     SAVAPI_OPTION_ARCHIVE_SCAN,
00243
00250     SAVAPI_OPTION_ARCHIVE_MAX_SIZE,
00257     SAVAPI_OPTION_ARCHIVE_MAX_REC,
00264     SAVAPI_OPTION_ARCHIVE_MAX_RATIO,
00271     SAVAPI_OPTION_ARCHIVE_MAX_COUNT,
00276     SAVAPI_OPTION_MAILBOX_SCAN,
00281     SAVAPI_OPTION_HEUR_MACRO,
00292     SAVAPI_OPTION_HEUR_LEVEL,
00301     SAVAPI_OPTION_SCAN_TEMP,
00307     SAVAPI_OPTION_SCAN_TIMEOUT,
00312     SAVAPI_OPTION_REPAIR,
00317     SAVAPI_OPTION_NOTIFY_REPAIR,
00322     SAVAPI_OPTION_NOTIFY_OFFICE,
00327     SAVAPI_OPTION_NOTIFY_OFFICE_MACRO,
00332     SAVAPI_OPTION_NOTIFY_ALERTURL,
00337     SAVAPI_OPTION_DETECT_ADSPY,
00342     SAVAPI_OPTION_DETECT_APPL,
00347     SAVAPI_OPTION_DETECT_BDC,
00352     SAVAPI_OPTION_DETECT_DIAL,
00357     SAVAPI_OPTION_DETECT_GAME,
00362     SAVAPI_OPTION_DETECT_HIDDENEXT,
00367     SAVAPI_OPTION_DETECT_JOKE,
00373     SAVAPI_OPTION_DETECT_PCK,
00378     SAVAPI_OPTION_DETECT_PHISH,
00383     SAVAPI_OPTION_DETECT_SPR,
00389     SAVAPI_OPTION_IFRAMES_URL,
00394     SAVAPI_OPTION_REPORT_ENCRYPTED_MIME,
00404     SAVAPI_OPTION_SCAN_MODE,
00409     SAVAPI_OPTION_MIME_SCAN,
00414     SAVAPI_OPTION_PGP_SCAN,
00420     SAVAPI_OPTION_SCAN_PROGRESS,
00425     SAVAPI_OPTION_DETECT_ADWARE,
00430     SAVAPI_OPTION_DETECT_PFS,
00434     SAVAPI_OPTION_RESERVED1,
00438     SAVAPI_OPTION_RESERVED2,
00445     SAVAPI_OPTION_APC_CONNECTION_TIMEOUT,
00452     SAVAPI_OPTION_APC_SCAN_TIMEOUT,
00459     SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL,
00468     SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL,
00473     SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART,
00478     SAVAPI_OPTION_DETECT_PUA,
00489     SAVAPI_OPTION_APC_REPORT_SCAN_TTL,
00494     SAVAPI_OPTION_FPC,
00501     SAVAPI_OPTION_FPC_TIMEOUT,
00511     SAVAPI_OPTION_APC_PE_MODE,

```

```

00521     SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY,
00530     SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED,
00540     SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY,
00550     SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL,
00560     SAVAPI_OPTION_APC_ELF_MODE,
00570     SAVAPI_OPTION_APC_MACH_O_MODE,
00571
00572     /*
00573     * SET options (write only)
00574     * "SET" requests are available to configure SAVAPI. Usually a "SET" request also
00575     * has a "GET" request counterpart to retrieve current settings. However, the
00576     * following commands do not have a "GET" counterpart and are therefore labeled
00577     * as "write only".
00578     */
00579
00586     SAVAPI_OPTION_PRODUCT = 1000,
00588     SAVAPI_OPTION_DETECT_ALLTYPES,
00594     SAVAPI_OPTION_SCAN_TIMEOUTS,
00595
00596     /*
00597     * GET options (read only)
00598     * "GET" requests are available to retrieve current SAVAPI settings.
00599     * Usually a "GET" request also has a "SET" request counterpart to configure
00600     * SAVAPI. However, the following commands do not have a "SET" counterpart
00601     * and are therefore labeled as "read only".
00602     */
00603
00608     SAVAPI_OPTION_SAVAPI = 2000,
00610     SAVAPI_OPTION_AVE_VERSION,
00612     SAVAPI_OPTION_VDF_VERSION,
00617     SAVAPI_OPTION_PID,
00619     SAVAPI_OPTION_EXPIRE,
00621     SAVAPI_OPTION_VDFSIGCOUNT,
00627     SAVAPI_OPTION_SELECTABLE_DETECT,
00629     SAVAPI_OPTION_DESCR_ADSPY,
00631     SAVAPI_OPTION_DESCR_APPL,
00633     SAVAPI_OPTION_DESCR_BDC,
00635     SAVAPI_OPTION_DESCR_DIAL,
00637     SAVAPI_OPTION_DESCR_GAME,
00639     SAVAPI_OPTION_DESCR_HIDDENEXT,
00641     SAVAPI_OPTION_DESCR_JOKE,
00644     SAVAPI_OPTION_DESCR_PCK,
00646     SAVAPI_OPTION_DESCR_PHISH,
00648     SAVAPI_OPTION_DESCR_SPR,
00650     SAVAPI_OPTION_VDF_DATE,
00658     SAVAPI_OPTION_MALWARE_NAMES_FILE,
00660     SAVAPI_OPTION_DESCR_ADWARE,
00662     SAVAPI_OPTION_DESCR_PFS,
00664     SAVAPI_OPTION_DESCR_PUA,
00665 } SAVAPI_OPTION;
00666
00667
00682 typedef enum SAVAPI_global_option
00683 {
00701     SAVAPI_OPTION_G_OA_EXTENSIONS_LIST = 3000,
00736     SAVAPI_OPTION_G_OA_EXCEPTED_FILES,
00773     SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES,
00780     SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES,
00786     SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES,
00792     SAVAPI_OPTION_G_OA_CACHE_SCAN_NETWORK_DRIVES,
00799     SAVAPI_OPTION_G_OA_SCAN_TIMEOUT,
00811     SAVAPI_OPTION_G_OA_MALWARE_RESPONSE_TTL,
00817     SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT = 4000,
00824     SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES,
00838     SAVAPI_OPTION_G_PROXY,
00839 } SAVAPI_GLOBAL_OPTION;
00840
00846 typedef enum
00847 {
00851     SAVAPI_OA_RESULT_ALLOW = 0,
00852
00856     SAVAPI_OA_RESULT_DENY,
00857
00861     SAVAPI_OA_RESULT_DELETE,
00862
00866     SAVAPI_OA_RESULT_RENAME,
00867
00871     SAVAPI_OA_RESULT_WIPE
00872 } SAVAPI_OA_SCAN_RESULT;
00873
00879 typedef enum
00880 {
00882     SAVAPI_APC_SCAN_CONTINUE,
00884     SAVAPI_APC_SCAN_STOP,
00886     SAVAPI_APC_SCAN_REPORT,
00887 } SAVAPI_APC_SCAN_RESULT;
00888

```

```

00899 #define SAVAPI_CALLBACK_REPORT_FILE_STATUS 0
00900
00905 #define SAVAPI_CALLBACK_REPORT_ERROR 3
00906
00913 #define SAVAPI_CALLBACK_PRE_SCAN 4
00914
00920 #define SAVAPI_CALLBACK_ARCHIVE_OPEN 5
00921
00925 #define SAVAPI_CALLBACK_PROGRESS_REPORT 6
00926
00932 #define SAVAPI_CALLBACK_CONTENT_REPORT 7
00933
00938 #define SAVAPI_CALLBACK_SCAN_DETAILS_REPORT 8
00939
00944 #define SAVAPI_CALLBACK_OA_FILE_RESULT 9
00945
00964 #define SAVAPI_CALLBACK_APC_SCAN 10
00965
00974 #define SAVAPI_REPORT_ALERTURL 1
00976 #define SAVAPI_REPORT_REPAIRABLE 2
00977
00987 #define SAVAPI_REPORT_CONTENT_IFRAME 0
00988
01002 #define SAVAPI_SIGNAL_SCAN_ABORT 1
01003
01013 #define SAVAPI_SCAN_STATUS_CLEAN 0
01015 #define SAVAPI_SCAN_STATUS_INFECTED 1
01017 #define SAVAPI_SCAN_STATUS_SUSPICIOUS 2
01019 #define SAVAPI_SCAN_STATUS_ERROR 3
01021 #define SAVAPI_SCAN_STATUS_FINISHED 4
01022
01031 #define SAVAPI_FTYPE_REGULAR 4
01033 #define SAVAPI_FTYPE_ARCHIVE 1
01035 #define SAVAPI_FTYPE_IN_ARCHIVE 2
01036
01049 #define SAVAPI_FLAG_LAST_FILENAME_DEFAULT 1 « 0
01050
01060 typedef struct SAVAPI_global_init
01061 {
01063     unsigned int    api_major_version;
01064
01066     unsigned int    api_minor_version;
01067
01072     unsigned int    program_type;
01073
01079     SAVAPI_TCHAR    *engine_dirpath;
01080
01085     SAVAPI_TCHAR    *vdfs_dirpath;
01086
01088     SAVAPI_TCHAR    *avll_dirpath;
01089
01096     SAVAPI_TCHAR    *key_file_name;
01097 } SAVAPI_GLOBAL_INIT;
01098
01102 typedef enum SAVAPI_APC_scan_mode
01103 {
01107     SAVAPI_APC_SCAN_MODE_CHECK_ONLY = 1,
01108
01112     SAVAPI_APC_SCAN_MODE_FULL
01113 } SAVAPI_APC_SCAN_MODE;
01114
01118 typedef struct SAVAPI_APC_global_init
01119 {
01125     SAVAPI_TCHAR    *cert_dir;
01126
01134     SAVAPI_TCHAR    *temp_dir;
01135
01141     SAVAPI_TCHAR    *lib_dir;
01142
01146     SAVAPI_APC_SCAN_MODE apc_mode;
01147
01157     SAVAPI_SIZE_T    cache_size;
01158
01162     unsigned int    dump_cache_file;
01163
01171     SAVAPI_TCHAR*    cache_file_path;
01172
01177     unsigned int    blackout_retries;
01178
01183     unsigned int    blackout_timeout;
01184
01204     char    *proxy;
01205 } SAVAPI_APC_GLOBAL_INIT;
01206
01210 typedef struct SAVAPI_instance_init
01211 {
01215     unsigned int    flags;

```

```

01219     unsigned int connection_timeout;
01223     SAVAPI_TCHAR *host_name;
01227     unsigned int port;
01231     unsigned int scan_timeout;
01235     unsigned int get_timeout;
01239     unsigned int set_timeout;
01243     SAVAPI_TCHAR *username;
01247     SAVAPI_TCHAR *password;
01248 } SAVAPI_INSTANCE_INIT;
01249
01250
01254 typedef struct SAVAPI_file_info
01255 {
01257     SAVAPI_TCHAR *name;
01261     unsigned int type;
01263     unsigned int level;
01264 } SAVAPI_FILE_INFO;
01265
01269 typedef struct SAVAPI_malware_info
01270 {
01272     SAVAPI_TCHAR *name;
01278     SAVAPI_TCHAR *type;
01280     SAVAPI_TCHAR *message;
01282     SAVAPI_TCHAR *app_flags;
01284     unsigned int removable;
01286     unsigned short strict;
01287 } SAVAPI_MALWARE_INFO;
01288
01289
01293 typedef struct SAVAPI_pre_scan_data
01294 {
01296     unsigned int flags;
01298     SAVAPI_FILE_INFO file_info;
01299 } SAVAPI_PRESCAN_DATA;
01300
01304 typedef struct SAVAPI_archive_open_data
01305 {
01307     unsigned int flags;
01309     SAVAPI_FILE_INFO file_info;
01310 } SAVAPI_ARCHIVE_OPEN_DATA;
01311
01320 typedef struct SAVAPI_key_value
01321 {
01323     unsigned int id;
01325     unsigned int type;
01327     char *value;
01328 } SAVAPI_KEY_VALUE;
01329
01334 typedef struct SAVAPI_file_status_data
01335 {
01337     unsigned int flags;
01339     unsigned int scan_answer;
01341     SAVAPI_FILE_INFO file_info;
01347     SAVAPI_MALWARE_INFO malware_info;
01351     unsigned int warning;
01355     unsigned int info;
01356 } SAVAPI_FILE_STATUS_DATA;
01357
01362 typedef struct SAVAPI_OA_file_result_data
01363 {
01365     SAVAPI_TCHAR *filename;
01367     unsigned int pid;
01369     unsigned char *sid;
01371     unsigned long sid_len;
01372 } SAVAPI_OA_FILE_RESULT_DATA;
01373
01383 typedef struct SAVAPI_error_data
01384 {
01386     SAVAPI_FILE_INFO file_info;
01388     unsigned int category;
01390     unsigned int level;
01395     int code;
01399     SAVAPI_KEY_VALUE *options;
01400 } SAVAPI_ERROR_DATA;
01401
01405 typedef enum
01406 {
01408     SAVAPI_APC_STAGE_PRE_FILTER,
01410     SAVAPI_APC_STAGE_PRE_HASH_CHECK,
01412     SAVAPI_APC_STAGE_PRE_UPLOAD,
01414     SAVAPI_APC_STAGE_POST_SCAN,
01415 } SAVAPI_APC_SCAN_STAGE;
01416
01420 typedef enum
01421 {
01423     SAVAPI_APC_ANSWER_CLEAN = 1,
01425     SAVAPI_APC_ANSWER_INFECTED,

```

```

01426 } SAVAPI_APC_SCAN_ANSWER;
01427
01431 typedef struct
01432 {
01434     SAVAPI_APC_SCAN_ANSWER scan_answer;
01438     SAVAPI_MALWARE_INFO malware_info;
01443     unsigned int store_cache;
01448     SAVAPI_SIZE_T ttl;
01449 } SAVAPI_APC_REPORT_DATA;
01450
01460 typedef void * SAVAPI_FD;
01461
01470 typedef SAVAPI_STATUS(CC *APC_set_report_info_t)(SAVAPI_FD savapi_fd, SAVAPI_APC_REPORT_DATA *data);
01471
01481 typedef struct SAVAPI_apc_scan_data
01482 {
01484     SAVAPI_FILE_INFO file_info;
01486     SAVAPI_APC_SCAN_STAGE stage;
01491     char *hash;
01497     int risk_rating_level;
01499     void *fops_handle;
01501     void *fops_context;
01503     unsigned int flags;
01505     SAVAPI_FD savapi_fd;
01510     APC_set_report_info_t set_report_info;
01511 } SAVAPI_APC_SCAN_DATA;
01512
01516 typedef struct SAVAPI_report_progress_data
01517 {
01519     unsigned int flags;
01521     SAVAPI_TCHAR *message;
01522 } SAVAPI_REPORT_PROGRESS_DATA;
01523
01528 typedef struct SAVAPI_iframe_url_data
01529 {
01531     unsigned int attribute;
01533     SAVAPI_TCHAR *url;
01534 } SAVAPI_IFRAME_URL_DATA;
01535
01540 typedef struct SAVAPI_report_content_data
01541 {
01543     unsigned int flags;
01545     unsigned int type;
01547     SAVAPI_FILE_INFO file_info;
01549     union _content_data
01550     {
01552         SAVAPI_IFRAME_URL_DATA *iframeurl_data;
01553     } content_data;
01554 } SAVAPI_REPORT_CONTENT_DATA;
01556
01560 typedef struct SAVAPI_alert_url_data
01561 {
01563     SAVAPI_TCHAR *alert_url;
01565     SAVAPI_FILE_INFO file_info;
01566 } SAVAPI_ALERT_URL_DATA;
01567
01571 typedef struct SAVAPI_repairable_data
01572 {
01574     SAVAPI_FILE_INFO file_info;
01576     SAVAPI_MALWARE_INFO malware_info;
01577 } SAVAPI_REPAIRABLE_DATA;
01578
01582 typedef struct SAVAPI_report_scan_details_data
01583 {
01585     unsigned int flags;
01587     unsigned int type;
01589     union _scan_details_data
01590     {
01592         SAVAPI_ALERT_URL_DATA *alert_url_data;
01594         SAVAPI_REPAIRABLE_DATA *repairable_data;
01595     } scan_details_data;
01596 } SAVAPI_REPORT_SCAN_DETAILS_DATA;
01597
01601 typedef struct SAVAPI_callback_data
01602 {
01604     unsigned int type;
01606     unsigned int version;
01608     unsigned int flags;
01614     void *user_data;
01618     union specific_data
01619     {
01621         SAVAPI_PRESCAN_DATA *pre_scan_data;
01623         SAVAPI_ARCHIVE_OPEN_DATA *archive_open_data;
01625         SAVAPI_FILE_STATUS_DATA *file_status_data;
01627         SAVAPI_ERROR_DATA *error_data;
01629         SAVAPI_REPORT_PROGRESS_DATA *report_progress_data;

```

```

01631         SAVAPI_REPORT_CONTENT_DATA           *report_content_data;
01633         SAVAPI_REPORT_SCAN_DETAILS_DATA       *report_scan_details_data;
01635         SAVAPI_OA_FILE_RESULT_DATA            *oa_file_result_data;
01637         SAVAPI_APC_SCAN_DATA                   *apc_scan_data;
01639         void                                   *private_data;
01640     } callback_data;
01641 } SAVAPI_CALLBACK_DATA;
01642
01647 typedef struct SAVAPI_simple_scan_file_data
01648 {
01650     SAVAPI_TCHAR *name;
01652     unsigned int scan_answer;
01657     unsigned int error_level;
01662     unsigned int error_code;
01664     SAVAPI_TCHAR *malware_name;
01670     SAVAPI_TCHAR *malware_type;
01671 } SAVAPI_SIMPLE_SCAN_FILE_DATA;
01672
01676 typedef struct SAVAPI_simple_scan_statistics
01677 {
01681     unsigned int total_files;
01683     unsigned int infections;
01685     unsigned int suspicions;
01687     unsigned int errors;
01688 } SAVAPI_SIMPLE_SCAN_STATISTICS;
01689
01694 typedef struct SAVAPI_simple_scan_output
01695 {
01699     SAVAPI_SIMPLE_SCAN_FILE_DATA *files;
01701     unsigned int count;
01703     SAVAPI_SIMPLE_SCAN_STATISTICS stats;
01704 } SAVAPI_SIMPLE_SCAN_OUTPUT;
01705
01709 typedef struct SAVAPI_signal_data
01710 {
01712     unsigned int signal_id;
01713
01720     void *signal_data;
01721 } SAVAPI_SIGNAL_DATA;
01722
01726 typedef struct SAVAPI_command_data
01727 {
01729     unsigned int signal_id;
01730
01737     /*      union signal_specific_data
01738     {
01739     } signal_data;
01740     */
01741     void *command_data;
01742 } SAVAPI_COMMAND_DATA;
01743
01747 typedef struct SAVAPI_version
01748 {
01750     unsigned int major;
01752     unsigned int minor;
01754     unsigned int build_major;
01756     unsigned int build_minor;
01757 } SAVAPI_VERSION;
01758
01762 typedef enum _SAVAPI_log_level
01763 {
01767     SAVAPI_LOG_DEBUG = 0,
01769     SAVAPI_LOG_INFO,
01771     SAVAPI_LOG_WARNING,
01773     SAVAPI_LOG_ALERT,
01775     SAVAPI_LOG_ERROR
01776 } SAVAPI_LOG_LEVEL;
01777
01781 typedef enum SAVAPI_engine_module_type
01782 {
01783     /* AVE module type */
01784     SAVAPI_ENGINE_MODULE_AVE = 0,
01785
01786     /* VDF module type */
01787     SAVAPI_ENGINE_MODULE_VDF
01788 } SAVAPI_ENGINE_MODULE_TYPE;
01789
01794 typedef struct SAVAPI_OA_global_init
01795 {
01802     unsigned int threads_number;
01803
01812     unsigned int scm_pending_time;
01813 } SAVAPI_OA_GLOBAL_INIT;
01814
01825 typedef int(CC *SAVAPI_CALLBACK)(SAVAPI_CALLBACK_DATA *data);
01826
01834 typedef void(CC *SAVAPI_LOG_CALLBACK)(SAVAPI_LOG_LEVEL log_level, const SAVAPI_TCHAR *message, void

```

```

    *user_data);
01835
01842 typedef int(CC *SAVAPI_ENGINE_MODULE_CALLBACK)(const SAVAPI_TCHAR *name, SAVAPI_ENGINE_MODULE_TYPE
type, void *user_data);
01843
01848 typedef int(CC *SAVAPI_OA_INSTANCE_CALLBACK)(SAVAPI_FD savapi_fd);
01849
01850 #ifdef __cplusplus
01851 extern "C"
01852 {
01853 #endif
01854
01869 typedef SAVAPI_STATUS (CC *SAVAPI_set_log_callback_t)(SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL
min_level, void *user_data);
01870 typedef SAVAPI_STATUS (CC *SAVAPI_initialize_t)(SAVAPI_GLOBAL_INIT *savapi_init);
01871 typedef void(CC *SAVAPI_set_quickload_init_t)();
01872 typedef SAVAPI_STATUS (CC *SAVAPI_uninitialize_t)();
01873 typedef SAVAPI_STATUS (CC *SAVAPI_get_version_t)(SAVAPI_VERSION *version);
01874 typedef SAVAPI_STATUS (CC *SAVAPI_engine_versions_get_t)(SAVAPI_VERSION *ave_version, SAVAPI_VERSION
*avpack_version, SAVAPI_VERSION *vdf_version);
01875 typedef SAVAPI_STATUS (CC *SAVAPI_APC_get_version_t)(SAVAPI_VERSION *apc_version);
01876 typedef SAVAPI_STATUS (CC *SAVAPI_create_instance_t)(SAVAPI_INSTANCE_INIT *init, SAVAPI_FD
*savapi_fd);
01877 typedef SAVAPI_STATUS (CC *SAVAPI_release_instance_t)(SAVAPI_FD *savapi_fd);
01878 typedef SAVAPI_STATUS (CC *SAVAPI_set_user_data_t)(SAVAPI_FD savapi_fd, void *user_data);
01879 typedef SAVAPI_STATUS (CC *SAVAPI_get_user_data_t)(SAVAPI_FD savapi_fd, void **user_data);
01880 typedef SAVAPI_STATUS (CC *SAVAPI_is_running_ex_t)(const SAVAPI_TCHAR *hostname, unsigned int port);
01881 typedef SAVAPI_STATUS (CC *SAVAPI_register_callback_t)(SAVAPI_FD savapi_fd, unsigned int callback_id,
SAVAPI_CALLBACK callback);
01882 typedef SAVAPI_STATUS (CC *SAVAPI_unregister_callback_t)(SAVAPI_FD savapi_fd, unsigned int
callback_id, SAVAPI_CALLBACK callback);
01883 typedef SAVAPI_STATUS (CC *SAVAPI_scan_t)(SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name);
01884 typedef SAVAPI_STATUS (CC *SAVAPI_simple_scan_t)(SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name,
SAVAPI_SIMPLE_SCAN_OUTPUT *output);
01885 typedef SAVAPI_STATUS (CC *SAVAPI_set_t)(SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR
*buffer);
01886 typedef SAVAPI_STATUS (CC *SAVAPI_get_t)(SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR
*buffer, SAVAPI_SIZE_T *buffer_size);
01887 typedef SAVAPI_STATUS (CC *SAVAPI_send_signal_t)(SAVAPI_FD savapi_fd, unsigned int signal_id,
SAVAPI_SIGNAL_DATA* data);
01888 typedef SAVAPI_STATUS (CC *SAVAPI_set_fops_t)(SAVAPI_FD savapi_fd, void *fops_pointer, void
*fops_context);
01889 typedef SAVAPI_STATUS (CC *SAVAPI_get_fops_t)(SAVAPI_FD savapi_fd, void **fops_pointer, void
**fops_context);
01890 typedef void (CC *SAVAPI_free_t)(void **ptr);
01891 typedef SAVAPI_STATUS (CC *SAVAPI_reload_engine_ex_t)(const SAVAPI_GLOBAL_INIT *global_init);
01892 typedef SAVAPI_STATUS (CC *SAVAPI_extract_malware_names_t)(const SAVAPI_TCHAR *dir_path);
01893 typedef SAVAPI_STATUS (CC *SAVAPI_engine_modules_get_t)(const SAVAPI_GLOBAL_INIT *init,
SAVAPI_ENGINE_MODULE_CALLBACK module_func, void *user_data);
01894 typedef SAVAPI_STATUS (CC *SAVAPI_global_set_t)(SAVAPI_GLOBAL_OPTION option_id, SAVAPI_TCHAR *buffer);
01895
01896 typedef SAVAPI_STATUS (CC *SAVAPI_APC_initialize_t)(SAVAPI_APC_GLOBAL_INIT *savapi_apc_init);
01897 typedef SAVAPI_STATUS (CC *SAVAPI_APC_uninitialize_t)();
01898
01899 typedef SAVAPI_STATUS (CC *SAVAPI_OA_initialize_t)(SAVAPI_OA_GLOBAL_INIT *savapi_oa_init);
01900 typedef SAVAPI_STATUS (CC *SAVAPI_OA_uninitialize_t)();
01901 typedef SAVAPI_STATUS (CC *SAVAPI_OA_create_instances_t)(SAVAPI_OA_INSTANCE_CALLBACK init_func,
SAVAPI_OA_INSTANCE_CALLBACK uninit_func);
01902 typedef SAVAPI_STATUS (CC *SAVAPI_OA_start_scan_t)();
01903 typedef SAVAPI_STATUS (CC *SAVAPI_OA_stop_scan_t)();
01904
01923 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_set_log_callback(SAVAPI_LOG_CALLBACK log_fct, SAVAPI_LOG_LEVEL
min_level, void *user_data);
01924
01939 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_initialize(SAVAPI_GLOBAL_INIT *savapi_init);
01940
01951 void SAVAPI_EXP CC SAVAPI_set_quickload_init();
01952
01963 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_uninitialize();
01964
01977 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_APC_initialize(SAVAPI_APC_GLOBAL_INIT *savapi_apc_init);
01978
01989 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_APC_uninitialize();
01990
01999 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_APC_get_version(SAVAPI_VERSION *version);
02000
02007 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_get_version(SAVAPI_VERSION *version);
02008
02019 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_create_instance(SAVAPI_INSTANCE_INIT *init, SAVAPI_FD *savapi_fd);
02020
02034 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_release_instance(SAVAPI_FD *savapi_fd);
02035
02048 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_set_user_data(SAVAPI_FD savapi_fd, void *user_data);
02049
02058 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_get_user_data(SAVAPI_FD savapi_fd, void **user_data);
02059
02072 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_is_running();

```



```

02073
02084 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_is_running_ex(const SAVAPI_TCHAR *hostname, unsigned int port);
02085
02098 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_register_callback(SAVAPI_FD savapi_fd, unsigned int callback_id,
SAVAPI_CALLBACK callback);
02099
02111 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_unregister_callback(SAVAPI_FD savapi_fd, unsigned int callback_id,
SAVAPI_CALLBACK callback);
02112
02135 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_scan(SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name);
02136
02168 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_simple_scan(SAVAPI_FD savapi_fd, SAVAPI_TCHAR *file_name,
SAVAPI_SIMPLE_SCAN_OUTPUT *output);
02169
02179 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_set(SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR
*buffer);
02180
02193 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_get(SAVAPI_FD savapi_fd, SAVAPI_OPTION option_id, SAVAPI_TCHAR
*buffer, SAVAPI_SIZE_T *buffer_size);
02194
02202 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_get_dynamic_detect(SAVAPI_TCHAR *type, int *id);
02203
02220 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_send_signal(SAVAPI_FD savapi_fd, unsigned int signal_id,
SAVAPI_SIGNAL_DATA* data);
02221
02231 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_set_fops(SAVAPI_FD savapi_fd, void *fops_pointer, void
*fops_context);
02232
02242 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_get_fops(SAVAPI_FD savapi_fd, void **fops_pointer, void
**fops_context);
02243
02249 void SAVAPI_EXP CC SAVAPI_free(void **ptr);
02250
02273 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_reload_engine();
02274
02285 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_engine_versions_get(SAVAPI_VERSION *ave_version, SAVAPI_VERSION
*avpack_version, SAVAPI_VERSION *vdf_version);
02286
02302 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_reload_engine_ex(const SAVAPI_GLOBAL_INIT *global_init);
02303
02332 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_extract_malware_names(const SAVAPI_TCHAR *dir_path);
02333
02343 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_engine_modules_get(const SAVAPI_GLOBAL_INIT *init,
SAVAPI_ENGINE_MODULE_CALLBACK module_func, void *user_data);
02344
02353 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_global_set(SAVAPI_GLOBAL_OPTION optionId, SAVAPI_TCHAR *buffer);
02354
02370 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_OA_initialize(SAVAPI_OA_GLOBAL_INIT *oa_global_init);
02371
02383 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_OA_uninitialize();
02384
02399 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_OA_create_instances(SAVAPI_OA_INSTANCE_CALLBACK init_func,
SAVAPI_OA_INSTANCE_CALLBACK uninit_func);
02400
02407 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_OA_start_scan();
02408
02414 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_OA_stop_scan();
02415
02423 SAVAPI_EXP SAVAPI_STATUS CC SAVAPI_FPC_disable_preinit();
02424
02426 #ifdef __cplusplus
02427 }
02428 #endif
02429 #endif /* SAVAPI_H__ */

```

8.9 savapi_errors.h File Reference

Typedefs

- typedef enum [SAVAPI_status](#) SAVAPI_STATUS

Enumerations

- enum [SAVAPI_status](#) {
[SAVAPI_S_OK](#) = 0 , [SAVAPI_E_INVALID_PARAMETER](#) = 1 ,

```

SAVAPI_E_ALREADY_INITIALIZED = 2 , SAVAPI_E_NOT_INITIALIZED = 3 ,
SAVAPI_E_BUFFER_TOO_SMALL = 4 , SAVAPI_E_CONNECTION_MODE_NOT_SET = 5 ,
SAVAPI_E_HOSTNAME_NOT_SET = 6 , SAVAPI_E_NO_MEMORY = 7 ,
SAVAPI_E_VDF_NOT_FOUND = 8 , SAVAPI_E_VDF_READ = 9 ,
SAVAPI_E_VDF_CRC = 10 , SAVAPI_E_VDF_VERSION = 11 ,
SAVAPI_E_WRONG_ENGINE = 12 , SAVAPI_E_ENGINE_NOT_FOUND = 13 ,
SAVAPI_E_SELFCHK_PATCHED = 14 , SAVAPI_E_SELFCHK_FILE_ERR = 15 ,
SAVAPI_E_SELFCHK_FILE_CRC = 16 , SAVAPI_E_KEYFILE = 17 ,
SAVAPI_E_INTERNAL = 18 , SAVAPI_E_NOT_SUPPORTED = 19 ,
SAVAPI_E_RESULT_FILE_NOT_FOUND = 20 , SAVAPI_E_OPTION_NOT_SUPPORTED = 21 ,
SAVAPI_E_HIT_MAX_REC = 22 , SAVAPI_E_HIT_MAX_SIZE = 23 ,
SAVAPI_E_HIT_MAX_RATIO = 24 , SAVAPI_E_ENCRYPTED = 25 ,
SAVAPI_E_UNSUPPORTED = 26 , SAVAPI_E_PROC_ERROR = 27 ,
SAVAPI_E_INCOMPLETE = 28 , SAVAPI_E_PARTIAL = 29 ,
SAVAPI_E_HIT_MAX_COUNT = 30 , SAVAPI_E_ABORTED = 31 ,
SAVAPI_E_TIMEOUT = 32 , SAVAPI_E_FILE_OPEN = 33 ,
SAVAPI_E_FILE_READ = 34 , SAVAPI_E_FILE_WRITE = 35 ,
SAVAPI_E_INVALID_VALUE = 36 , SAVAPI_E_CHDIR_FAILED = 37 ,
SAVAPI_E_NOT_ABSOLUTE_PATH = 38 , SAVAPI_E_DIR_NOT_EXISTS = 39 ,
SAVAPI_E_MATCHED = 40 , SAVAPI_E_CONVERSION_FAILED = 41 ,
SAVAPI_E_CONNECTION_FAILED = 42 , SAVAPI_E_RECEIVE_FAILED = 43 ,
SAVAPI_E_SEND_FAILED = 44 , SAVAPI_E_OPTION_VALUE_INVALID = 45 ,
SAVAPI_E_REPAIR_FAILED = 46 , SAVAPI_E_FILE_CREATE = 47 ,
SAVAPI_E_FILE_DELETE = 48 , SAVAPI_E_FILE_CLOSE = 49 ,
SAVAPI_E_UNKNOWN = 50 , SAVAPI_E_PREFIX_SET = 51 ,
SAVAPI_E_PREFIX_GET = 52 , SAVAPI_E_INVALID_QUERY = 53 ,
SAVAPI_E_KEY_NO_KEYFILE = 54 , SAVAPI_E_KEY_ACCESS_DENIED = 55 ,
SAVAPI_E_KEY_INVALID_HEADER = 56 , SAVAPI_E_KEY_KEYFILE_VERSION = 57 ,
SAVAPI_E_KEY_NO_LICENSE = 58 , SAVAPI_E_KEY_FILE_INVALID = 59 ,
SAVAPI_E_KEY_RECORD_INVALID = 60 , SAVAPI_E_KEY_EVAL_VERSION = 61 ,
SAVAPI_E_KEY_DEMO_VERSION = 62 , SAVAPI_E_KEY_ILLEGAL_LICENSE = 63 ,
SAVAPI_E_KEY_EXPIRED = 64 , SAVAPI_E_KEY_READ = 65 ,
SAVAPI_E_LICENSE_RESTRICTION = 66 , SAVAPI_E_LOADING_ENGINE_MODULES = 67 ,
SAVAPI_E_BUSY = 68 , SAVAPI_E_ENCRYPTED_MIME = 69 ,
SAVAPI_E_NON_ADDRESSABLE = 70 , SAVAPI_E_MEMORY_LIMIT = 71 ,
SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ = 72 , SAVAPI_E_PROC_BAD_HEADER = 73 ,
SAVAPI_E_PROC_INVALID_COMPRESSED_DATA = 74 , SAVAPI_E_PROC_OBSOLETE = 75 ,
SAVAPI_E_PROC_BAD_FORMAT = 76 , SAVAPI_E_PROC_HEADER_CRC = 77 ,
SAVAPI_E_PROC_DATA_CRC = 78 , SAVAPI_E_PROC_FILE_CRC = 79 ,
SAVAPI_E_PROC_BAD_TABLE = 80 , SAVAPI_E_PROC_UNEXPECTED_EOF = 81 ,
SAVAPI_E_PROC_ARCHIVE_HANDLE = 82 , SAVAPI_E_PROC_NO_FILES_TO_EXTRACT = 83 ,
SAVAPI_E_PROC_CALLBACK = 84 , SAVAPI_E_PROC_TOTAL_LOSS = 85 ,
SAVAPI_E_APC_ERROR = 86 , SAVAPI_E_APC_CONNECTION = 87 ,
SAVAPI_E_APC_NOT_SUPPORTED = 88 , SAVAPI_E_APC_TIMEOUT = 89 ,
SAVAPI_E_APC_TEMPORARILY_DISABLED = 90 , SAVAPI_E_APC_INCOMPLETE = 91 ,
SAVAPI_E_APC_NO_LICENSE = 92 , SAVAPI_E_APC_AUTHENTICATION = 93 ,
SAVAPI_E_APC_AUTH_RETRY_LATER = 94 , SAVAPI_E_APC_RANDOM_ID = 95 ,
SAVAPI_E_APC_NOT_INITIALIZED = 96 , SAVAPI_E_APC_ALREADY_INITIALIZED = 97 ,
SAVAPI_E_APC_DISABLED = 98 , SAVAPI_E_APC_TIMEOUT_RESTRICTION = 99 ,
SAVAPI_E_APC_UNKNOWN_CATEGORY = 100 , SAVAPI_E_APC_QUOTA = 101 ,
SAVAPI_E_UNSUPPORTED_COMPRESSION = 1000 , SAVAPI_E_OA_NOT_INITIALIZED = 2000 ,
SAVAPI_E_OA_INITIALIZED = 2001 , SAVAPI_E_OA_NO_INSTANCES_CREATED = 2002 ,
SAVAPI_E_OA_INSTANCES_CREATED = 2003 , SAVAPI_E_OA_NO_SCAN_IN_PROGRESS = 2004 ,
SAVAPI_E_OA_SCAN_IN_PROGRESS = 2005 , SAVAPI_E_OA_NO_LICENSE = 2006 ,
SAVAPI_E_OA_ERROR = 2007 , SAVAPI_E_OA_NO_PRIVILEGES = 2008 ,
SAVAPI_E_OA_DRIVERS = 2009 , SAVAPI_E_FPC_TIMEOUT_RESTRICTION = 3000 ,
SAVAPI_S_INFECTED = 4000 , SAVAPI_S_SUSPICIOUS = 4001 ,
SAVAPI_E_SCAN_ERROR = 4002 }

```

8.10 savapi_errors.h

[Go to the documentation of this file.](#)

```

00001 #ifndef SAVAPI_ERRORS_H__
00002 #define SAVAPI_ERRORS_H__
00003
00008 typedef enum SAVAPI_status
00009 {
00013     SAVAPI_S_OK = 0,
00019     SAVAPI_E_INVALID_PARAMETER = 1,
00025     SAVAPI_E_ALREADY_INITIALIZED = 2,
00031     SAVAPI_E_NOT_INITIALIZED = 3,
00037     SAVAPI_E_BUFFER_TOO_SMALL = 4,
00044     SAVAPI_E_CONNECTION_MODE_NOT_SET = 5,
00051     SAVAPI_E_HOSTNAME_NOT_SET = 6,
00056     SAVAPI_E_NO_MEMORY = 7,
00062     SAVAPI_E_VDF_NOT_FOUND = 8,
00067     SAVAPI_E_VDF_READ = 9,
00073     SAVAPI_E_VDF_CRC = 10,
00079     SAVAPI_E_VDF_VERSION = 11,
00085     SAVAPI_E_WRONG_ENGINE = 12,
00090     SAVAPI_E_ENGINE_NOT_FOUND = 13,
00096     SAVAPI_E_SELFCHK_PATCHED = 14,
00101     SAVAPI_E_SELFCHK_FILE_ERR = 15,
00107     SAVAPI_E_SELFCHK_FILE_CRC = 16,
00111     SAVAPI_E_KEYFILE = 17,
00118     SAVAPI_E_INTERNAL = 18,
00130     SAVAPI_E_NOT_SUPPORTED = 19,
00135     SAVAPI_E_RESULT_FILE_NOT_FOUND = 20,
00141     SAVAPI_E_OPTION_NOT_SUPPORTED = 21,
00148     SAVAPI_E_HIT_MAX_REC = 22,
00154     SAVAPI_E_HIT_MAX_SIZE = 23,
00160     SAVAPI_E_HIT_MAX_RATIO = 24,
00167     SAVAPI_E_ENCRYPTED = 25,
00173     SAVAPI_E_UNSUPPORTED = 26,
00179     SAVAPI_E_PROC_ERROR = 27,
00184     SAVAPI_E_INCOMPLETE = 28,
00191     SAVAPI_E_PARTIAL = 29,
00197     SAVAPI_E_HIT_MAX_COUNT = 30,
00202     SAVAPI_E_ABORTED = 31,
00207     SAVAPI_E_TIMEOUT = 32,
00212     SAVAPI_E_FILE_OPEN = 33,
00219     SAVAPI_E_FILE_READ = 34,
00225     SAVAPI_E_FILE_WRITE = 35,
00233     SAVAPI_E_INVALID_VALUE = 36,
00240     SAVAPI_E_CHDIR_FAILED = 37,
00246     SAVAPI_E_NOT_ABSOLUTE_PATH = 38,
00253     SAVAPI_E_DIR_NOT_EXISTS = 39,
00258     SAVAPI_E_MATCHED = 40,
00266     SAVAPI_E_CONVERSION_FAILED = 41,
00272     SAVAPI_E_CONNECTION_FAILED = 42,
00278     SAVAPI_E_RECEIVE_FAILED = 43,
00284     SAVAPI_E_SEND_FAILED = 44,
00290     SAVAPI_E_OPTION_VALUE_INVALID = 45,
00294     SAVAPI_E_REPAIR_FAILED = 46,
00300     SAVAPI_E_FILE_CREATE = 47,
00306     SAVAPI_E_FILE_DELETE = 48,
00312     SAVAPI_E_FILE_CLOSE = 49,
00317     SAVAPI_E_UNKNOWN = 50,
00323     SAVAPI_E_PREFIX_SET = 51,
00329     SAVAPI_E_PREFIX_GET = 52,
00336     SAVAPI_E_INVALID_QUERY = 53,
00340     SAVAPI_E_KEY_NO_KEYFILE = 54,
00344     SAVAPI_E_KEY_ACCESS_DENIED = 55,
00348     SAVAPI_E_KEY_INVALID_HEADER = 56,
00352     SAVAPI_E_KEY_KEYFILE_VERSION = 57,
00356     SAVAPI_E_KEY_NO_LICENSE = 58,
00360     SAVAPI_E_KEY_FILE_INVALID = 59,
00364     SAVAPI_E_KEY_RECORD_INVALID = 60,
00368     SAVAPI_E_KEY_EVAL_VERSION = 61,
00372     SAVAPI_E_KEY_DEMO_VERSION = 62,
00376     SAVAPI_E_KEY_ILLEGAL_LICENSE = 63,
00380     SAVAPI_E_KEY_EXPIRED = 64,
00384     SAVAPI_E_KEY_READ = 65,
00389     SAVAPI_E_LICENSE_RESTRICTION = 66,
00395     SAVAPI_E_LOADING_ENGINE_MODULES = 67,
00402     SAVAPI_E_BUSY = 68,
00407     SAVAPI_E_ENCRYPTED_MIME = 69,
00414     SAVAPI_E_NON_ADDRESSABLE = 70,
00420     SAVAPI_E_MEMORY_LIMIT = 71,
00425     SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ = 72,
00430     SAVAPI_E_PROC_BAD_HEADER = 73,
00436     SAVAPI_E_PROC_INVALID_COMPRESSED_DATA = 74,
00442     SAVAPI_E_PROC_OBSOLETE = 75,
00448     SAVAPI_E_PROC_BAD_FORMAT = 76,

```

```

00453     SAVAPI_E_PROC_HEADER_CRC = 77,
00458     SAVAPI_E_PROC_DATA_CRC = 78,
00463     SAVAPI_E_PROC_FILE_CRC = 79,
00468     SAVAPI_E_PROC_BAD_TABLE = 80,
00473     SAVAPI_E_PROC_UNEXPECTED_EOF = 81,
00478     SAVAPI_E_PROC_ARCHIVE_HANDLE = 82,
00483     SAVAPI_E_PROC_NO_FILES_TO_EXTRACT = 83,
00488     SAVAPI_E_PROC_CALLBACK = 84,
00493     SAVAPI_E_PROC_TOTAL_LOSS = 85,
00497     SAVAPI_E_APC_ERROR = 86,
00502     SAVAPI_E_APC_CONNECTION = 87,
00507     SAVAPI_E_APC_NOT_SUPPORTED = 88,
00513     SAVAPI_E_APC_TIMEOUT = 89,
00517     SAVAPI_E_APC_TEMPORARILY_DISABLED = 90,
00522     SAVAPI_E_APC_INCOMPLETE = 91,
00526     SAVAPI_E_APC_NO_LICENSE = 92,
00530     SAVAPI_E_APC_AUTHENTICATION = 93,
00534     SAVAPI_E_APC_AUTH_RETRY_LATER = 94,
00540     SAVAPI_E_APC_RANDOM_ID = 95,
00546     SAVAPI_E_APC_NOT_INITIALIZED = 96,
00552     SAVAPI_E_APC_ALREADY_INITIALIZED = 97,
00557     SAVAPI_E_APC_DISABLED = 98,
00563     SAVAPI_E_APC_TIMEOUT_RESTRICTION = 99,
00571     SAVAPI_E_APC_UNKNOWN_CATEGORY = 100,
00576     SAVAPI_E_APC_QUOTA = 101,
00577
00582     SAVAPI_E_UNSUPPORTED_COMPRESSION = 1000,
00583
00589     SAVAPI_E_OA_NOT_INITIALIZED = 2000,
00595     SAVAPI_E_OA_INITIALIZED = 2001,
00600     SAVAPI_E_OA_NO_INSTANCES_CREATED = 2002,
00606     SAVAPI_E_OA_INSTANCES_CREATED = 2003,
00612     SAVAPI_E_OA_NO_SCAN_IN_PROGRESS = 2004,
00618     SAVAPI_E_OA_SCAN_IN_PROGRESS = 2005,
00622     SAVAPI_E_OA_NO_LICENSE = 2006,
00626     SAVAPI_E_OA_ERROR = 2007,
00631     SAVAPI_E_OA_NO_PRIVILEGES = 2008,
00635     SAVAPI_E_OA_DRIVERS = 2009,
00636
00642     SAVAPI_E_FPC_TIMEOUT_RESTRICTION = 3000,
00643
00644     /*
00645     * \brief The SAVAPI Simple Scan has detected at least one infected file.
00646     * \note This code is only returned by the \ref SAVAPI_simple_scan function.
00647     */
00648     SAVAPI_S_INFECTED = 4000,
00649
00650     /*
00651     * \brief The SAVAPI Simple Scan has detected at least one suspicious file, but no infected ones.
00652     * \note This code is only returned by the \ref SAVAPI_simple_scan function.
00653     */
00654     SAVAPI_S_SUSPICIOUS = 4001,
00655
00656     /*
00657     * \brief The SAVAPI Simple Scan has returned an error for at least one file,
00658     * but no infected or suspicious ones
00659     * \note This code is only returned by the \ref SAVAPI_simple_scan function.
00660     */
00661     SAVAPI_E_SCAN_ERROR = 4002,
00662 } SAVAPI_STATUS;
00663
00666 #endif /* SAVAPI_ERRORS_H__ */

```

8.11 savapi_plg.h File Reference

Data Structures

- `struct SAVAPI_PLG_info_t`
Structure containing the plugin's information (name, version etc.)

Macros

- `#define SAVAPI_PLG_EXP`
- `#define SAVAPI_PLG_tchar_t char`

- `#define _T(x) x`
- `#define SAVAPI_PLG_MAX_STR 1024`
Defines the maximum size for the plugin's information fields.
- `#define SAVAPI_PLG_MAX_VER_STR 64`
- `#define SAVAPI_PLG_VER_MAJ 0 /** Savapi's plugins interface major version */`
- `#define SAVAPI_PLG_VER_MIN 1 /** Savapi's plugins interface minor version */`
- `#define SAVAPI_EPLG_SUCCESS 0 /** Operation ended with success */`
- `#define SAVAPI_EPLG_UKNW_INTERFACE 1 /** The requested interface is not supported */`
- `#define SAVAPI_EPLG_INVALID 2 /** An invalid parameter has been provided */`
- `#define SAVAPI_EPLG_INIT 3 /** Plugin or instance already initialized */`
- `#define SAVAPI_EPLG_NO_MEM 4 /** Memory allocation error */`
- `#define SAVAPI_EPLG_INTERNAL 5 /** An internal error occurred */`
- `#define SAVAPI_PLG_MAIN_FUNC savapi_plg_main`

Typedefs

- `typedef void SAVAPI_PLG_options_t`
- `typedef void SAVAPI_PLG_inst_options_t`
- `typedef void SAVAPI_PLG_instance_t`
- `typedef int SAVAPI_PLG_status_t`
Plugin's return codes type.
- `typedef SAVAPI_PLG_status_t(* SAVAPI_PLG_init_t) (const SAVAPI_PLG_options_t *options)`
Prototype for a plugin's initialize function.
- `typedef void(* SAVAPI_PLG_uninit_t) ()`
Prototype for a plugin's uninitialize function.
- `typedef SAVAPI_PLG_status_t(* SAVAPI_PLG_instance_create_t) (SAVAPI_PLG_instance_t **instance, const SAVAPI_PLG_tchar_t *interface_id, const SAVAPI_PLG_inst_options_t *options)`
Prototype for a plugin's instance create function.
- `typedef void(* SAVAPI_PLG_instance_release_t) (SAVAPI_PLG_instance_t **instance)`
Prototype for a plugin's instance destroy function.
- `typedef struct _SAVAPI_PLG_info_t SAVAPI_PLG_info_t`
Structure containing the plugin's information (name, version etc.)
- `typedef SAVAPI_PLG_status_t(* SAVAPI_PLG_main_t) (const SAVAPI_PLG_info_t **savapi_plg_info)`
Main function that must be exported by the plugin.

Functions

- `SAVAPI_PLG_EXP SAVAPI_PLG_status_t SAVAPI_PLG_MAIN_FUNC (const SAVAPI_PLG_info_t **plg↔_info)`

8.12 savapi_plg.h

Go to the documentation of this file.

```
00001 #ifndef _PLUGINS_SAVAPI_PLG_H_
00002 #define _PLUGINS_SAVAPI_PLG_H_
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif
00027
00032 #ifdef _WINDOWS
00034 #   define SAVAPI_PLG_EXP __declspec(dllexport)
00035 #   ifdef CC
00036 #       define CC _cdecl
```

```

00037 #      endif /* CC */
00038 #else /* UNIXes */
00039 #      define SAVAPI_PLG_EXP
00040 #      ifndef CC
00041 #          define CC /* calling convention may not be defined on unix */
00042 #      endif /* CC */
00043 #endif /* _WINDOWS */
00044
00045 #ifdef _WINDOWS
00046     #include <tchar.h>
00047     #define SAVAPI_PLG_tchar_t TCHAR
00048 #else /* UNIXes - using Utf8 encoding */
00049     #define SAVAPI_PLG_tchar_t char
00050     #define _T(x) x
00051 #endif /* _WINDOWS */
00052
00053 #define SAVAPI_PLG_MAX_STR      1024
00054 #define SAVAPI_PLG_MAX_VER_STR  64
00055
00056 #define SAVAPI_PLG_VER_MAJ      0
00057 #define SAVAPI_PLG_VER_MIN      1
00058 #define SAVAPI_EPLG_SUCCESS     0
00059 #define SAVAPI_EPLG_UKNW_INTERFACE 1
00060 #define SAVAPI_EPLG_INVALID     2
00061 #define SAVAPI_EPLG_INIT        3
00062 #define SAVAPI_EPLG_NO_MEM      4
00063 #define SAVAPI_EPLG_INTERNAL    5
00064
00065 typedef void SAVAPI_PLG_options_t;
00066 typedef void SAVAPI_PLG_inst_options_t;
00067 typedef void SAVAPI_PLG_instance_t;
00068 typedef int SAVAPI_PLG_status_t;
00069 typedef SAVAPI_PLG_status_t (CC *SAVAPI_PLG_init_t) (const SAVAPI_PLG_options_t *options);
00070 typedef void (CC *SAVAPI_PLG_uninit_t) ();
00071 typedef SAVAPI_PLG_status_t (CC *SAVAPI_PLG_instance_create_t) (SAVAPI_PLG_instance_t **instance,
00072     const SAVAPI_PLG_tchar_t *interface_id, const SAVAPI_PLG_inst_options_t *options);
00073 typedef void (CC *SAVAPI_PLG_instance_release_t) (SAVAPI_PLG_instance_t **instance);
00074
00075 typedef struct _SAVAPI_PLG_info_t
00076 {
00077     SAVAPI_PLG_init_t      init;
00078     SAVAPI_PLG_uninit_t    uninit;
00079     SAVAPI_PLG_instance_create_t instance_create;
00080     SAVAPI_PLG_instance_release_t instance_release;
00081     int interface_ver_maj;
00082     int interface_ver_min;
00083     SAVAPI_PLG_tchar_t name[SAVAPI_PLG_MAX_STR];
00084     SAVAPI_PLG_tchar_t version[SAVAPI_PLG_MAX_VER_STR];
00085     void *res;
00086 } SAVAPI_PLG_info_t;
00087
00088 typedef SAVAPI_PLG_status_t (CC *SAVAPI_PLG_main_t) (const SAVAPI_PLG_info_t **savapi_plg_info);
00089
00090 #define SAVAPI_PLG_MAIN_FUNC      savapi_plg_main
00091 SAVAPI_PLG_EXP SAVAPI_PLG_status_t CC SAVAPI_PLG_MAIN_FUNC(const SAVAPI_PLG_info_t **plg_info);
00092
00093 #ifdef __cplusplus
00094 }
00095 #endif
00096 #endif /* _PLUGINS_SAVAPI_PLG_H */

```

8.13 savapi_plg_fops.h File Reference

Data Structures

- struct [_SAVAPI_PLG_fops_instance_t](#)

Macros

- #define [SAVAPI_PLG_AVE_FOPS_T](#)("SAVAPI_AVE_FOPS")

Typedefs

- typedef const char ** [SAVAPI_PLG_fops_options_t](#)
- typedef void [SAVAPI_PLG_fops_inst_options_t](#)
- typedef struct [_SAVAPI_PLG_fops_instance_t](#) [SAVAPI_PLG_fops_instance_t](#)

8.14 savapi_plg_fops.h

[Go to the documentation of this file.](#)

```
00001 #ifndef SAVAPI_PLG_FOPS_H
00002 #define SAVAPI_PLG_FOPS_H
00003
00004 #ifdef __cplusplus
00005 extern "C" {
00006 #endif
00007
00008 #include "fops.h"
00009
00014 #define SAVAPI_PLG_AVE_FOPS      _T("SAVAPI_AVE_FOPS")
00015
00016 typedef const char** SAVAPI_PLG_fops_options_t;
00017 typedef void SAVAPI_PLG_fops_inst_options_t;
00019 typedef struct _SAVAPI_PLG_fops_instance_t
00020 {
00021     AVE_FOPS fops;
00022 } SAVAPI_PLG_fops_instance_t;
00023
00029 #ifdef __cplusplus
00030 }
00031 #endif
00032
00033 #endif /* SAVAPI_PLG_FOPS_H */
```

8.15 stchar.h File Reference

Conversion between SAVAPI_TCHAR and char/TCHAR.

Macros

- #define [SAVAPI_EXP](#)
- #define [SAVAPI_TCHAR](#) char
- #define [SAVAPI_SIZE_T](#) unsigned int

Typedefs

- typedef [SAVAPI_STATUS](#)(* [STCHARToChar_t](#)) (char **pDest, const [SAVAPI_TCHAR](#) *pSrc)
- typedef [SAVAPI_STATUS](#)(* [CharToSTCHAR_t](#)) ([SAVAPI_TCHAR](#) **pDest, const char *pSrc)

Functions

- [SAVAPI_EXP SAVAPI_STATUS CharToSTCHAR](#) ([SAVAPI_TCHAR](#) **pDest, const char *pSrc)
Convert from char//TCHAR to a SAVAPI_TCHAR.
- [SAVAPI_EXP SAVAPI_STATUS STCHARToChar](#) (char **pDest, const [SAVAPI_TCHAR](#) *pSrc)
Convert from a SAVAPI_TCHAR buffer to a char//TCHAR.

8.15.1 Detailed Description

Conversion between SAVAPI_TCHAR and char/TCHAR.

UNICODE (WIN) SAVAPI_TCHAR = wchar_t(UCS2) | char(UTF-8) UNICODE (UNIX) SAVAPI_TCHAR = wchar_t(UCS2) | char(locale) ANSI (WIN + UNIX) SAVAPI_TCHAR = char(locale) | char(locale)

8.15.2 Macro Definition Documentation

8.15.2.1 SAVAPI_EXP

```
#define SAVAPI_EXP
```

8.15.2.2 SAVAPI_SIZE_T

```
#define SAVAPI_SIZE_T unsigned int
```

8.15.2.3 SAVAPI_TCHAR

```
#define SAVAPI_TCHAR char
```

8.15.3 Function Documentation

8.15.3.1 CharToSTCHAR()

```
SAVAPI_EXP SAVAPI_STATUS CharToSTCHAR (  
    SAVAPI_TCHAR ** pDest,  
    const char * pSrc )
```

Convert from char//TCHAR to a SAVAPI_TCHAR.

Parameters

<i>pDest</i>	[OUT]: Pointer to a SAVAPI_TCHAR that will hold the converted buffer
<i>pSrc</i>	[IN]: The buffer to convert.

Returns

SAVAPI_S_OK for success or an error code

Note

The pDest parameter will be internally allocated. The caller is responsible to release the memory by calling [SAVAPI_free\(\)](#) on pDest.

8.15.3.2 STCHARToChar()

```
SAVAPI_EXP SAVAPI_STATUS STCHARToChar (
    char ** pDest,
    const SAVAPI_TCHAR * pSrc )
```

Convert from a SAVAPI_TCHAR buffer to a char//TCHAR.

Parameters

<i>pDest</i>	[OUT]: Pointer to a char that will hold the converted buffer
<i>pSrc</i>	[IN]: Pointer to the SAVAPI_TCHAR to convert

Returns

SAVAPI_S_OK for success or an error code

Note

The pDest parameter will be internally allocated. The caller is responsible to release the memory by calling [SAVAPI_free\(\)](#) on pDest.

8.16 stchar.h

[Go to the documentation of this file.](#)

```
00001 #if !defined(_SAVAPI_TCHAR_CONVERSION_H_)
00002 #define _SAVAPI_TCHAR_CONVERSION_H_
00012 #include "savapi_errors.h"
00013
00014 #ifdef __cplusplus
00015 extern "C" {
00016 #endif
00017
00018 /* Checks for the platform defines: _WINDOWS or _UNIX */
00019 #if !defined(_WINDOWS) && !defined(_UNIX)
00020 # error "Please define a supported platform!"
00021 #endif /* !defined(_WINDOWS) && !defined(_UNIX) */
00022
00023 /* Checks for the build encoding defines: _UNICODE or _ANSI */
00024 #if !defined(_UNICODE) && !defined(_ANSI)
00025 # error "Please define a supported encoding!"
00026 #endif /* !defined(_UNICODE) && !defined(_ANSI) */
00027
00028 #ifdef _WINDOWS
00029 # ifdef MAKINGDLL_SAVAPI
00030 # define SAVAPI_EXP __declspec(dllexport)
00031 # else /* when using the DLL */
00032 # define SAVAPI_EXP __declspec(dllimport)
00033 # endif /* MAKINGDLL_SAVAPI */
00034 # ifndef CC
00035 # define CC _cdecl
00036 # endif /* CC */
00037 #else /* on UNIX */
00038 # define SAVAPI_EXP
00039 # ifndef CC
00040 # define CC /* calling convention may not be defined on unix */
00041 # endif /* CC */
00042 #endif /* _WINDOWS */
00043
00044 #if defined(_WINDOWS)
00045 # include <tchar.h>
00046 # define SAVAPI_TCHAR TCHAR
00047 # define SAVAPI_SIZE_T size_t
00048 #else /* _UNIX */
00049 # if defined(_UNICODE) /* UNIX, UNICODE */
00050 # if defined(_SAVAPI_UTF8)
00051 # include <stddef.h>
```

```

00052 #         define SAVAPI_TCHAR char
00053 #     else /* wchar_t */
00054 #         include <wchar.h>
00055 #         define SAVAPI_TCHAR wchar_t
00056 #     endif /* _SAVAPI_UTF8 */
00057 #     define SAVAPI_SIZE_T size_t
00058 #     else /* UNIX, ANSI */
00059 #         define SAVAPI_TCHAR char
00060 #         define SAVAPI_SIZE_T unsigned int
00061 #     endif /* defined(_UNICODE) */
00062 #endif /* _WINDOWS */
00063
00071 typedef SAVAPI_STATUS (CC *STCHARToChar_t)(char **pDest, const SAVAPI_TCHAR *pSrc);
00072 typedef SAVAPI_STATUS (CC *CharToSTCHAR_t)(SAVAPI_TCHAR **pDest, const char *pSrc);
00073 #ifdef _WINDOWS
00074     typedef SAVAPI_STATUS (CC *STCHARToTCHAR_t)(TCHAR **pDest, const SAVAPI_TCHAR *pSrc);
00075     typedef SAVAPI_STATUS (CC *TCHARToSTCHAR_t)(SAVAPI_TCHAR **pDest, const TCHAR *pSrc);
00076 #endif /* _WINDOWS */
00077
00090 SAVAPI_EXP SAVAPI_STATUS CC CharToSTCHAR(SAVAPI_TCHAR **pDest, const char *pSrc);
00091 #ifdef _WINDOWS
00092 SAVAPI_EXP SAVAPI_STATUS CC TCHARToSTCHAR(SAVAPI_TCHAR **pDest, const TCHAR *pSrc);
00093 #endif /* _WINDOWS */
00094
00095
00105 SAVAPI_EXP SAVAPI_STATUS CC STCHARToChar(char **pDest, const SAVAPI_TCHAR *pSrc);
00106 #ifdef _WINDOWS
00107 SAVAPI_EXP SAVAPI_STATUS CC STCHARToTCHAR(TCHAR **pDest, const SAVAPI_TCHAR *pSrc);
00108 #endif /* _WINDOWS */
00109
00110 #ifdef __cplusplus
00111 }
00112 #endif
00113
00114 #endif /* _SAVAPI_TCHAR_CONVERSION_H__ */

```

Index

- [_AVE_STRUCT_FILE_OPERATIONS, 109](#)
 - [fops_access, 110](#)
 - [fops_close, 111](#)
 - [fops_flush, 111](#)
 - [fops_free, 111](#)
 - [fops_get_last_error, 112](#)
 - [fops_getc, 112](#)
 - [fops_getfattr, 112](#)
 - [fops_getfsz, 113](#)
 - [fops_getfs, 113](#)
 - [fops_malloc, 114](#)
 - [fops_open, 114](#)
 - [fops_putc, 115](#)
 - [fops_puts, 115](#)
 - [fops_read, 116](#)
 - [fops_rename, 116](#)
 - [fops_seek, 117](#)
 - [fops_setfattr, 117](#)
 - [fops_tell, 118](#)
 - [fops_ungetc, 118](#)
 - [fops_unlink, 119](#)
 - [fops_write, 119](#)
- [_SAVAPI_PLG_fops_instance_t, 120](#)
 - [fops, 121](#)
- [_SAVAPI_PLG_info_t, 121](#)
 - [init, 121](#)
 - [instance_create, 122](#)
 - [instance_release, 122](#)
 - [interface_ver_maj, 122](#)
 - [interface_ver_min, 122](#)
 - [name, 122](#)
 - [res, 122](#)
 - [uninit, 122](#)
 - [version, 123](#)
- [_SAVAPI_log_level](#)
 - [SAVAPI structures, 55](#)
- [_T](#)
 - [Plugin defines, 101](#)
- [_cdecl](#)
 - [fopstypes.h, 173](#)
- [_fattr_t](#)
 - [fops.h, 169](#)
- [_fpos_t](#)
 - [fops.h, 169](#)
- [_user_fops](#)
 - [fops.h, 170](#)
- [alert_url](#)
 - [SAVAPI_alert_url_data, 124](#)
- [alert_url_data](#)
 - [SAVAPI_report_scan_details_data::_scan_details_data, 123](#)
- [apc_mode](#)
 - [SAVAPI_APC_global_init, 125](#)
- [apc_scan_data](#)
 - [SAVAPI_callback_data::specific_data, 160](#)
- [APC_set_report_info_t](#)
 - [SAVAPI typedefs, 57](#)
- [API version, 13](#)
 - [SAVAPI_API_MAJOR_VERSION, 13](#)
 - [SAVAPI_API_MINOR_VERSION, 13](#)
- [api_major_version](#)
 - [SAVAPI_global_init, 139](#)
- [api_minor_version](#)
 - [SAVAPI_global_init, 139](#)
- [app_flags](#)
 - [SAVAPI_malware_info, 146](#)
- [archive_open_data](#)
 - [SAVAPI_callback_data::specific_data, 160](#)
- [asc2bin](#)
 - [asc_bin.h, 163](#)
- [asc_bin.h, 163](#)
 - [asc2bin, 163](#)
 - [bin2asc, 164](#)
 - [bin2hex, 164](#)
 - [hex2bin, 165](#)
- [attribute](#)
 - [SAVAPI_iframe_url_data, 141](#)
- [AVE_FOPS](#)
 - [fops.h, 170](#)
- [avll_dirpath](#)
 - [SAVAPI_global_init, 139](#)
- [bin2asc](#)
 - [asc_bin.h, 164](#)
- [bin2hex](#)
 - [asc_bin.h, 164](#)
- [bin2hex_t](#)
 - [SAVAPI_hex2bin function pointers, 11](#)
- [blackout_retries](#)
 - [SAVAPI_APC_global_init, 125](#)
- [blackout_timeout](#)
 - [SAVAPI_APC_global_init, 125](#)
- [build_major](#)
 - [SAVAPI_version, 159](#)
- [build_minor](#)
 - [SAVAPI_version, 159](#)
- [cache_file_path](#)
 - [SAVAPI_APC_global_init, 126](#)

- cache_size
 - SAVAPI_APC_global_init, 126
- callback_data
 - SAVAPI_callback_data, 133
- Callbacks' ids, 39
 - SAVAPI_CALLBACK_APC_SCAN, 40
 - SAVAPI_CALLBACK_ARCHIVE_OPEN, 40
 - SAVAPI_CALLBACK_CONTENT_REPORT, 41
 - SAVAPI_CALLBACK_OA_FILE_RESULT, 41
 - SAVAPI_CALLBACK_PRE_SCAN, 41
 - SAVAPI_CALLBACK_PROGRESS_REPORT, 41
 - SAVAPI_CALLBACK_REPORT_ERROR, 42
 - SAVAPI_CALLBACK_REPORT_FILE_STATUS, 42
 - SAVAPI_CALLBACK_SCAN_DETAILS_REPORT, 42
- category
 - SAVAPI_error_data, 135
- cert_dir
 - SAVAPI_APC_global_init, 126
- CharToSTCHAR
 - stchar.h, 198
- CharToSTCHAR_t
 - SAVAPI STCHAR function pointers, 108
- check_free_mem
 - fops.h, 170
- code
 - SAVAPI_error_data, 135
- command_data
 - SAVAPI_command_data, 134
- connection_timeout
 - SAVAPI_instance_init, 142
- content_data
 - SAVAPI_report_content_data, 151
- count
 - SAVAPI_simple_scan_output, 157
- dump_cache_file
 - SAVAPI_APC_global_init, 126
- e_tempname
 - fops.h, 170
- engine_dirpath
 - SAVAPI_global_init, 140
- Error levels, 15
 - SAVAPI_ELEVEL_ERROR, 15
 - SAVAPI_ELEVEL_INFO, 15
 - SAVAPI_ELEVEL_WARNING, 15
- Error or information categories, 14
 - SAVAPI_ECAT_APC_REPORT_TTL, 14
 - SAVAPI_ECAT_ERROR_GENERIC, 14
 - SAVAPI_ECAT_ERROR_IO, 14
 - SAVAPI_ECAT_ERROR_SCAN, 14
 - SAVAPI_ECAT_ERROR_UNPACK, 14
- error_code
 - SAVAPI_simple_scan_file_data, 155
- error_data
 - SAVAPI_callback_data::specific_data, 160
- error_level
 - SAVAPI_simple_scan_file_data, 155
- errors
 - SAVAPI_simple_scan_statistics, 158
- f_check_mem
 - fops.h, 170
- File types, 45
 - SAVAPI_FTYPE_ARCHIVE, 45
 - SAVAPI_FTYPE_IN_ARCHIVE, 45
 - SAVAPI_FTYPE_REGULAR, 46
- file_info
 - SAVAPI_alert_url_data, 124
 - SAVAPI_apc_scan_data, 129
 - SAVAPI_archive_open_data, 132
 - SAVAPI_error_data, 135
 - SAVAPI_file_status_data, 138
 - SAVAPI_pre_scan_data, 149
 - SAVAPI_repairable_data, 150
 - SAVAPI_report_content_data, 151
- file_status_data
 - SAVAPI_callback_data::specific_data, 160
- filename
 - SAVAPI_OA_file_result_data, 147
- Filename flags, 46
 - SAVAPI_FLAG_LAST_FILENAME_DEFAULT, 46
- files
 - SAVAPI_simple_scan_output, 157
- flags
 - SAVAPI_apc_scan_data, 129
 - SAVAPI_archive_open_data, 132
 - SAVAPI_callback_data, 133
 - SAVAPI_file_status_data, 138
 - SAVAPI_instance_init, 142
 - SAVAPI_pre_scan_data, 149
 - SAVAPI_report_content_data, 151
 - SAVAPI_report_progress_data, 152
 - SAVAPI_report_scan_details_data, 153
- fops
 - _SAVAPI_PLG_fops_instance_t, 121
- fops.h, 166
 - _fattr_t, 169
 - _fpos_t, 169
 - _user_fops, 170
 - AVE_FOPS, 170
 - check_free_mem, 170
 - e_tempname, 170
 - f_check_mem, 170
 - FOPS_ERROR, 168
 - FOPS_HANDLE, 170
 - FOPS_INVALID_HANDLE, 168
 - OPEN_CR, 168
 - OPEN_RO, 169
 - OPEN_RW, 169
 - SEEK_CUR, 169
 - SEEK_END, 169
 - SEEK_SET, 169
- fops_access
 - _AVE_STRUCT_FILE_OPERATIONS, 110
- fops_close
 - _AVE_STRUCT_FILE_OPERATIONS, 111

- fops_context
 - SAVAPI_apc_scan_data, 130
- FOPS_ERROR
 - fops.h, 168
- fops_flush
 - _AVE_STRUCT_FILE_OPERATIONS, 111
- fops_free
 - _AVE_STRUCT_FILE_OPERATIONS, 111
- fops_get_last_error
 - _AVE_STRUCT_FILE_OPERATIONS, 112
- fops_getc
 - _AVE_STRUCT_FILE_OPERATIONS, 112
- fops_getfattr
 - _AVE_STRUCT_FILE_OPERATIONS, 112
- fops_getfsize
 - _AVE_STRUCT_FILE_OPERATIONS, 113
- fops_gets
 - _AVE_STRUCT_FILE_OPERATIONS, 113
- FOPS_HANDLE
 - fops.h, 170
- fops_handle
 - SAVAPI_apc_scan_data, 130
- FOPS_INVALID_HANDLE
 - fops.h, 168
- fops_malloc
 - _AVE_STRUCT_FILE_OPERATIONS, 114
- fops_open
 - _AVE_STRUCT_FILE_OPERATIONS, 114
- fops_putc
 - _AVE_STRUCT_FILE_OPERATIONS, 115
- fops_puts
 - _AVE_STRUCT_FILE_OPERATIONS, 115
- fops_read
 - _AVE_STRUCT_FILE_OPERATIONS, 116
- fops_rename
 - _AVE_STRUCT_FILE_OPERATIONS, 116
- fops_seek
 - _AVE_STRUCT_FILE_OPERATIONS, 117
- fops_setfattr
 - _AVE_STRUCT_FILE_OPERATIONS, 117
- fops_tell
 - _AVE_STRUCT_FILE_OPERATIONS, 118
- fops_ungetc
 - _AVE_STRUCT_FILE_OPERATIONS, 118
- fops_unlink
 - _AVE_STRUCT_FILE_OPERATIONS, 119
- fops_write
 - _AVE_STRUCT_FILE_OPERATIONS, 119
- fopstypes.h, 172
 - _cdecl, 173
 - INT16, 173
 - INT32, 173
 - INT64, 173
 - INT8, 173
 - UINT16, 173
 - UINT32, 174
 - UINT64, 174
 - UINT8, 174
- get_timeout
 - SAVAPI_instance_init, 142
- hash
 - SAVAPI_apc_scan_data, 130
- hex2bin
 - asc_bin.h, 165
- hex2bin_t
 - SAVAPI_hex2bin_function_pointers, 11
- host_name
 - SAVAPI_instance_init, 142
- id
 - SAVAPI_key_value, 145
- Iframes informations, 18
 - SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL, 18
 - SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE, 18
 - SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS, 18
 - SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS, 18
- iframeurl_data
 - SAVAPI_report_content_data::_content_data, 120
- infections
 - SAVAPI_simple_scan_statistics, 158
- info
 - SAVAPI_file_status_data, 138
- init
 - _SAVAPI_PLG_info_t, 121
- Initialization flags, 16
 - SAVAPI_FLAG_USE_LOCAL_SOCKET, 16
 - SAVAPI_FLAG_USE_TCP, 16
- instance_create
 - _SAVAPI_PLG_info_t, 122
- instance_release
 - _SAVAPI_PLG_info_t, 122
- INT16
 - fopstypes.h, 173
- INT32
 - fopstypes.h, 173
- INT64
 - fopstypes.h, 173
- INT8
 - fopstypes.h, 173
- interface_ver_maj
 - _SAVAPI_PLG_info_t, 122
- interface_ver_min
 - _SAVAPI_PLG_info_t, 122
- key_file_name
 - SAVAPI_global_init, 140
- level
 - SAVAPI_error_data, 136
 - SAVAPI_file_info, 136
- lib_dir
 - SAVAPI_APC_global_init, 127

- major
 - SAVAPI_version, 159
- malware_info
 - SAVAPI_APC_REPORT_DATA, 128
 - SAVAPI_file_status_data, 138
 - SAVAPI_repairable_data, 150
- malware_name
 - SAVAPI_simple_scan_file_data, 155
- malware_type
 - SAVAPI_simple_scan_file_data, 156
- message
 - SAVAPI_malware_info, 146
 - SAVAPI_report_progress_data, 152
- minor
 - SAVAPI_version, 159
- name
 - _SAVAPI_PLG_info_t, 122
 - SAVAPI_file_info, 137
 - SAVAPI_malware_info, 146
 - SAVAPI_simple_scan_file_data, 156
- oa_file_result_data
 - SAVAPI_callback_data::specific_data, 161
- OPEN_CR
 - fops.h, 168
- OPEN_RO
 - fops.h, 169
- OPEN_RW
 - fops.h, 169
- options
 - SAVAPI_error_data, 136
- password
 - SAVAPI_instance_init, 143
- pid
 - SAVAPI_OA_file_result_data, 147
- Plugin defines, 101
 - _T, 101
 - SAVAPI_PLG_EXP, 101
 - SAVAPI_PLG_MAX_STR, 102
 - SAVAPI_PLG_MAX_VER_STR, 102
 - SAVAPI_PLG_tchar_t, 102
 - SAVAPI_PLG_VER_MAJ, 102
 - SAVAPI_PLG_VER_MIN, 102
- Plugin return statuses, 102
 - SAVAPI_EPLG_INIT, 103
 - SAVAPI_EPLG_INTERNAL, 103
 - SAVAPI_EPLG_INVALID, 103
 - SAVAPI_EPLG_NO_MEM, 103
 - SAVAPI_EPLG_SUCCESS, 103
 - SAVAPI_EPLG_UKNW_INTERFACE, 103
- Plugin's general typedefs, 104
 - SAVAPI_PLG_init_t, 105
 - SAVAPI_PLG_instance_create_t, 105
 - SAVAPI_PLG_instance_release_t, 105
 - SAVAPI_PLG_status_t, 105
 - SAVAPI_PLG_uninit_t, 105
- Plugin's specific typedefs, 104
 - SAVAPI_PLG_inst_options_t, 104
 - SAVAPI_PLG_instance_t, 104
 - SAVAPI_PLG_options_t, 104
- Plugin's structures definitions, 106
 - SAVAPI_PLG_info_t, 106
 - SAVAPI_PLG_MAIN_FUNC, 106, 107
 - SAVAPI_PLG_main_t, 106
- port
 - SAVAPI_instance_init, 143
- pre_scan_data
 - SAVAPI_callback_data::specific_data, 161
- private_data
 - SAVAPI_callback_data::specific_data, 161
- program_type
 - SAVAPI_global_init, 140
- proxy
 - SAVAPI_APC_global_init, 127
- removable
 - SAVAPI_malware_info, 146
- repairable_data
 - SAVAPI_report_scan_details_data::scan_details_data, 123
- report_content_data
 - SAVAPI_callback_data::specific_data, 161
- report_progress_data
 - SAVAPI_callback_data::specific_data, 161
- report_scan_details_data
 - SAVAPI_callback_data::specific_data, 161
- res
 - _SAVAPI_PLG_info_t, 122
- risk_rating_level
 - SAVAPI_apc_scan_data, 130
- SAVAPI APC scan callback return values, 39
 - SAVAPI_APC_SCAN_CONTINUE, 39
 - SAVAPI_APC_SCAN_REPORT, 39
 - SAVAPI_APC_SCAN_RESULT, 39
 - SAVAPI_APC_SCAN_STOP, 39
- SAVAPI constants, 12
 - SAVAPI_SYMBOL, 12
 - STR, 12
- SAVAPI function pointers, 59
- SAVAPI functions, 66
 - SAVAPI_APC_get_version, 68
 - SAVAPI_APC_initialize, 68
 - SAVAPI_APC_uninitialize, 69
 - SAVAPI_create_instance, 69
 - SAVAPI_engine_modules_get, 70
 - SAVAPI_engine_versions_get, 70
 - SAVAPI_extract_malware_names, 71
 - SAVAPI_FPC_disable_preinit, 71
 - SAVAPI_free, 72
 - SAVAPI_get, 72
 - SAVAPI_get_dynamic_detect, 73
 - SAVAPI_get_fops, 73
 - SAVAPI_get_user_data, 74
 - SAVAPI_get_version, 74
 - SAVAPI_global_set, 74

- SAVAPI_initialize, 75
- SAVAPI_is_running, 75
- SAVAPI_is_running_ex, 76
- SAVAPI_OA_create_instances, 76
- SAVAPI_OA_initialize, 77
- SAVAPI_OA_start_scan, 77
- SAVAPI_OA_stop_scan, 78
- SAVAPI_OA_uninitialize, 78
- SAVAPI_register_callback, 78
- SAVAPI_release_instance, 79
- SAVAPI_reload_engine, 79
- SAVAPI_reload_engine_ex, 80
- SAVAPI_scan, 81
- SAVAPI_send_signal, 81
- SAVAPI_set, 82
- SAVAPI_set_fops, 83
- SAVAPI_set_log_callback, 83
- SAVAPI_set_quickload_init, 84
- SAVAPI_set_user_data, 84
- SAVAPI_simple_scan, 85
- SAVAPI_uninitialize, 85
- SAVAPI_unregister_callback, 86
- SAVAPI global options, 33
 - SAVAPI_GLOBAL_OPTION, 33
 - SAVAPI_global_option, 33
 - SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES, 37
 - SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT, 37
 - SAVAPI_OPTION_G_OA_CACHE_SCAN_NETWORK_DRIVES, 37
 - SAVAPI_OPTION_G_OA_EXCEPTED_FILES, 35
 - SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES, 36
 - SAVAPI_OPTION_G_OA_EXTENSIONS_LIST, 34
 - SAVAPI_OPTION_G_OA_MALWARE_RESPONSE_TTL, 37
 - SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES, 36
 - SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES, 36
 - SAVAPI_OPTION_G_OA_SCAN_TIMEOUT, 37
 - SAVAPI_OPTION_G_PROXY, 38
- SAVAPI hex2bin function pointers, 11
 - bin2hex_t, 11
 - hex2bin_t, 11
- SAVAPI main function pointers, 59
 - SAVAPI_APC_get_version_t, 61
 - SAVAPI_APC_initialize_t, 61
 - SAVAPI_APC_uninitialize_t, 61
 - SAVAPI_create_instance_t, 61
 - SAVAPI_engine_modules_get_t, 61
 - SAVAPI_engine_versions_get_t, 61
 - SAVAPI_extract_malware_names_t, 61
 - SAVAPI_free_t, 62
 - SAVAPI_get_fops_t, 62
 - SAVAPI_get_t, 62
 - SAVAPI_get_user_data_t, 62
 - SAVAPI_get_version_t, 62
 - SAVAPI_global_set_t, 62
 - SAVAPI_initialize_t, 62
 - SAVAPI_is_running_ex_t, 63
 - SAVAPI_OA_create_instances_t, 63
 - SAVAPI_OA_initialize_t, 63
 - SAVAPI_OA_start_scan_t, 63
 - SAVAPI_OA_stop_scan_t, 63
 - SAVAPI_OA_uninitialize_t, 63
 - SAVAPI_register_callback_t, 63
 - SAVAPI_release_instance_t, 64
 - SAVAPI_reload_engine_ex_t, 64
 - SAVAPI_scan_t, 64
 - SAVAPI_send_signal_t, 64
 - SAVAPI_set_fops_t, 64
 - SAVAPI_set_log_callback_t, 64
 - SAVAPI_set_quickload_init_t, 64
 - SAVAPI_set_t, 65
 - SAVAPI_set_user_data_t, 65
 - SAVAPI_simple_scan_t, 65
 - SAVAPI_uninitialize_t, 65
 - SAVAPI_unregister_callback_t, 65
- SAVAPI OnAccess result, 38
 - SAVAPI_OA_RESULT_ALLOW, 38
 - SAVAPI_OA_RESULT_DELETE, 38
 - SAVAPI_OA_RESULT_DENY, 38
 - SAVAPI_OA_RESULT_RENAME, 38
 - SAVAPI_OA_RESULT_WIPE, 38
 - SAVAPI_OA_SCAN_RESULT, 38
- SAVAPI options, 21
 - SAVAPI_OPTION, 22
 - SAVAPI_option, 22
 - SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL, 27
 - SAVAPI_OPTION_APC_CONNECTION_TIMEOUT, 27
 - SAVAPI_OPTION_APC_ELF_MODE, 31
 - SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY, 30
 - SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED, 30
 - SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL, 30
 - SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY, 29
 - SAVAPI_OPTION_APC_MACH_O_MODE, 31
 - SAVAPI_OPTION_APC_PE_MODE, 29
 - SAVAPI_OPTION_APC_REPORT_SCAN_TTL, 28
 - SAVAPI_OPTION_APC_SCAN_TIMEOUT, 27
 - SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL, 28
 - SAVAPI_OPTION_ARCHIVE_MAX_COUNT, 23
 - SAVAPI_OPTION_ARCHIVE_MAX_RATIO, 23
 - SAVAPI_OPTION_ARCHIVE_MAX_REC, 23
 - SAVAPI_OPTION_ARCHIVE_MAX_SIZE, 23
 - SAVAPI_OPTION_ARCHIVE_SCAN, 22
 - SAVAPI_OPTION_AVE_VERSION, 32
 - SAVAPI_OPTION_CONF, 22

- SAVAPI_OPTION_CWD, 22
- SAVAPI_OPTION_DESCR_ADSPY, 32
- SAVAPI_OPTION_DESCR_ADWARE, 32
- SAVAPI_OPTION_DESCR_APPL, 32
- SAVAPI_OPTION_DESCR_BDC, 32
- SAVAPI_OPTION_DESCR_DIAL, 32
- SAVAPI_OPTION_DESCR_GAME, 32
- SAVAPI_OPTION_DESCR_HIDDENEXT, 32
- SAVAPI_OPTION_DESCR_JOKE, 32
- SAVAPI_OPTION_DESCR_PCK, 32
- SAVAPI_OPTION_DESCR_PFS, 32
- SAVAPI_OPTION_DESCR_PHISH, 32
- SAVAPI_OPTION_DESCR_PUA, 32
- SAVAPI_OPTION_DESCR_SPR, 32
- SAVAPI_OPTION_DETECT_ADSPY, 25
- SAVAPI_OPTION_DETECT_ADWARE, 26
- SAVAPI_OPTION_DETECT_ALLTYPES, 31
- SAVAPI_OPTION_DETECT_APPL, 25
- SAVAPI_OPTION_DETECT_BDC, 25
- SAVAPI_OPTION_DETECT_DIAL, 25
- SAVAPI_OPTION_DETECT_GAME, 25
- SAVAPI_OPTION_DETECT_HIDDENEXT, 25
- SAVAPI_OPTION_DETECT_JOKE, 25
- SAVAPI_OPTION_DETECT_PCK, 25
- SAVAPI_OPTION_DETECT_PFS, 27
- SAVAPI_OPTION_DETECT_PHISH, 25
- SAVAPI_OPTION_DETECT_PUA, 28
- SAVAPI_OPTION_DETECT_SPR, 26
- SAVAPI_OPTION_EXPIRE, 32
- SAVAPI_OPTION_FPC, 28
- SAVAPI_OPTION_FPC_TIMEOUT, 29
- SAVAPI_OPTION_HEUR_LEVEL, 24
- SAVAPI_OPTION_HEUR_MACRO, 23
- SAVAPI_OPTION_IFRAMES_URL, 26
- SAVAPI_OPTION_MAILBOX_SCAN, 23
- SAVAPI_OPTION_MALWARE_NAMES_FILE, 32
- SAVAPI_OPTION_MIME_SCAN, 26
- SAVAPI_OPTION_NOTIFY_ALERTURL, 25
- SAVAPI_OPTION_NOTIFY_OFFICE, 24
- SAVAPI_OPTION_NOTIFY_OFFICE_MACRO, 25
- SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART, 28
- SAVAPI_OPTION_NOTIFY_REPAIR, 24
- SAVAPI_OPTION_PGP_SCAN, 26
- SAVAPI_OPTION_PID, 32
- SAVAPI_OPTION_PRODUCT, 31
- SAVAPI_OPTION_REPAIR, 24
- SAVAPI_OPTION_REPORT_ENCRYPTED_MIME, 26
- SAVAPI_OPTION_RESERVED1, 27
- SAVAPI_OPTION_RESERVED2, 27
- SAVAPI_OPTION_SAVAPI, 32
- SAVAPI_OPTION_SCAN_MODE, 26
- SAVAPI_OPTION_SCAN_PROGRESS, 26
- SAVAPI_OPTION_SCAN_TEMP, 24
- SAVAPI_OPTION_SCAN_TIMEOUT, 24
- SAVAPI_OPTION_SCAN_TIMEOUTS, 31
- SAVAPI_OPTION_SELECTABLE_DETECT, 32
- SAVAPI_OPTION_VDF_DATE, 32
- SAVAPI_OPTION_VDF_VERSION, 32
- SAVAPI_OPTION_VDFSIGCOUNT, 32
- SAVAPI report content types, 43
 - SAVAPI_REPORT_CONTENT_IFRAME, 43
- SAVAPI report scan details types, 42
 - SAVAPI_REPORT_ALERTURL, 42
 - SAVAPI_REPORT_REPAIRABLE, 43
- SAVAPI return codes, 87
 - SAVAPI_E_ABORTED, 92
 - SAVAPI_E_ALREADY_INITIALIZED, 88
 - SAVAPI_E_APC_ALREADY_INITIALIZED, 99
 - SAVAPI_E_APC_AUTH_RETRY_LATER, 98
 - SAVAPI_E_APC_AUTHENTICATION, 98
 - SAVAPI_E_APC_CONNECTION, 98
 - SAVAPI_E_APC_DISABLED, 99
 - SAVAPI_E_APC_ERROR, 98
 - SAVAPI_E_APC_INCOMPLETE, 98
 - SAVAPI_E_APC_NO_LICENSE, 98
 - SAVAPI_E_APC_NOT_INITIALIZED, 99
 - SAVAPI_E_APC_NOT_SUPPORTED, 98
 - SAVAPI_E_APC_QUOTA, 99
 - SAVAPI_E_APC_RANDOM_ID, 99
 - SAVAPI_E_APC_TEMPORARILY_DISABLED, 98
 - SAVAPI_E_APC_TIMEOUT, 98
 - SAVAPI_E_APC_TIMEOUT_RESTRICTION, 99
 - SAVAPI_E_APC_UNKNOWN_CATEGORY, 99
 - SAVAPI_E_BUFFER_TOO_SMALL, 89
 - SAVAPI_E_BUSY, 96
 - SAVAPI_E_CHDIR_FAILED, 93
 - SAVAPI_E_CONNECTION_FAILED, 94
 - SAVAPI_E_CONNECTION_MODE_NOT_SET, 89
 - SAVAPI_E_CONVERSION_FAILED, 94
 - SAVAPI_E_DIR_NOT_EXISTS, 93
 - SAVAPI_E_ENCRYPTED, 92
 - SAVAPI_E_ENCRYPTED_MIME, 96
 - SAVAPI_E_ENGINE_NOT_FOUND, 90
 - SAVAPI_E_FILE_CLOSE, 95
 - SAVAPI_E_FILE_CREATE, 94
 - SAVAPI_E_FILE_DELETE, 95
 - SAVAPI_E_FILE_OPEN, 93
 - SAVAPI_E_FILE_READ, 93
 - SAVAPI_E_FILE_WRITE, 93
 - SAVAPI_E_FPC_TIMEOUT_RESTRICTION, 101
 - SAVAPI_E_HIT_MAX_COUNT, 92
 - SAVAPI_E_HIT_MAX_RATIO, 91
 - SAVAPI_E_HIT_MAX_REC, 91
 - SAVAPI_E_HIT_MAX_SIZE, 91
 - SAVAPI_E_HOSTNAME_NOT_SET, 89
 - SAVAPI_E_INCOMPLETE, 92
 - SAVAPI_E_INTERNAL, 90
 - SAVAPI_E_INVALID_PARAMETER, 88
 - SAVAPI_E_INVALID_QUERY, 95
 - SAVAPI_E_INVALID_VALUE, 93
 - SAVAPI_E_KEY_ACCESS_DENIED, 95
 - SAVAPI_E_KEY_DEMO_VERSION, 95
 - SAVAPI_E_KEY_EVAL_VERSION, 95
 - SAVAPI_E_KEY_EXPIRED, 95

- SAVAPI_E_KEY_FILE_INVALID, 95
- SAVAPI_E_KEY_ILLEGAL_LICENSE, 95
- SAVAPI_E_KEY_INVALID_HEADER, 95
- SAVAPI_E_KEY_KEYFILE_VERSION, 95
- SAVAPI_E_KEY_NO_KEYFILE, 95
- SAVAPI_E_KEY_NO_LICENSE, 95
- SAVAPI_E_KEY_READ, 96
- SAVAPI_E_KEY_RECORD_INVALID, 95
- SAVAPI_E_KEYFILE, 90
- SAVAPI_E_LICENSE_RESTRICTION, 96
- SAVAPI_E_LOADING_ENGINE_MODULES, 96
- SAVAPI_E_MATCHED, 94
- SAVAPI_E_MEMORY_LIMIT, 96
- SAVAPI_E_NO_MEMORY, 89
- SAVAPI_E_NON_ADDRESSABLE, 96
- SAVAPI_E_NOT_ABSOLUTE_PATH, 93
- SAVAPI_E_NOT_INITIALIZED, 89
- SAVAPI_E_NOT_SUPPORTED, 91
- SAVAPI_E_OA_DRIVERS, 101
- SAVAPI_E_OA_ERROR, 100
- SAVAPI_E_OA_INITIALIZED, 100
- SAVAPI_E_OA_INSTANCES_CREATED, 100
- SAVAPI_E_OA_NO_INSTANCES_CREATED, 100
- SAVAPI_E_OA_NO_LICENSE, 100
- SAVAPI_E_OA_NO_PRIVILEGES, 101
- SAVAPI_E_OA_NO_SCAN_IN_PROGRESS, 100
- SAVAPI_E_OA_NOT_INITIALIZED, 100
- SAVAPI_E_OA_SCAN_IN_PROGRESS, 100
- SAVAPI_E_OPTION_NOT_SUPPORTED, 91
- SAVAPI_E_OPTION_VALUE_INVALID, 94
- SAVAPI_E_PARTIAL, 92
- SAVAPI_E_PREFIX_GET, 95
- SAVAPI_E_PREFIX_SET, 95
- SAVAPI_E_PROC_ARCHIVE_HANDLE, 97
- SAVAPI_E_PROC_BAD_FORMAT, 97
- SAVAPI_E_PROC_BAD_HEADER, 96
- SAVAPI_E_PROC_BAD_TABLE, 97
- SAVAPI_E_PROC_CALLBACK, 98
- SAVAPI_E_PROC_DATA_CRC, 97
- SAVAPI_E_PROC_ERROR, 92
- SAVAPI_E_PROC_FILE_CRC, 97
- SAVAPI_E_PROC_HEADER_CRC, 97
- SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ, 96
- SAVAPI_E_PROC_INVALID_COMPRESSED_DATA, 97
- SAVAPI_E_PROC_NO_FILES_TO_EXTRACT, 98
- SAVAPI_E_PROC_OBSOLETE, 97
- SAVAPI_E_PROC_TOTAL_LOSS, 98
- SAVAPI_E_PROC_UNEXPECTED_EOF, 97
- SAVAPI_E_RECEIVE_FAILED, 94
- SAVAPI_E_REPAIR_FAILED, 94
- SAVAPI_E_RESULT_FILE_NOT_FOUND, 91
- SAVAPI_E_SCAN_ERROR, 101
- SAVAPI_E_SELFCHK_FILE_CRC, 90
- SAVAPI_E_SELFCHK_FILE_ERR, 90
- SAVAPI_E_SELFCHK_PATCHED, 90
- SAVAPI_E_SEND_FAILED, 94
- SAVAPI_E_TIMEOUT, 92
- SAVAPI_E_UNKNOWN, 95
- SAVAPI_E_UNSUPPORTED, 92
- SAVAPI_E_UNSUPPORTED_COMPRESSION, 100
- SAVAPI_E_VDF_CRC, 89
- SAVAPI_E_VDF_NOT_FOUND, 89
- SAVAPI_E_VDF_READ, 89
- SAVAPI_E_VDF_VERSION, 90
- SAVAPI_E_WRONG_ENGINE, 90
- SAVAPI_S_INFECTED, 101
- SAVAPI_S_OK, 88
- SAVAPI_S_SUSPICIOUS, 101
- SAVAPI_STATUS, 88
- SAVAPI_status, 88
- SAVAPI scan statuses, 44
 - SAVAPI_SCAN_STATUS_CLEAN, 44
 - SAVAPI_SCAN_STATUS_ERROR, 44
 - SAVAPI_SCAN_STATUS_FINISHED, 44
 - SAVAPI_SCAN_STATUS_INFECTED, 45
 - SAVAPI_SCAN_STATUS_SUSPICIOUS, 45
- SAVAPI signals, 43
 - SAVAPI_SIGNAL_SCAN_ABORT, 43
- SAVAPI STCHAR function pointers, 108
 - CharToSTCHAR_t, 108
 - STCHARToChar_t, 108
- SAVAPI structures, 47
 - _SAVAPI_log_level, 55
 - SAVAPI_ALERT_URL_DATA, 49
 - SAVAPI_APC_ANSWER_CLEAN, 56
 - SAVAPI_APC_ANSWER_INFECTED, 56
 - SAVAPI_APC_GLOBAL_INIT, 50
 - SAVAPI_APC_SCAN_ANSWER, 55
 - SAVAPI_APC_SCAN_DATA, 50
 - SAVAPI_APC_SCAN_MODE, 50
 - SAVAPI_APC_scan_mode, 56
 - SAVAPI_APC_SCAN_MODE_CHECK_ONLY, 56
 - SAVAPI_APC_SCAN_MODE_FULL, 56
 - SAVAPI_APC_SCAN_STAGE, 56
 - SAVAPI_APC_STAGE_POST_SCAN, 56
 - SAVAPI_APC_STAGE_PRE_FILTER, 56
 - SAVAPI_APC_STAGE_PRE_HASH_CHECK, 56
 - SAVAPI_APC_STAGE_PRE_UPLOAD, 56
 - SAVAPI_ARCHIVE_OPEN_DATA, 50
 - SAVAPI_CALLBACK_DATA, 50
 - SAVAPI_COMMAND_DATA, 50
 - SAVAPI_ENGINE_MODULE_AVE, 57
 - SAVAPI_ENGINE_MODULE_TYPE, 51
 - SAVAPI_engine_module_type, 56
 - SAVAPI_ENGINE_MODULE_VDF, 57
 - SAVAPI_ERROR_DATA, 51
 - SAVAPI_FILE_INFO, 51
 - SAVAPI_FILE_STATUS_DATA, 51
 - SAVAPI_GLOBAL_INIT, 51
 - SAVAPI_IFRAME_URL_DATA, 52
 - SAVAPI_INSTANCE_INIT, 52
 - SAVAPI_KEY_VALUE, 52
 - SAVAPI_LOG_ALERT, 55

- SAVAPI_LOG_DEBUG, 55
- SAVAPI_LOG_ERROR, 55
- SAVAPI_LOG_INFO, 55
- SAVAPI_LOG_LEVEL, 52
- SAVAPI_LOG_WARNING, 55
- SAVAPI_MALWARE_INFO, 52
- SAVAPI_OA_FILE_RESULT_DATA, 53
- SAVAPI_OA_GLOBAL_INIT, 53
- SAVAPI_PRESCAN_DATA, 53
- SAVAPI_REPAIRABLE_DATA, 53
- SAVAPI_REPORT_CONTENT_DATA, 53
- SAVAPI_REPORT_PROGRESS_DATA, 54
- SAVAPI_REPORT_SCAN_DETAILS_DATA, 54
- SAVAPI_SIGNAL_DATA, 54
- SAVAPI_SIMPLE_SCAN_FILE_DATA, 54
- SAVAPI_SIMPLE_SCAN_OUTPUT, 54
- SAVAPI_SIMPLE_SCAN_STATISTICS, 55
- SAVAPI_VERSION, 55
- SAVAPI typedefs, 57
 - APC_set_report_info_t, 57
 - SAVAPI_CALLBACK, 58
 - SAVAPI_ENGINE_MODULE_CALLBACK, 58
 - SAVAPI_FD, 58
 - SAVAPI_LOG_CALLBACK, 58
 - SAVAPI_OA_INSTANCE_CALLBACK, 59
- savapi.h, 175
- SAVAPI_ALERT_URL_DATA
 - SAVAPI structures, 49
- SAVAPI_alert_url_data, 124
 - alert_url, 124
 - file_info, 124
- SAVAPI_APC_ANSWER_CLEAN
 - SAVAPI structures, 56
- SAVAPI_APC_ANSWER_INFECTED
 - SAVAPI structures, 56
- SAVAPI_APC_get_version
 - SAVAPI functions, 68
- SAVAPI_APC_get_version_t
 - SAVAPI main function pointers, 61
- SAVAPI_APC_GLOBAL_INIT
 - SAVAPI structures, 50
- SAVAPI_APC_global_init, 125
 - apc_mode, 125
 - blackout_retries, 125
 - blackout_timeout, 125
 - cache_file_path, 126
 - cache_size, 126
 - cert_dir, 126
 - dump_cache_file, 126
 - lib_dir, 127
 - proxy, 127
 - temp_dir, 127
- SAVAPI_APC_initialize
 - SAVAPI functions, 68
- SAVAPI_APC_initialize_t
 - SAVAPI main function pointers, 61
- SAVAPI_APC_REPORT_DATA, 128
 - malware_info, 128
 - scan_answer, 128
 - store_cache, 128
 - tll, 128
- SAVAPI_APC_SCAN_ANSWER
 - SAVAPI structures, 55
- SAVAPI_APC_SCAN_CONTINUE
 - SAVAPI APC scan callback return values, 39
- SAVAPI_APC_SCAN_DATA
 - SAVAPI structures, 50
- SAVAPI_apc_scan_data, 129
 - file_info, 129
 - flags, 129
 - fops_context, 130
 - fops_handle, 130
 - hash, 130
 - risk_rating_level, 130
 - savapi_fd, 130
 - set_report_info, 131
 - stage, 131
- SAVAPI_APC_SCAN_MODE
 - SAVAPI structures, 50
- SAVAPI_APC_scan_mode
 - SAVAPI structures, 56
- SAVAPI_APC_SCAN_MODE_CHECK_ONLY
 - SAVAPI structures, 56
- SAVAPI_APC_SCAN_MODE_FULL
 - SAVAPI structures, 56
- SAVAPI_APC_SCAN_REPORT
 - SAVAPI APC scan callback return values, 39
- SAVAPI_APC_SCAN_RESULT
 - SAVAPI APC scan callback return values, 39
- SAVAPI_APC_SCAN_STAGE
 - SAVAPI structures, 56
- SAVAPI_APC_SCAN_STOP
 - SAVAPI APC scan callback return values, 39
- SAVAPI_APC_STAGE_POST_SCAN
 - SAVAPI structures, 56
- SAVAPI_APC_STAGE_PRE_FILTER
 - SAVAPI structures, 56
- SAVAPI_APC_STAGE_PRE_HASH_CHECK
 - SAVAPI structures, 56
- SAVAPI_APC_STAGE_PRE_UPLOAD
 - SAVAPI structures, 56
- SAVAPI_APC_uninitialize
 - SAVAPI functions, 69
- SAVAPI_APC_uninitialize_t
 - SAVAPI main function pointers, 61
- SAVAPI_API_MAJOR_VERSION
 - API version, 13
- SAVAPI_API_MINOR_VERSION
 - API version, 13
- SAVAPI_ARCHIVE_OPEN_DATA
 - SAVAPI structures, 50
- SAVAPI_archive_open_data, 131
 - file_info, 132
 - flags, 132
- SAVAPI_CALLBACK
 - SAVAPI typedefs, 58

- SAVAPI_CALLBACK_APC_SCAN
 - Callbacks' ids, [40](#)
- SAVAPI_CALLBACK_ARCHIVE_OPEN
 - Callbacks' ids, [40](#)
- SAVAPI_CALLBACK_CONTENT_REPORT
 - Callbacks' ids, [41](#)
- SAVAPI_CALLBACK_DATA
 - SAVAPI structures, [50](#)
- SAVAPI_callback_data, [132](#)
 - callback_data, [133](#)
 - flags, [133](#)
 - type, [133](#)
 - user_data, [133](#)
 - version, [133](#)
- SAVAPI_callback_data::specific_data, [160](#)
 - apc_scan_data, [160](#)
 - archive_open_data, [160](#)
 - error_data, [160](#)
 - file_status_data, [160](#)
 - oa_file_result_data, [161](#)
 - pre_scan_data, [161](#)
 - private_data, [161](#)
 - report_content_data, [161](#)
 - report_progress_data, [161](#)
 - report_scan_details_data, [161](#)
- SAVAPI_CALLBACK_OA_FILE_RESULT
 - Callbacks' ids, [41](#)
- SAVAPI_CALLBACK_PRE_SCAN
 - Callbacks' ids, [41](#)
- SAVAPI_CALLBACK_PROGRESS_REPORT
 - Callbacks' ids, [41](#)
- SAVAPI_CALLBACK_REPORT_ERROR
 - Callbacks' ids, [42](#)
- SAVAPI_CALLBACK_REPORT_FILE_STATUS
 - Callbacks' ids, [42](#)
- SAVAPI_CALLBACK_SCAN_DETAILS_REPORT
 - Callbacks' ids, [42](#)
- SAVAPI_COMMAND_DATA
 - SAVAPI structures, [50](#)
- SAVAPI_command_data, [134](#)
 - command_data, [134](#)
 - signal_id, [134](#)
- SAVAPI_create_instance
 - SAVAPI functions, [69](#)
- SAVAPI_create_instance_t
 - SAVAPI main function pointers, [61](#)
- SAVAPI_E_ABORTED
 - SAVAPI return codes, [92](#)
- SAVAPI_E_ALREADY_INITIALIZED
 - SAVAPI return codes, [88](#)
- SAVAPI_E_APC_ALREADY_INITIALIZED
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_AUTH_RETRY_LATER
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_AUTHENTICATION
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_CONNECTION
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_DISABLED
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_ERROR
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_INCOMPLETE
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_NO_LICENSE
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_NOT_INITIALIZED
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_NOT_SUPPORTED
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_QUOTA
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_RANDOM_ID
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_TEMPORARILY_DISABLED
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_TIMEOUT
 - SAVAPI return codes, [98](#)
- SAVAPI_E_APC_TIMEOUT_RESTRICTION
 - SAVAPI return codes, [99](#)
- SAVAPI_E_APC_UNKNOWN_CATEGORY
 - SAVAPI return codes, [99](#)
- SAVAPI_E_BUFFER_TOO_SMALL
 - SAVAPI return codes, [89](#)
- SAVAPI_E_BUSY
 - SAVAPI return codes, [96](#)
- SAVAPI_E_CHDIR_FAILED
 - SAVAPI return codes, [93](#)
- SAVAPI_E_CONNECTION_FAILED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_CONNECTION_MODE_NOT_SET
 - SAVAPI return codes, [89](#)
- SAVAPI_E_CONVERSION_FAILED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_DIR_NOT_EXISTS
 - SAVAPI return codes, [93](#)
- SAVAPI_E_ENCRYPTED
 - SAVAPI return codes, [92](#)
- SAVAPI_E_ENCRYPTED_MIME
 - SAVAPI return codes, [96](#)
- SAVAPI_E_ENGINE_NOT_FOUND
 - SAVAPI return codes, [90](#)
- SAVAPI_E_FILE_CLOSE
 - SAVAPI return codes, [95](#)
- SAVAPI_E_FILE_CREATE
 - SAVAPI return codes, [94](#)
- SAVAPI_E_FILE_DELETE
 - SAVAPI return codes, [95](#)
- SAVAPI_E_FILE_OPEN
 - SAVAPI return codes, [93](#)
- SAVAPI_E_FILE_READ
 - SAVAPI return codes, [93](#)
- SAVAPI_E_FILE_WRITE
 - SAVAPI return codes, [93](#)
- SAVAPI_E_FPC_TIMEOUT_RESTRICTION
 - SAVAPI return codes, [101](#)

- SAVAPI_E_HIT_MAX_COUNT
 - SAVAPI return codes, [92](#)
- SAVAPI_E_HIT_MAX_RATIO
 - SAVAPI return codes, [91](#)
- SAVAPI_E_HIT_MAX_REC
 - SAVAPI return codes, [91](#)
- SAVAPI_E_HIT_MAX_SIZE
 - SAVAPI return codes, [91](#)
- SAVAPI_E_HOSTNAME_NOT_SET
 - SAVAPI return codes, [89](#)
- SAVAPI_E_INCOMPLETE
 - SAVAPI return codes, [92](#)
- SAVAPI_E_INTERNAL
 - SAVAPI return codes, [90](#)
- SAVAPI_E_INVALID_PARAMETER
 - SAVAPI return codes, [88](#)
- SAVAPI_E_INVALID_QUERY
 - SAVAPI return codes, [95](#)
- SAVAPI_E_INVALID_VALUE
 - SAVAPI return codes, [93](#)
- SAVAPI_E_KEY_ACCESS_DENIED
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_DEMO_VERSION
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_EVAL_VERSION
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_EXPIRED
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_FILE_INVALID
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_ILLEGAL_LICENSE
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_INVALID_HEADER
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_KEYFILE_VERSION
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_NO_KEYFILE
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_NO_LICENSE
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEY_READ
 - SAVAPI return codes, [96](#)
- SAVAPI_E_KEY_RECORD_INVALID
 - SAVAPI return codes, [95](#)
- SAVAPI_E_KEYFILE
 - SAVAPI return codes, [90](#)
- SAVAPI_E_LICENSE_RESTRICTION
 - SAVAPI return codes, [96](#)
- SAVAPI_E_LOADING_ENGINE_MODULES
 - SAVAPI return codes, [96](#)
- SAVAPI_E_MATCHED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_MEMORY_LIMIT
 - SAVAPI return codes, [96](#)
- SAVAPI_E_NO_MEMORY
 - SAVAPI return codes, [89](#)
- SAVAPI_E_NON_ADDRESSABLE
 - SAVAPI return codes, [96](#)
- SAVAPI_E_NOT_ABSOLUTE_PATH
 - SAVAPI return codes, [93](#)
- SAVAPI_E_NOT_INITIALIZED
 - SAVAPI return codes, [89](#)
- SAVAPI_E_NOT_SUPPORTED
 - SAVAPI return codes, [91](#)
- SAVAPI_E_OA_DRIVERS
 - SAVAPI return codes, [101](#)
- SAVAPI_E_OA_ERROR
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_INITIALIZED
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_INSTANCES_CREATED
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_NO_INSTANCES_CREATED
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_NO_LICENSE
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_NO_PRIVILEGES
 - SAVAPI return codes, [101](#)
- SAVAPI_E_OA_NO_SCAN_IN_PROGRESS
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_NOT_INITIALIZED
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OA_SCAN_IN_PROGRESS
 - SAVAPI return codes, [100](#)
- SAVAPI_E_OPTION_NOT_SUPPORTED
 - SAVAPI return codes, [91](#)
- SAVAPI_E_OPTION_VALUE_INVALID
 - SAVAPI return codes, [94](#)
- SAVAPI_E_PARTIAL
 - SAVAPI return codes, [92](#)
- SAVAPI_E_PREFIX_GET
 - SAVAPI return codes, [95](#)
- SAVAPI_E_PREFIX_SET
 - SAVAPI return codes, [95](#)
- SAVAPI_E_PROC_ARCHIVE_HANDLE
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_BAD_FORMAT
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_BAD_HEADER
 - SAVAPI return codes, [96](#)
- SAVAPI_E_PROC_BAD_TABLE
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_CALLBACK
 - SAVAPI return codes, [98](#)
- SAVAPI_E_PROC_DATA_CRC
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_ERROR
 - SAVAPI return codes, [92](#)
- SAVAPI_E_PROC_FILE_CRC
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_HEADER_CRC
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_INCOMPLETE_BLOCK_READ
 - SAVAPI return codes, [96](#)
- SAVAPI_E_PROC_INVALID_COMPRESSED_DATA
 - SAVAPI return codes, [97](#)

- SAVAPI_E_PROC_NO_FILES_TO_EXTRACT
 - SAVAPI return codes, [98](#)
- SAVAPI_E_PROC_OBSOLETE
 - SAVAPI return codes, [97](#)
- SAVAPI_E_PROC_TOTAL_LOSS
 - SAVAPI return codes, [98](#)
- SAVAPI_E_PROC_UNEXPECTED_EOF
 - SAVAPI return codes, [97](#)
- SAVAPI_E_RECEIVE_FAILED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_REPAIR_FAILED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_RESULT_FILE_NOT_FOUND
 - SAVAPI return codes, [91](#)
- SAVAPI_E_SCAN_ERROR
 - SAVAPI return codes, [101](#)
- SAVAPI_E_SELFCHK_FILE_CRC
 - SAVAPI return codes, [90](#)
- SAVAPI_E_SELFCHK_FILE_ERR
 - SAVAPI return codes, [90](#)
- SAVAPI_E_SELFCHK_PATCHED
 - SAVAPI return codes, [90](#)
- SAVAPI_E_SEND_FAILED
 - SAVAPI return codes, [94](#)
- SAVAPI_E_TIMEOUT
 - SAVAPI return codes, [92](#)
- SAVAPI_E_UNKNOWN
 - SAVAPI return codes, [95](#)
- SAVAPI_E_UNSUPPORTED
 - SAVAPI return codes, [92](#)
- SAVAPI_E_UNSUPPORTED_COMPRESSION
 - SAVAPI return codes, [100](#)
- SAVAPI_E_VDF_CRC
 - SAVAPI return codes, [89](#)
- SAVAPI_E_VDF_NOT_FOUND
 - SAVAPI return codes, [89](#)
- SAVAPI_E_VDF_READ
 - SAVAPI return codes, [89](#)
- SAVAPI_E_VDF_VERSION
 - SAVAPI return codes, [90](#)
- SAVAPI_E_WRONG_ENGINE
 - SAVAPI return codes, [90](#)
- SAVAPI_ECAT_APC_REPORT_TTL
 - Error or information categories, [14](#)
- SAVAPI_ECAT_ERROR_GENERIC
 - Error or information categories, [14](#)
- SAVAPI_ECAT_ERROR_IO
 - Error or information categories, [14](#)
- SAVAPI_ECAT_ERROR_SCAN
 - Error or information categories, [14](#)
- SAVAPI_ECAT_ERROR_UNPACK
 - Error or information categories, [14](#)
- SAVAPI_ELEVEL_ERROR
 - Error levels, [15](#)
- SAVAPI_ELEVEL_INFO
 - Error levels, [15](#)
- SAVAPI_ELEVEL_WARNING
 - Error levels, [15](#)
- SAVAPI_ENGINE_MODULE_AVE
 - SAVAPI structures, [57](#)
- SAVAPI_ENGINE_MODULE_CALLBACK
 - SAVAPI typedefs, [58](#)
- SAVAPI_ENGINE_MODULE_TYPE
 - SAVAPI structures, [51](#)
- SAVAPI_engine_module_type
 - SAVAPI structures, [56](#)
- SAVAPI_ENGINE_MODULE_VDF
 - SAVAPI structures, [57](#)
- SAVAPI_engine_modules_get
 - SAVAPI functions, [70](#)
- SAVAPI_engine_modules_get_t
 - SAVAPI main function pointers, [61](#)
- SAVAPI_engine_versions_get
 - SAVAPI functions, [70](#)
- SAVAPI_engine_versions_get_t
 - SAVAPI main function pointers, [61](#)
- SAVAPI_EPLG_INIT
 - Plugin return statuses, [103](#)
- SAVAPI_EPLG_INTERNAL
 - Plugin return statuses, [103](#)
- SAVAPI_EPLG_INVALID
 - Plugin return statuses, [103](#)
- SAVAPI_EPLG_NO_MEM
 - Plugin return statuses, [103](#)
- SAVAPI_EPLG_SUCCESS
 - Plugin return statuses, [103](#)
- SAVAPI_EPLG_UKNW_INTERFACE
 - Plugin return statuses, [103](#)
- SAVAPI_ERROR_DATA
 - SAVAPI structures, [51](#)
- SAVAPI_error_data, [135](#)
 - category, [135](#)
 - code, [135](#)
 - file_info, [135](#)
 - level, [136](#)
 - options, [136](#)
- savapi_errors.h, [191](#)
- SAVAPI_EXP
 - stchar.h, [198](#)
- SAVAPI_extract_malware_names
 - SAVAPI functions, [71](#)
- SAVAPI_extract_malware_names_t
 - SAVAPI main function pointers, [61](#)
- SAVAPI_FD
 - SAVAPI typedefs, [58](#)
- savapi_fd
 - SAVAPI_apc_scan_data, [130](#)
- SAVAPI_FILE_INFO
 - SAVAPI structures, [51](#)
- SAVAPI_file_info, [136](#)
 - level, [136](#)
 - name, [137](#)
 - type, [137](#)
- SAVAPI_FILE_STATUS_DATA
 - SAVAPI structures, [51](#)
- SAVAPI_file_status_data, [137](#)

- file_info, 138
- flags, 138
- info, 138
- malware_info, 138
- scan_answer, 138
- warning, 138
- SAVAPI_FLAG_LAST_FILENAME_DEFAULT
 - Filename flags, 46
- SAVAPI_FLAG_USE_LOCAL_SOCKET
 - Initialization flags, 16
- SAVAPI_FLAG_USE_TCP
 - Initialization flags, 16
- SAVAPI_FPC_disable_preinit
 - SAVAPI functions, 71
- SAVAPI_free
 - SAVAPI functions, 72
- SAVAPI_free_t
 - SAVAPI main function pointers, 62
- SAVAPI_FTYPE_ARCHIVE
 - File types, 45
- SAVAPI_FTYPE_IN_ARCHIVE
 - File types, 45
- SAVAPI_FTYPE_REGULAR
 - File types, 46
- SAVAPI_get
 - SAVAPI functions, 72
- SAVAPI_get_dynamic_detect
 - SAVAPI functions, 73
- SAVAPI_get_fops
 - SAVAPI functions, 73
- SAVAPI_get_fops_t
 - SAVAPI main function pointers, 62
- SAVAPI_get_t
 - SAVAPI main function pointers, 62
- SAVAPI_get_user_data
 - SAVAPI functions, 74
- SAVAPI_get_user_data_t
 - SAVAPI main function pointers, 62
- SAVAPI_get_version
 - SAVAPI functions, 74
- SAVAPI_get_version_t
 - SAVAPI main function pointers, 62
- SAVAPI_GLOBAL_INIT
 - SAVAPI structures, 51
- SAVAPI_global_init, 139
 - api_major_version, 139
 - api_minor_version, 139
 - avll_dirpath, 139
 - engine_dirpath, 140
 - key_file_name, 140
 - program_type, 140
 - vdfs_dirpath, 140
- SAVAPI_GLOBAL_OPTION
 - SAVAPI global options, 33
- SAVAPI_global_option
 - SAVAPI global options, 33
- SAVAPI_global_set
 - SAVAPI functions, 74
- SAVAPI_global_set_t
 - SAVAPI main function pointers, 62
- SAVAPI_HTML_CONTENT_ATTRIB_EXTRASMALL
 - Iframes informations, 18
- SAVAPI_HTML_CONTENT_ATTRIB_INVISIBLE
 - Iframes informations, 18
- SAVAPI_HTML_CONTENT_ATTRIB_MALICIOUS
 - Iframes informations, 18
- SAVAPI_HTML_CONTENT_ATTRIB_ODDPOS
 - Iframes informations, 18
- SAVAPI_I_ACTIVE_CONTENT_PRESENT
 - Scan information, 19
- SAVAPI_I_ALL_MACROS_DELETED
 - Scan information, 19
- SAVAPI_I_APC_RESULT_EXPIRY
 - Scan information, 19
- SAVAPI_I_APC_SCAN_DURATION
 - Scan information, 19
- SAVAPI_I_MACRO_AUTOSTART
 - Scan information, 20
- SAVAPI_I_MACROS_PRESENT
 - Scan information, 20
- SAVAPI_I_MAILBOX
 - Scan information, 20
- SAVAPI_I_OLE_ENCRYPTED
 - Scan information, 20
- SAVAPI_I_OLEFILE
 - Scan information, 20
- SAVAPI_I_TEMPLATE
 - Scan information, 20
- SAVAPI_IFRAME_URL_DATA
 - SAVAPI structures, 52
- SAVAPI_iframe_url_data, 141
 - attribute, 141
 - url, 141
- SAVAPI_initialize
 - SAVAPI functions, 75
- SAVAPI_initialize_t
 - SAVAPI main function pointers, 62
- SAVAPI_INSTANCE_INIT
 - SAVAPI structures, 52
- SAVAPI_instance_init, 141
 - connection_timeout, 142
 - flags, 142
 - get_timeout, 142
 - host_name, 142
 - password, 143
 - port, 143
 - scan_timeout, 143
 - set_timeout, 143
 - username, 144
- SAVAPI_is_running
 - SAVAPI functions, 75
- SAVAPI_is_running_ex
 - SAVAPI functions, 76
- SAVAPI_is_running_ex_t
 - SAVAPI main function pointers, 63
- SAVAPI_KEY_VALUE

- SAVAPI structures, 52
- SAVAPI_key_value, 144
 - id, 145
 - type, 145
 - value, 145
- SAVAPI_LOG_ALERT
 - SAVAPI structures, 55
- SAVAPI_LOG_CALLBACK
 - SAVAPI typedefs, 58
- SAVAPI_LOG_DEBUG
 - SAVAPI structures, 55
- SAVAPI_LOG_ERROR
 - SAVAPI structures, 55
- SAVAPI_LOG_INFO
 - SAVAPI structures, 55
- SAVAPI_LOG_LEVEL
 - SAVAPI structures, 52
- SAVAPI_LOG_WARNING
 - SAVAPI structures, 55
- SAVAPI_MALWARE_INFO
 - SAVAPI structures, 52
- SAVAPI_malware_info, 145
 - app_flags, 146
 - message, 146
 - name, 146
 - removable, 146
 - strict, 146
 - type, 146
- SAVAPI_OA_create_instances
 - SAVAPI functions, 76
- SAVAPI_OA_create_instances_t
 - SAVAPI main function pointers, 63
- SAVAPI_OA_FILE_RESULT_DATA
 - SAVAPI structures, 53
- SAVAPI_OA_file_result_data, 147
 - filename, 147
 - pid, 147
 - sid, 147
 - sid_len, 147
- SAVAPI_OA_GLOBAL_INIT
 - SAVAPI structures, 53
- SAVAPI_OA_global_init, 148
 - scm_pending_time, 148
 - threads_number, 148
- SAVAPI_OA_initialize
 - SAVAPI functions, 77
- SAVAPI_OA_initialize_t
 - SAVAPI main function pointers, 63
- SAVAPI_OA_INSTANCE_CALLBACK
 - SAVAPI typedefs, 59
- SAVAPI_OA_RESULT_ALLOW
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_RESULT_DELETE
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_RESULT_DENY
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_RESULT_RENAME
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_RESULT_WIPE
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_SCAN_RESULT
 - SAVAPI OnAccess result, 38
- SAVAPI_OA_start_scan
 - SAVAPI functions, 77
- SAVAPI_OA_start_scan_t
 - SAVAPI main function pointers, 63
- SAVAPI_OA_stop_scan
 - SAVAPI functions, 78
- SAVAPI_OA_stop_scan_t
 - SAVAPI main function pointers, 63
- SAVAPI_OA_uninitialize
 - SAVAPI functions, 78
- SAVAPI_OA_uninitialize_t
 - SAVAPI main function pointers, 63
- SAVAPI_OPTION
 - SAVAPI options, 22
- SAVAPI_option
 - SAVAPI options, 22
- SAVAPI_OPTION_APC_CHECK_RISK_RATING_LEVEL
 - SAVAPI options, 27
- SAVAPI_OPTION_APC_CONNECTION_TIMEOUT
 - SAVAPI options, 27
- SAVAPI_OPTION_APC_ELF_MODE
 - SAVAPI options, 31
- SAVAPI_OPTION_APC_FILE_EXTENSIONS_CHECK_ONLY
 - SAVAPI options, 30
- SAVAPI_OPTION_APC_FILE_EXTENSIONS_DISABLED
 - SAVAPI options, 30
- SAVAPI_OPTION_APC_FILE_EXTENSIONS_FULL
 - SAVAPI options, 30
- SAVAPI_OPTION_APC_FILE_EXTENSIONS_POLICY
 - SAVAPI options, 29
- SAVAPI_OPTION_APC_MACH_O_MODE
 - SAVAPI options, 31
- SAVAPI_OPTION_APC_PE_MODE
 - SAVAPI options, 29
- SAVAPI_OPTION_APC_REPORT_SCAN_TTL
 - SAVAPI options, 28
- SAVAPI_OPTION_APC_SCAN_TIMEOUT
 - SAVAPI options, 27
- SAVAPI_OPTION_APC_UPLOAD_RISK_RATING_LEVEL
 - SAVAPI options, 28
- SAVAPI_OPTION_ARCHIVE_MAX_COUNT
 - SAVAPI options, 23
- SAVAPI_OPTION_ARCHIVE_MAX_RATIO
 - SAVAPI options, 23
- SAVAPI_OPTION_ARCHIVE_MAX_REC
 - SAVAPI options, 23
- SAVAPI_OPTION_ARCHIVE_MAX_SIZE
 - SAVAPI options, 23
- SAVAPI_OPTION_ARCHIVE_SCAN
 - SAVAPI options, 22
- SAVAPI_OPTION_AVE_VERSION
 - SAVAPI options, 32
- SAVAPI_OPTION_CONF
 - SAVAPI options, 22

- SAVAPI_OPTION_CWD
 - SAVAPI options, [22](#)
- SAVAPI_OPTION_DESCR_ADSPY
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_ADWARE
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_APPL
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_BDC
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_DIAL
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_GAME
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_HIDDENEXT
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_JOKE
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_PCK
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_PFS
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_PHISH
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_PUA
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DESCR_SPR
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_DETECT_ADSPY
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_ADWARE
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_DETECT_ALLTYPES
 - SAVAPI options, [31](#)
- SAVAPI_OPTION_DETECT_APPL
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_BDC
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_DIAL
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_GAME
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_HIDDENEXT
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_JOKE
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_PCK
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_PFS
 - SAVAPI options, [27](#)
- SAVAPI_OPTION_DETECT_PHISH
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_DETECT_PUA
 - SAVAPI options, [28](#)
- SAVAPI_OPTION_DETECT_SPR
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_EXPIRE
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_FPC
 - SAVAPI options, [28](#)
- SAVAPI_OPTION_FPC_TIMEOUT
 - SAVAPI options, [29](#)
- SAVAPI_OPTION_G_FPC_BLACKOUT_RETRIES
 - SAVAPI global options, [37](#)
- SAVAPI_OPTION_G_FPC_BLACKOUT_TIMEOUT
 - SAVAPI global options, [37](#)
- SAVAPI_OPTION_G_OA_CACHE_SCAN_NETWORK_DRIVES
 - SAVAPI global options, [37](#)
- SAVAPI_OPTION_G_OA_EXCEPTED_FILES
 - SAVAPI global options, [35](#)
- SAVAPI_OPTION_G_OA_EXCEPTED_PROCESSES
 - SAVAPI global options, [36](#)
- SAVAPI_OPTION_G_OA_EXTENSIONS_LIST
 - SAVAPI global options, [34](#)
- SAVAPI_OPTION_G_OA_MALWARE_RESPONSE_TTL
 - SAVAPI global options, [37](#)
- SAVAPI_OPTION_G_OA_SCAN_AT_FILE_CHANGES
 - SAVAPI global options, [36](#)
- SAVAPI_OPTION_G_OA_SCAN_NETWORK_DRIVES
 - SAVAPI global options, [36](#)
- SAVAPI_OPTION_G_OA_SCAN_TIMEOUT
 - SAVAPI global options, [37](#)
- SAVAPI_OPTION_G_PROXY
 - SAVAPI global options, [38](#)
- SAVAPI_OPTION_HEUR_LEVEL
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_HEUR_MACRO
 - SAVAPI options, [23](#)
- SAVAPI_OPTION_IFRAMES_URL
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_MAILBOX_SCAN
 - SAVAPI options, [23](#)
- SAVAPI_OPTION_MALWARE_NAMES_FILE
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_MIME_SCAN
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_NOTIFY_ALERTURL
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_NOTIFY_OFFICE
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_NOTIFY_OFFICE_MACRO
 - SAVAPI options, [25](#)
- SAVAPI_OPTION_NOTIFY_OFFICE_MACRO_AUTOSTART
 - SAVAPI options, [28](#)
- SAVAPI_OPTION_NOTIFY_REPAIR
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_PGP_SCAN
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_PID
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_PRODUCT
 - SAVAPI options, [31](#)
- SAVAPI_OPTION_REPAIR
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_REPORT_ENCRYPTED_MIME
 - SAVAPI options, [26](#)

- SAVAPI_OPTION_RESERVED1
 - SAVAPI options, [27](#)
- SAVAPI_OPTION_RESERVED2
 - SAVAPI options, [27](#)
- SAVAPI_OPTION_SAVAPI
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_SCAN_MODE
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_SCAN_PROGRESS
 - SAVAPI options, [26](#)
- SAVAPI_OPTION_SCAN_TEMP
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_SCAN_TIMEOUT
 - SAVAPI options, [24](#)
- SAVAPI_OPTION_SCAN_TIMEOUTS
 - SAVAPI options, [31](#)
- SAVAPI_OPTION_SELECTABLE_DETECT
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_VDF_DATE
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_VDF_VERSION
 - SAVAPI options, [32](#)
- SAVAPI_OPTION_VDFSIGCOUNT
 - SAVAPI options, [32](#)
- savapi_plg.h, [194](#)
- SAVAPI_PLG_AVE_FOPS
 - types for a SAVAPI FOPS plugin, [107](#)
- SAVAPI_PLG_EXP
 - Plugin defines, [101](#)
- savapi_plg_fops.h, [196](#)
- SAVAPI_PLG_fops_inst_options_t
 - types for a SAVAPI FOPS plugin, [108](#)
- SAVAPI_PLG_fops_instance_t
 - types for a SAVAPI FOPS plugin, [108](#)
- SAVAPI_PLG_fops_options_t
 - types for a SAVAPI FOPS plugin, [108](#)
- SAVAPI_PLG_info_t
 - Plugin's structures definitions, [106](#)
- SAVAPI_PLG_init_t
 - Plugin's general typedefs, [105](#)
- SAVAPI_PLG_inst_options_t
 - Plugin's specific typedefs, [104](#)
- SAVAPI_PLG_instance_create_t
 - Plugin's general typedefs, [105](#)
- SAVAPI_PLG_instance_release_t
 - Plugin's general typedefs, [105](#)
- SAVAPI_PLG_instance_t
 - Plugin's specific typedefs, [104](#)
- SAVAPI_PLG_MAIN_FUNC
 - Plugin's structures definitions, [106](#), [107](#)
- SAVAPI_PLG_main_t
 - Plugin's structures definitions, [106](#)
- SAVAPI_PLG_MAX_STR
 - Plugin defines, [102](#)
- SAVAPI_PLG_MAX_VER_STR
 - Plugin defines, [102](#)
- SAVAPI_PLG_options_t
 - Plugin's specific typedefs, [104](#)
- SAVAPI_PLG_status_t
 - Plugin's general typedefs, [105](#)
- SAVAPI_PLG_tchar_t
 - Plugin defines, [102](#)
- SAVAPI_PLG_uninit_t
 - Plugin's general typedefs, [105](#)
- SAVAPI_PLG_VER_MAJ
 - Plugin defines, [102](#)
- SAVAPI_PLG_VER_MIN
 - Plugin defines, [102](#)
- SAVAPI_pre_scan_data, [149](#)
 - file_info, [149](#)
 - flags, [149](#)
- SAVAPI_PRESCAN_DATA
 - SAVAPI structures, [53](#)
- SAVAPI_register_callback
 - SAVAPI functions, [78](#)
- SAVAPI_register_callback_t
 - SAVAPI main function pointers, [63](#)
- SAVAPI_release_instance
 - SAVAPI functions, [79](#)
- SAVAPI_release_instance_t
 - SAVAPI main function pointers, [64](#)
- SAVAPI_reload_engine
 - SAVAPI functions, [79](#)
- SAVAPI_reload_engine_ex
 - SAVAPI functions, [80](#)
- SAVAPI_reload_engine_ex_t
 - SAVAPI main function pointers, [64](#)
- SAVAPI_REPAIRABLE_DATA
 - SAVAPI structures, [53](#)
- SAVAPI_repairable_data, [150](#)
 - file_info, [150](#)
 - malware_info, [150](#)
- SAVAPI_REPORT_ALERTURL
 - SAVAPI report scan details types, [42](#)
- SAVAPI_REPORT_CONTENT_DATA
 - SAVAPI structures, [53](#)
- SAVAPI_report_content_data, [151](#)
 - content_data, [151](#)
 - file_info, [151](#)
 - flags, [151](#)
 - type, [151](#)
- SAVAPI_report_content_data::_content_data, [120](#)
 - iframeurl_data, [120](#)
- SAVAPI_REPORT_CONTENT_IFRAME
 - SAVAPI report content types, [43](#)
- SAVAPI_REPORT_PROGRESS_DATA
 - SAVAPI structures, [54](#)
- SAVAPI_report_progress_data, [152](#)
 - flags, [152](#)
 - message, [152](#)
- SAVAPI_REPORT_REPAIRABLE
 - SAVAPI report scan details types, [43](#)
- SAVAPI_REPORT_SCAN_DETAILS_DATA
 - SAVAPI structures, [54](#)
- SAVAPI_report_scan_details_data, [153](#)
 - flags, [153](#)

- scan_details_data, 153
- type, 153
- SAVAPI_report_scan_details_data::_scan_details_data, 123
- alert_url_data, 123
- repairable_data, 123
- SAVAPI_S_INFECTED
 - SAVAPI return codes, 101
- SAVAPI_S_OK
 - SAVAPI return codes, 88
- SAVAPI_S_SUSPICIOUS
 - SAVAPI return codes, 101
- SAVAPI_scan
 - SAVAPI functions, 81
- SAVAPI_SCAN_STATUS_CLEAN
 - SAVAPI scan statuses, 44
- SAVAPI_SCAN_STATUS_ERROR
 - SAVAPI scan statuses, 44
- SAVAPI_SCAN_STATUS_FINISHED
 - SAVAPI scan statuses, 44
- SAVAPI_SCAN_STATUS_INFECTED
 - SAVAPI scan statuses, 45
- SAVAPI_SCAN_STATUS_SUSPICIOUS
 - SAVAPI scan statuses, 45
- SAVAPI_scan_t
 - SAVAPI main function pointers, 64
- SAVAPI_send_signal
 - SAVAPI functions, 81
- SAVAPI_send_signal_t
 - SAVAPI main function pointers, 64
- SAVAPI_set
 - SAVAPI functions, 82
- SAVAPI_set_fops
 - SAVAPI functions, 83
- SAVAPI_set_fops_t
 - SAVAPI main function pointers, 64
- SAVAPI_set_log_callback
 - SAVAPI functions, 83
- SAVAPI_set_log_callback_t
 - SAVAPI main function pointers, 64
- SAVAPI_set_quickload_init
 - SAVAPI functions, 84
- SAVAPI_set_quickload_init_t
 - SAVAPI main function pointers, 64
- SAVAPI_set_t
 - SAVAPI main function pointers, 65
- SAVAPI_set_user_data
 - SAVAPI functions, 84
- SAVAPI_set_user_data_t
 - SAVAPI main function pointers, 65
- SAVAPI_SIGNAL_DATA
 - SAVAPI structures, 54
- SAVAPI_signal_data, 154
 - signal_data, 154
 - signal_id, 154
- SAVAPI_SIGNAL_SCAN_ABORT
 - SAVAPI signals, 43
- SAVAPI_simple_scan
 - SAVAPI functions, 85
- SAVAPI_SIMPLE_SCAN_FILE_DATA
 - SAVAPI structures, 54
- SAVAPI_simple_scan_file_data, 155
 - error_code, 155
 - error_level, 155
 - malware_name, 155
 - malware_type, 156
 - name, 156
 - scan_answer, 156
- SAVAPI_SIMPLE_SCAN_OUTPUT
 - SAVAPI structures, 54
- SAVAPI_simple_scan_output, 156
 - count, 157
 - files, 157
 - stats, 157
- SAVAPI_SIMPLE_SCAN_STATISTICS
 - SAVAPI structures, 55
- SAVAPI_simple_scan_statistics, 157
 - errors, 158
 - infections, 158
 - suspicious, 158
 - total_files, 158
- SAVAPI_simple_scan_t
 - SAVAPI main function pointers, 65
- SAVAPI_SIZE_T
 - stchar.h, 198
- SAVAPI_STATUS
 - SAVAPI return codes, 88
- SAVAPI_status
 - SAVAPI return codes, 88
- SAVAPI_SYMBOL
 - SAVAPI constants, 12
- SAVAPI_TCHAR
 - stchar.h, 198
- SAVAPI_uninitialize
 - SAVAPI functions, 85
- SAVAPI_uninitialize_t
 - SAVAPI main function pointers, 65
- SAVAPI_unregister_callback
 - SAVAPI functions, 86
- SAVAPI_unregister_callback_t
 - SAVAPI main function pointers, 65
- SAVAPI_VERSION
 - SAVAPI structures, 55
- SAVAPI_version, 158
 - build_major, 159
 - build_minor, 159
 - major, 159
 - minor, 159
- SAVAPI_W_DAMAGED
 - Scan warnings, 17
- SAVAPI_W_HEADER_MALFORMED
 - Scan warnings, 17
- SAVAPI_W_MAX_EXTRACTED
 - Scan warnings, 17
- SAVAPI_W_OLE_DAMAGED
 - Scan warnings, 17

- SAVAPI_W_POTENTIAL_ARCH_BOMB
 - Scan warnings, [17](#)
- SAVAPI_W_PROGRESS_ABORT
 - Scan warnings, [17](#)
- SAVAPI_W_RATIO_EXCEEDED
 - Scan warnings, [17](#)
- SAVAPI_W_SUSPICIOUS
 - Scan warnings, [17](#)
- Scan information, [19](#)
 - SAVAPI_I_ACTIVE_CONTENT_PRESENT, [19](#)
 - SAVAPI_I_ALL_MACROS_DELETED, [19](#)
 - SAVAPI_I_APC_RESULT_EXPIRY, [19](#)
 - SAVAPI_I_APC_SCAN_DURATION, [19](#)
 - SAVAPI_I_MACRO_AUTOSTART, [20](#)
 - SAVAPI_I_MACROS_PRESENT, [20](#)
 - SAVAPI_I_MAILBOX, [20](#)
 - SAVAPI_I_OLE_ENCRYPTED, [20](#)
 - SAVAPI_I_OLEFILE, [20](#)
 - SAVAPI_I_TEMPLATE, [20](#)
- Scan warnings, [16](#)
 - SAVAPI_W_DAMAGED, [17](#)
 - SAVAPI_W_HEADER_MALFORMED, [17](#)
 - SAVAPI_W_MAX_EXTRACTED, [17](#)
 - SAVAPI_W_OLE_DAMAGED, [17](#)
 - SAVAPI_W_POTENTIAL_ARCH_BOMB, [17](#)
 - SAVAPI_W_PROGRESS_ABORT, [17](#)
 - SAVAPI_W_RATIO_EXCEEDED, [17](#)
 - SAVAPI_W_SUSPICIOUS, [17](#)
- scan_answer
 - SAVAPI_APC_REPORT_DATA, [128](#)
 - SAVAPI_file_status_data, [138](#)
 - SAVAPI_simple_scan_file_data, [156](#)
- scan_details_data
 - SAVAPI_report_scan_details_data, [153](#)
- scan_timeout
 - SAVAPI_instance_init, [143](#)
- scm_pending_time
 - SAVAPI_OA_global_init, [148](#)
- SEEK_CUR
 - fops.h, [169](#)
- SEEK_END
 - fops.h, [169](#)
- SEEK_SET
 - fops.h, [169](#)
- set_report_info
 - SAVAPI_apc_scan_data, [131](#)
- set_timeout
 - SAVAPI_instance_init, [143](#)
- sid
 - SAVAPI_OA_file_result_data, [147](#)
- sid_len
 - SAVAPI_OA_file_result_data, [147](#)
- signal_data
 - SAVAPI_signal_data, [154](#)
- signal_id
 - SAVAPI_command_data, [134](#)
 - SAVAPI_signal_data, [154](#)
- stage
 - SAVAPI_apc_scan_data, [131](#)
- stats
 - SAVAPI_simple_scan_output, [157](#)
- stchar.h, [197](#)
 - CharToSTCHAR, [198](#)
 - SAVAPI_EXP, [198](#)
 - SAVAPI_SIZE_T, [198](#)
 - SAVAPI_TCHAR, [198](#)
 - STCHARToChar, [198](#)
- STCHARToChar
 - stchar.h, [198](#)
- STCHARToChar_t
 - SAVAPI STCHAR function pointers, [108](#)
- store_cache
 - SAVAPI_APC_REPORT_DATA, [128](#)
- STR
 - SAVAPI constants, [12](#)
- strict
 - SAVAPI_malware_info, [146](#)
- suspicious
 - SAVAPI_simple_scan_statistics, [158](#)
- temp_dir
 - SAVAPI_APC_global_init, [127](#)
- threads_number
 - SAVAPI_OA_global_init, [148](#)
- total_files
 - SAVAPI_simple_scan_statistics, [158](#)
- ttl
 - SAVAPI_APC_REPORT_DATA, [128](#)
- type
 - SAVAPI_callback_data, [133](#)
 - SAVAPI_file_info, [137](#)
 - SAVAPI_key_value, [145](#)
 - SAVAPI_malware_info, [146](#)
 - SAVAPI_report_content_data, [151](#)
 - SAVAPI_report_scan_details_data, [153](#)
- types for a SAVAPI FOPS plugin, [107](#)
 - SAVAPI_PLG_AVE_FOPS, [107](#)
 - SAVAPI_PLG_fops_inst_options_t, [108](#)
 - SAVAPI_PLG_fops_instance_t, [108](#)
 - SAVAPI_PLG_fops_options_t, [108](#)
- UINT16
 - fopstypes.h, [173](#)
- UINT32
 - fopstypes.h, [174](#)
- UINT64
 - fopstypes.h, [174](#)
- UINT8
 - fopstypes.h, [174](#)
- uninit
 - _SAVAPI_PLG_info_t, [122](#)
- url
 - SAVAPI_iframe_url_data, [141](#)
- user_data
 - SAVAPI_callback_data, [133](#)
- username
 - SAVAPI_instance_init, [144](#)

value
 SAVAPI_key_value, [145](#)
vdfs_dirpath
 SAVAPI_global_init, [140](#)
version
 _SAVAPI_PLG_info_t, [123](#)
 SAVAPI_callback_data, [133](#)

warning
 SAVAPI_file_status_data, [138](#)