wolfSentry Embedded Firewall/IDPS v1.6.2 API Reference

WolfSSL Inc., Tue Jan 2 2024

Chapter 1

wolfSentry – The Wolfssl Embedded Firewall/IDPS

Description

wolfSentry is the wolfSSL embedded IDPS (Intrusion Detection and Prevention System). In simple terms, wolfSentry is an embedded firewall engine (both static and fully dynamic), with prefix-based and wildcard-capable lookup of known hosts/netblocks qualified by interface, address family, protocol, port, and other traffic parameters. Additionally, wolfSentry can be used as a dynamically configurable logic hub, arbitrarily associating user-defined events with user-defined actions, contextualized by connection attributes. The evolution of client-server relationships can thus be tracked in detail, freely passing traffic matching expected usage patterns, while efficiently rejecting abusive traffic.

wolfSentry is fully integrated with the lwIP stack, through a patchset in the `lwip/` subdirectory of the source tree, and has basic integration with the wolfSSL library for application-level filtering of inbound and outbound connections.

The wolfSentry engine is dynamically configurable programmatically through an API, or from a textual input file in JSON supplied to the engine, or dynamically and incrementally with JSON fragments, or any combination of these methods. Reconfiguration is protected by transactional semantics, and advanced internal locks on threaded targets assure seamless service availability with atomic policy transition. Callbacks allow for transport-agnostic remote logging, e.g. through MQTT, syslog, or DDS message buses.

wolfSentry is designed from the ground up to function well in resource-constrained, bare-metal, and realtime environments, with algorithms to stay within designated maximum memory footprints and maintain deterministic throughput. This allows full firewall and IDPS functionality on embedded targets such as FreeRTOS, Nucleus, NUTTX, Zephyr, VxWorks, and Green Hills Integrity, and on ARM and other common embedded CPUs and MCUs. wolfSentry with dynamic firewalling can add as little as 64k to the code footprint, and 32k to the volatile state footprint, and can fully leverage the existing logic and state of applications and sibling libraries.

Documentation

With `doxygen` installed, the HTML version of the full API reference manual can be generated from the top of the wolfSentry source tree with `make doc-html`. This, and the source code itself, are the recommended API references.

The PDF version of the API reference manual is pregenerated and included with source distributions in the `doc/` subdirectory at `doc/wolfSentry_refman.pdf`. The latest version is always available on GitHub.

Dependencies

In its default build, wolfSentry depends on a POSIX runtime, specifically the heap allocator, clock_gettime, stdio, semaphore, pthreads, and string APIs. However, these dependencies can be avoided with various build-time options. The recipe

```
make STATIC=1 SINGLETHREADED=1 NO_STDIO=1 EXTRA_CFLAGS="-DWOLFSENTRY_NO_↩
CLOCK_BUILTIN -DWOLFSENTRY_NO_MALLOC_BUILTIN"
```

builds a libwolfsentry.a that depends on only a handful of basic string functions and the `inet_ntop()` library function (from POSIX.1-2001, and also implemented by lwIP). Allocator and time callbacks must then be set in a `struct wolfsentry_host_platform_interface` supplied to `wolfsentry_init()`.

The wolfSentry `Makefile` depends on a modern (v4.0+) Gnu `make`. The library itself can be built outside `make`, within another project/framework, by creating a user settings macro file and passing its path to the compiler with the `WOLFSENTRY_USER_SETTINGS_FILE` macro.

Building

wolfSentry was written with portability in mind, with provisions for non-POSIX and C89 targets. For example, all its dependencies can be met with the FreeRTOS/newlib-nano/lwIP runtime. If you have difficulty building wolfSentry, please don't hesitate to seek support through our support forums or contact us directly at support@wolfssl.com.

The current wolfSentry release can be downloaded from the wolfSSL website as a ZIP file, and developers can browse the release history and clone the wolfSentry Git repository for the latest pre-release updates.

There are several flags that can be passed to `make` to control the build parameters. `make` will store them at build time in `wolfsentry/wolfsentry_options.h` in the build tree. If you are not using `make`, then the C macro `WOLFSENTRY_USER_SETTINGS_FILE` should be defined to the path to a file containing settings, both when building wolfSentry and when building the application.

The following feature control variables are recognized. True/false features (`LWIP`, `NO_STDIO`, `NO_JSON`, etc.) are undefined by default, and activated when defined. Macros can be supplied using the `EXTRA_CFLAGS` option, or by placing them in a `USER_SETTINGS_FILE`. More detailed documentation for macros is available in the reference manual "Startup/Configuration/Shutdown Subsystem" topic.

make Option	Macro Option	Description
`SHELL`		Supplies an explicit/alternative path to `bash`.
`AWK`		Supplies an explicit/alternative path to Gnu `awk`.
`V`		Verbose `make` output e.g. `make V=1 -j test`
`USER_MAKE_CONF`		User-defined make clauses to include at the top of the main Makefile e.g. `make -j USER_MAKE_↩ CONF=Makefile.settings`
`EXTRA_CFLAGS`		Additional arguments to be passed verbatim to the compiler
`EXTRA_LDFLAGS`		Additional arguments to be passed verbatim to the linker

make Option	Macro Option	Description
SRC_TOP		The source code top level directory (default pwd -P)
BUILD_TOP		Build with artifacts in an alternate location (outside or in a subdirectory of the source tree) e.g. make BUILD_TOP=./build -j test
DEBUG		Compiler debugging flag to use (default -ggdb)
OPTIM		The optimizer flag to use (default -O3)
HOST		The target host tuple, for cross-compilation (default unset, i.e. native targeting)
RUNTIME		The target runtime ecosystem – default unset, FreeRTOS-lwIP and Linux-lwIP are recognized
C_WARNFLAGS		The warning flags to use (overriding the generally applicable defaults)
STATIC		Build statically linked unit tests
STRIPPED		Strip binaries of debugging symbols
FUNCTION_SECTIONS		Cull any unused object code (with function granularity) to minimize total size.
BUILD_DYNAMIC		Build dynamically linked library
VERY_QUIET		Inhibit all non-error output during build
TAR		Path to GNU tar binary for make dist, should be set to gtar for macOS
VERSION		The version to package for make dist
LWIP	WOLFSENTRY_LWIP	True/false – Activates appropriate build settings for lwIP
NO_STDIO_STREAMS	WOLFSENTRY_NO_STDIO_STREAMS	Define to omit functionality that depends on stdio stream I/O
	WOLFSENTRY_NO_STDIO_H	Define to inhibit inclusion of stdio.h
NO_ADDR_BITMASK_MATCHING	WOLFSENTRY_NO_ADDR_BITMASK_MATCHING	Define to omit support for bitmask matching of addresses, i.e. support only prefix matching.
NO_IPV6	WOLFSENTRY_NO_IPV6	Define to omit support for the IPv6 address family.
NO_JSON	WOLFSENTRY_NO_JSON	Define to omit JSON configuration support
NO_JSON_DOM	WOLFSENTRY_NO_JSON_DOM	Define to omit JSON DOM API
CALL_TRACE	WOLFSENTRY_DEBUG_CALL_TRACE	Define to activate runtime call stack logging (profusely verbose)
USER_SETTINGS_FILE	WOLFSENTRY_USER_SETTINGS_FILE	A substitute settings file, replacing autogenerated wolfsentry_settings.h

make Option	Macro Option	Description
SINGLETHREADED	WOLFSENTRY_↩ SINGLETHREADED	Define to omit thread safety logic, and replace thread safety functions and macros with no-op macros.
	WOLFSENTRY_NO_↩ PROTOCOL_NAMES	If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
	WOLFSENTRY_NO_↩ GETPROTOBY	Define to disable lookup and rendering of protocols and services by name.
	WOLFSENTRY_NO_ERROR_↩ STRINGS	If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
	WOLFSENTRY_NO_MALLOC_↩ BUILTINS	If defined, omit built-in heap allocator primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_allocator.
	WOLFSENTRY_HAVE_↩ NONGNU_ATOMICS	Define if gnu-style atomic intrinsics are not available. WOLFSENTRY↩ _ATOMIC_*() macro definitions for intrinsics will need to be supplied in WOLFSENTRY_↩ USER_SETTINGS_FILE (see wolfsentry_util.h).
	WOLFSENTRY_NO_CLOCK_↩ BUILTIN	If defined, omit built-in time primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_timecbs.
	WOLFSENTRY_NO_SEM_↩ BUILTIN	If defined, omit built-in semaphore primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in struct wolfsentry_semcbs.
	WOLFSENTRY_USE_↩ NONPOSIX_SEMAPHORES	Define if POSIX semaphore API is not available. If no non-POSIX builtin implementation is present in wolfsentry_util.c, then WOLFSENTRY_NO_SEM_BUILTIN must be set, and the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include a full semaphore implementation (shim set) in its wolfsentry_semcbs slot.
	WOLFSENTRY_SEMAPHORE_↩ INCLUDE	Define to the path of a header file declaring a semaphore API.

make Option	Macro Option	Description
	WOLFSENTRY_USE_↩ NONPOSIX_THREADS	Define if POSIX thread API is not available. WOLFSENTRY↩ _THREAD_INCLUDE, WOLFSENTRY_THREAD_ID_T, and WOLFSENTRY_THREAD_↩ GET_ID_HANDLER will need to be defined.
	WOLFSENTRY_THREAD_↩ INCLUDE	Define to the path of a header file declaring a threading API.
	WOLFSENTRY_THREAD_ID_T	Define to the appropriate type analogous to POSIX pthread_t.
	WOLFSENTRY_THREAD_GET↩ _ID_HANDLER	Define to the name of a void function analogous to POSIX pthread_self, returning a value of type WOLFSENTRY_↩ THREAD_ID_T.
	FREERTOS	Build for FreeRTOS

Build and Self-Test Examples

Building and testing libwolfsentry.a on Linux:

```
make -j test
```

Build verbosely:

```
make V=1 -j test
```

Build with artifacts in an alternate location (outside or in a subdirectory of the source tree):

```
make BUILD_TOP=./build -j test
```

Install from an alternate build location to a non-standard destination:

```
make BUILD_TOP=./build INSTALL_DIR=/usr INSTALL_LIBDIR=/usr/lib64 install
```

Build libwolfsentry.a and test it in various configurations:

```
make -j check
```

Build and test libwolfsentry.a without support for multithreading:

```
make -j SINGLETHREADED=1 test
```

Other available make flags are STATIC=1, STRIPPED=1, NO_JSON=1, and NO_JSON_DOM=1, and the defaults values for DEBUG, OPTIM, and C_WARNFLAGS can also be usefully overridden.

Build with a user-supplied makefile preamble to override defaults:

```
make -j USER_MAKE_CONF=Makefile.settings
```

(Makefile.settings can contain simple settings like OPTIM := -Os, or elaborate makefile code including additional rules and dependency mechanisms.)

Build the smallest simplest possible library:

```
make -j SINGLETHREADED=1 NO_STDIO=1 DEBUG= OPTIM=-Os EXTRA_CFLAGS="-DWOLFSENTRY↩
_NO_CLOCK_BUILTIN -DWOLFSENTRY_NO_MALLOC_BUILTIN -DWOLFSENTRY_NO_ERROR_↩
STRINGS -Wno-error=inline -Wno-inline"
```

Build and test with user settings:

```
make -j USER_SETTINGS_FILE=user_settings.h test
```

Build for FreeRTOS on ARM32, assuming FreeRTOS and lwIP source trees are located as shown:

```
make -j HOST=arm-none-eabi RUNTIME=FreeRTOS-lwIP FREERTOS_TOP=../third/↩
FreeRTOSv202212.00 LWIP_TOP=../third/lwip EXTRA_CFLAGS=-mcpu=cortex-m7
```

Project Examples

In the `wolfsentry/examples/` subdirectory are a set of example ports and applications, including a demo pop-up notification system implementing a toy TLS-enabled embedded web server, integrating with the Linux D-Bus facility.

More comprehensive examples of API usage are in `tests/unittests.c`, particularly `test_static_routes()`, `test_dynamic_rules()`, and `test_json()`, and the JSON configuration files at `tests/test-config*.json`.

In the wolfSSL repository, see code in `wolfssl/test.h` gated on `WOLFSSL_WOLFSENTRY_HOOKS`, including `wolfsentry_store_endpoints()`, `wolfSentry_NetworkFilterCallback()`, `wolfsentry_setup()`, and `tcp_connect_with_wolfSentry()`. See also code in `examples/server/server.c` and `examples/client/client.c` gated on `WOLFSSL_WOLFSENTRY_HOOKS`. Configure wolfssl with `--enable-wolfsentry` to build with wolfSentry integration, and use `--with-wolfsentry=/the/install/path` if wolfSentry is installed in a nonstandard location. The wolfSSL test client/server can be loaded with user-supplied wolfSentry JSON configurations from the command line, using `--wolfsentry-config <file>`.

Chapter 2

Building and Initializing wolfSentry for an application on FreeRTOS/lwIP

Building the wolfSentry library for FreeRTOS with lwIP and newlib-nano is supported directly by the top level `Makefile`. E.g., for an ARM Cortex M7, `libwolfsentry.a` can be built with

```
make HOST=arm-none-eabi EXTRA_CFLAGS='-mcpu=cortex-m7' RUNTIME=FreeRTOS-lwIP FREERTOS_TOP="$FREERTOS_TOP"
      LWIP_TOP="$LWIP_TOP"
```

`FREERTOS_TOP` is the path to the top of the FreeRTOS distribution, with `FreeRTOS/Source` directly under it, and `LWIP_TOP` is the path to the top of the lwIP distribution, with `src` directly under it.

The below code fragments can be added to a FreeRTOS application to enable wolfSentry with dynamically loaded policies (JSON). Many of the demonstrated code patterns are optional. The only calls that are indispensable are `wolfsentry_init()`, `wolfsentry_config_json_oneshot()`, and `wolfsentry_install_lwip_filter_callbacks()`. Each of these also has API variants that give the user more control.

```
#define WOLFSENTRY_SOURCE_ID WOLFSENTRY_SOURCE_ID_USER_BASE
#define WOLFSENTRY_ERROR_ID_USER_APP_ERR0 (WOLFSENTRY_ERROR_ID_USER_BASE-1)
 /* user-defined error IDs count down starting at WOLFSENTRY_ERROR_ID_USER_BASE (which is negative). */

#include <wolfsentry/wolfsentry_json.h>
#include <wolfsentry/wolfsentry_lwip.h>

static struct wolfsentry_context *wolfsentry_lwip_ctx = NULL;

static const struct wolfsentry_eventconfig demo_config = {
#ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
        .route_private_data_size = 64,
        .route_private_data_alignment = 0,         /* default alignment -- same as sizeof(void *). */
        .max_connection_count = 10,                /* by default, don't allow more than 10 simultaneous
                                                    * connections that match the same route.
                                                    */
        .derogatory_threshold_for_penaltybox = 4,  /* after 4 derogatory events matching the same route,
                                                    * put the route in penalty box status.
                                                    */
        .penaltybox_duration = 300,                /* keep routes in penalty box status for 5 minutes.
                                                    * denominated in seconds when passing to
                                                    * wolfsentry_init().
                                                    */
        .route_idle_time_for_purge = 0,            /* 0 to disable -- autopurge doesn't usually make
                                                    * much sense as a default config.
                                                    */
        .flags = WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY, /* automatically clear
                                                    * derogatory count for a route when a commendable
                                                    * event matches the route.
                                                    */
        .route_flags_to_add_on_insert = 0,
        .route_flags_to_clear_on_insert = 0,
        .action_res_filter_bits_set = 0,
        .action_res_filter_bits_unset = 0,
        .action_res_bits_to_add = 0,
        .action_res_bits_to_clear = 0
```

```
#else
        64,
        0,
        10,
        4,
        300,
        0,
        WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY,
        0,
        0,
        0,
        0,
        0,
        0
#endif
    };

/* This routine is to be called once by the application before any direct calls
 * to lwIP -- i.e., before lwip_init() or tcpip_init().
 */
wolfsentry_errcode_t activate_wolfsentry_lwip(const char *json_config, int json_config_len)
{
    wolfsentry_errcode_t ret;
    char err_buf[512]; /* buffer for detailed error messages from
                        * wolfsentry_config_json_oneshot().
                        */

    /* Allocate a thread state struct on the stack.  Note that the final
     * semicolon is supplied by the macro definition, so that in single-threaded
     * application builds this expands to nothing at all.
     */
    WOLFSENTRY_THREAD_HEADER_DECLS

    if (wolfsentry_lwip_ctx != NULL) {
        printf("activate_wolfsentry_lwip() called multiple times.\n");
        WOLFSENTRY_ERROR_RETURN(ALREADY);
    }

#ifdef WOLFSENTRY_ERROR_STRINGS
    /* Enable pretty-printing of the app source code filename for
     * WOLFSENTRY_ERROR_FMT/WOLFSENTRY_ERROR_FMT_ARGS().
     */
    ret = WOLFSENTRY_REGISTER_SOURCE();
    WOLFSENTRY_RERETURN_IF_ERROR(ret);

    /* Enable pretty-printing of an app-specific error code. */
    ret = WOLFSENTRY_REGISTER_ERROR(USER_APP_ERR0, "failure in application code");
    WOLFSENTRY_RERETURN_IF_ERROR(ret);
#endif

    /* Initialize the thread state struct -- this sets the thread ID. */
    WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(WOLFSENTRY_THREAD_FLAG_NONE);

    /* Call the main wolfSentry initialization routine.
     *
     * WOLFSENTRY_CONTEXT_ARGS_OUT() is a macro that abstracts away
     * conditionally passing the thread struct pointer to APIs that need it.  If
     * this is a single-threaded build (!defined(WOLFSENTRY_THREADSAFE)), then
     * the thread arg is omitted entirely.
     *
     * WOLFSENTRY_CONTEXT_ARGS_OUT_EX() is a variant that allows the caller to
     * supply the first arg explicitly, when "wolfsentry" is not the correct arg
     * to pass.  This is used here to pass a null pointer for the host platform
     * interface ("hpi").
     */
    ret = wolfsentry_init(
        wolfsentry_build_settings,
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(NULL /* hpi */),
        &demo_config,
        &wolfsentry_lwip_ctx);
    if (ret < 0) {
        printf("wolfsentry_init() failed: " WOLFSENTRY_ERROR_FMT "\n",
               WOLFSENTRY_ERROR_FMT_ARGS(ret));
        goto out;
    }

    /* Insert user-defined actions here, if any. */
    ret = wolfsentry_action_insert(
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),
        "my-action",
        WOLFSENTRY_LENGTH_NULL_TERMINATED,
        WOLFSENTRY_ACTION_FLAG_NONE,
        my_action_handler,
        NULL,
        NULL);
    if (ret < 0) {
        printf("wolfsentry_action_insert() failed: " WOLFSENTRY_ERROR_FMT "\n",
```

```
                WOLFSENTRY_ERROR_FMT_ARGS(ret));
        goto out;
    }

    if (json_config) {
        if (json_config_len < 0)
            json_config_len = (int)strlen(json_config);

        /* Do the initial load of the policy. */
        ret = wolfsentry_config_json_oneshot(
            WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),
            (unsigned char *)json_config,
            (size_t)json_config_len,
            WOLFSENTRY_CONFIG_LOAD_FLAG_NONE,
            err_buf,
            sizeof err_buf);
        if (ret < 0) {
            printf("wolfsentry_config_json_oneshot() failed: %s\n", err_buf);
            goto out;
        }
    } /* else the application will need to set up the policy programmatically,
       * or itself call wolfsentry_config_json_oneshot() or sibling APIs.
       */

    /* Install lwIP callbacks.  Once this call returns with success, all lwIP
     * traffic designated for filtration by the mask arguments shown below will
     * be subject to filtering (or other supplementary processing) according to
     * the policy loaded above.
     *
     * Note that if a given protocol is gated out of LWIP, its mask argument
     * must be passed as zero here, else the call will return
     * IMPLEMENTATION_MISSING error will occur.
     *
     * The callback installation also registers a cleanup routine that will be
     * called automatically by wolfsentry_shutdown().
     */

#define LWIP_ALL_EVENTS (                       \
        (1U << FILT_BINDING) |                  \
        (1U << FILT_DISSOCIATE) |               \
        (1U << FILT_LISTENING) |                \
        (1U << FILT_STOP_LISTENING) |           \
        (1U << FILT_CONNECTING) |               \
        (1U << FILT_ACCEPTING) |                \
        (1U << FILT_CLOSED) |                   \
        (1U << FILT_REMOTE_RESET) |             \
        (1U << FILT_RECEIVING) |                \
        (1U << FILT_SENDING) |                  \
        (1U << FILT_ADDR_UNREACHABLE) |         \
        (1U << FILT_PORT_UNREACHABLE) |         \
        (1U << FILT_INBOUND_ERR) |              \
        (1U << FILT_OUTBOUND_ERR))

    ret = wolfsentry_install_lwip_filter_callbacks(
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfsentry_lwip_ctx),

#if LWIP_ARP || LWIP_ETHERNET
        LWIP_ALL_EVENTS, /* ethernet_mask */
#else
        0,
#endif
#if LWIP_IPV4 || LWIP_IPV6
        LWIP_ALL_EVENTS, /* ip_mask */
#else
        0,
#endif
#if LWIP_ICMP || LWIP_ICMP6
        LWIP_ALL_EVENTS, /* icmp_mask */
#else
        0,
#endif
#if LWIP_TCP
        LWIP_ALL_EVENTS, /* tcp_mask */
#else
        0,
#endif
#if LWIP_UDP
        LWIP_ALL_EVENTS /* udp_mask */
#else
        0
#endif
        );
    if (ret < 0) {
        printf("wolfsentry_install_lwip_filter_callbacks: "
               WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
```

```
    }

out:
    if (ret < 0) {
        /* Clean up if initialization failed. */
        wolfsentry_errcode_t shutdown_ret =
            wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_OUT_EX(&wolfsentry_lwip_ctx));
        if (shutdown_ret < 0)
            printf("wolfsentry_shutdown: "
                   WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(shutdown_ret));
    }

    WOLFSENTRY_THREAD_TAILER_CHECKED(WOLFSENTRY_THREAD_FLAG_NONE);

    WOLFSENTRY_ERROR_RERETURN(ret);
}

/* to be called once by the application after any final calls to lwIP. */
wolfsentry_errcode_t shutdown_wolfsentry_lwip(void)
{
    wolfsentry_errcode_t ret;
    if (wolfsentry_lwip_ctx == NULL) {
        printf("shutdown_wolfsentry_lwip() called before successful activation.\n");
        return -1;
    }

    /* after successful shutdown, wolfsentry_lwip_ctx will once again be a null
     * pointer as it was before init.
     */
    ret = wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(&wolfsentry_lwip_ctx, NULL));
    if (ret < 0) {
        printf("wolfsentry_shutdown: "
               WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
    }

    return ret;
}
```

Chapter 3

Configuring wolfSentry using a JSON document

Most of the capabilities of wolfSentry can be configured, and dynamically reconfigured, by supplying JSON documents to the library. To use this capability, add the following to wolfSentry initialization in the application:

```
#include <wolfsentry/wolfsentry_json.h>
```

After initialization and installation of application-supplied callbacks (if any), call one of the APIs to load the config:

- `wolfsentry_config_json_oneshot()`
- `wolfsentry_config_json_oneshot_ex()`, with an additional `json_config` arg for fine control of JSON parsing (see `struct JSON_CONFIG` in `wolfsentry/centijson_sax.h`)
- streaming API:
  - `wolfsentry_config_json_init()` or `wolfsentry_config_json_init_ex()`
  - `wolfsentry_config_json_feed()`
  - `wolfsentry_config_json_fini()`

See `wolfsentry/wolfsentry_json.h` for details on arguments.

JSON Basics

wolfSentry configuration uses standard JSON syntax as defined in RFC 8259, as restricted by RFC 7493, with certain additional requirements. In particular, certain sections in the JSON document are restricted in their sequence of appearance.

- `"wolfsentry-config-version"` shall appear first, and each event definition shall appear before any definitions for events, routes, or default policies that refer to it through `"aux-parent-event"`, `"parent-event"`, or `"default-event"` clauses.
- Within event definitions, the `"label"`, `"priority"`, and `"config"` elements shall appear before any other elements.

These sequence constraints are necessary to allow for high efficiency SAX-style (sequential-incremental) loading of the configuration.

All wildcard flags are implicitly set on routes, and are cleared for fields with explicit assignments in the configuration. For example, if a route designates a particular "family", then WOLFSENTRY_ROUTE_FLAG_SA_FAMILY↩_WILDCARD will be implicitly cleared. Thus, wildcard flags need not be explicitly set or cleared in route definitions.

Note that certain element variants may be unavailable due to build settings:

- address_family_name: available if defined(WOLFSENTRY_PROTOCOL_NAMES)
- route_protocol_name: available if !defined(WOLFSENTRY_NO_GETPROTOBY)
- address_port_name: available if !defined(WOLFSENTRY_NO_GETPROTOBY)
- json_value_clause: available if defined(WOLFSENTRY_HAVE_JSON_DOM)

Caller-supplied event and action labels shall not begin with WOLFSENTRY_BUILTIN_LABEL_PREFIX (by default "%"), as these are reserved for built-ins.

"config-update" allows the default configuration to be updated. It is termed an "update" because wolfSentry is initially configured by the config argument to wolfsentry_init() (which can be passed in NULL, signifying built-in defaults). Note that times (wolfsentry_eventconfig.penaltybox_duration and wolfsentry_eventconfig.route_idle_time_for_purge) shall be passed to wolfsentry_init() denominated in seconds, notwithstanding the wolfsentry_time_t type of the members.

JSON load flags

The flags argument to wolfsentry_config_json_init() and wolfsentry_config_json_oneshot(), constructed by bitwise-or, changes the way the JSON is processed, as follows:

- WOLFSENTRY_CONFIG_LOAD_FLAG_NONE – Not a flag, but all-zeros, signifying default behavior: The wolfSentry core is locked, the current configuration is flushed, and the new configuration is loaded incrementally. Any error during load leaves wolfSentry in an undefined state that can be recovered with a subsequent flush and load that succeeds.
- WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH – Inhibit initial flush of configuration, to allow incremental load. Error during load leaves wolfSentry in an undefined state that can only be recovered with a subsequent flush and load that succeeds, unless WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN or WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT was also supplied.
- WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN – Load into a temporary configuration, and deallocate before return. Running configuration is unchanged.
- WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT – Load into a newly allocated configuration, and install it only if load completes successfully. On error, running configuration is unchanged. On success, the old configuration is deallocated.
- WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS – Inhibit loading of "routes" and "events" sections in the supplied JSON.
- WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES – At beginning of load process, retain all current configuration except for routes, which are flushed. This is convenient in combination with wolfsentry_route_table_dump_json_*() for save/restore of dynamically added routes.
- WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT – When processing user-defined JSON values, abort load on duplicate keys.

- WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST – When processing user-defined JSON values, for any given key in an object use the first occurrence encountered.
- WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST – When processing user-defined JSON values, for any given key in an object use the last occurrence encountered.
- WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER – When processing user-defined JSON values, store sequence information so that subsequent calls to wolfsentry_kv_render_value() or json_dom_dump(..., JSON_DOM_DUMP_PREFERDICTORDER) render objects in their supplied sequence, rather than lexically sorted.

Note that WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_* flags are allowed only if WOLFSENTRY_HAVE_JSON_DOM is defined in the build, as it is with default settings.

Overview of JSON syntax

Below is a JSON "lint" pseudodocument demonstrating all available configuration nodes, with value specifiers that refer to the ABNF definitions below. The allowed values are as in the ABNF formal syntax later in this document.

```
{
    "wolfsentry-config-version" : 1,
    "config-update" : {
        "max-connection-count" : uint32,
        "penalty-box-duration" : duration,
        "route-idle-time-for-purge" : duration,
        "derog-thresh-for-penalty-boxing" : uint16,
        "derog-thresh-ignore-commendable" : boolean,
        "commendable-clears-derogatory" : boolean,
        "route-flags-to-add-on-insert" : route_flag_list,
        "route-flags-to-clear-on-insert" : route_flag_list,
        "action-res-filter-bits-set" : action_res_flag_list,
        "action-res-filter-bits-unset" : action_res_flag_list,
        "action-res-bits-to-add" : action_res_flag_list,
        "action-res-bits-to-clear" : action_res_flag_list,
        "max-purgeable-routes" : uint32,
        "max-purgeable-idle-time" : duration
    },
    "events" : [
       { "label" : label,
         "priority" : uint16,
         "config" : {
            "max-connection-count" : uint32,
            "penalty-box-duration" : duration,
            "route-idle-time-for-purge" : duration,
            "derog-thresh-for-penalty-boxing" : uint16,
            "derog-thresh-ignore-commendable" : boolean,
            "commendable-clears-derogatory" : boolean,
            "route-flags-to-add-on-insert" : route_flag_list,
            "route-flags-to-clear-on-insert" : route_flag_list,
            "action-res-filter-bits-set" : action_res_flag_list,
            "action-res-filter-bits-unset" : action_res_flag_list,
            "action-res-bits-to-add" : action_res_flag_list,
            "action-res-bits-to-clear" : action_res_flag_list
         },
         "aux-parent-event"  : label,
         "post-actions" : action_list,
         "insert-actions" : action_list,
         "match-actions" : action_list,
         "update-actions" : action_list,
         "delete-actions" : action_list,
         "decision-actions" : action_list
      }
    ],
    "default-policies" : {
        "default-policy" : default_policy_value,
        "default-event" ":" label
    },
    "routes" : [
      {
        "parent-event" : label,
        "af-wild" : boolean,
        "raddr-wild" : boolean,
        "rport-wild" : boolean,
        "laddr-wild" : boolean,
```

```
        "lport-wild" : boolean,
        "riface-wild" : boolean,
        "liface-wild" : boolean,
        "tcplike-port-numbers" : boolean,
        "direction-in" : boolean,
        "direction-out" : boolean,
        "penalty-boxed" : boolean,
        "green-listed" : boolean,
        "dont-count-hits" : boolean,
        "dont-count-current-connections" : boolean,
        "port-reset" : boolean,

        "family" : address_family,
        "protocol" : route_protocol,
        "remote" : {
          "interface" : uint8,
          "address" : route_address,
          "prefix-bits" : uint16,
          "bitmask" : route_address,
          "port" : endpoint_port
        },
        "local" : {
          "interface" : uint8,
          "address" : route_address,
          "prefix-bits" : uint16,
          "bitmask" : route_address,
          "port" : endpoint_port
        }
      }
    ],
    "user-values" : {
      label : null,
      label : true,
      label : false,
      label : number_sint64,
      label : number_float,
      label : string,
      label : { "uint" : number_uint64 },
      label : { "sint" : number_sint64 },
      label : { "float" : number_float },
      label : { "string" : string_value },
      label : { "base64" : base64_value },
      label : { "json" : json_value }
    }
}
```

Descriptions of elements

`wolfsentry-config-version` – Shall appear first, with the value `1`.

`config-update` – Sets default and global parameters. The default parameters apply to routes that have no parent event, or a parent event with no config of its own.

- **`max-connection-count`** – If nonzero, the concurrent connection limit, beyond which additional connection requests are rejected.
- **`penalty-box-duration`** – If nonzero, the duration that a route stays in penalty box status before automatic release.
- **`derog-thresh-for-penalty-boxing`** – If nonzero, the threshold at which accumulated derogatory counts (from `WOLFSENTRY_ACTION_RES_DEROGATORY` incidents) automatically penalty boxes a route.
- **`derog-thresh-ignore-commendable`** – If true, then counts from `WOLFSENTRY_ACTION_RES_COMMENDABLE` are not subtracted from the derogatory count when checking for automatic penalty boxing.
- **`commendable-clears-derogatory`** – If true, then each count from `WOLFSENTRY_ACTION_RES_COMMENDABLE` zeroes the derogatory count.
- **`max-purgeable-routes`** – Global limit on the number of ephemeral routes to allow in the route table, beyond which the least recently matched ephemeral route is forced out early. Not allowed in **`config`** clauses of events.

- **max-purgeable-idle-time** – Global absolute maximum idle time for ephemeral routes, controlling purges of stale (expired) ephemeral routes with nonzero wolfsentry_route_metadata_exports.connection_co Default is no limit. Not allowed in **config** clauses of events.
- **route-idle-time-for-purge** – If nonzero, the time after the most recent dispatch match for a route to be garbage-collected. Useful primarily in **config** clauses of events (see **events** below).
- **route-flags-to-add-on-insert** – List of route flags to set on new routes upon insertion. Useful primarily in **config** clauses of events (see **events** below).
- **route-flags-to-clear-on-insert** – List of route flags to clear on new routes upon insertion. Useful primarily in **config** clauses of events (see **events** below).
- **action-res-filter-bits-set** – List of action_res flags that must be set at lookup time (dispatch) for referring routes to match. Useful primarily in **config** clauses of events (see **events** below).
- **action-res-filter-bits-unset** – List of action_res flags that must be clear at lookup time (dispatch) for referring routes to match. Useful primarily in **config** clauses of events (see **events** below).
- **action-res-bits-to-add** – List of action_res flags to be set upon match.
- **action-res-bits-to-clear** – List of action_res flags to be cleared upon match.

events – The list of events with their respective definitions. This section can appear more than once, but any given event definition shall precede any definitions that refer to it.

Each event is composed of the following elements, all of which are optional except for **label**. **label**, **priority**, and **config** shall appear before the other elements.

- **label** – The name by which the event is identified. See the definition of label in the ABNF grammar below for permissible values.
- **priority** – The priority of routes that have this event as their **parent-event** (see **routes** below). Lower number means higher priority.
- **config** – The configuration to associate with routes with this **parent-event**, as above for **config-update**.
- **aux-parent-event** – An event reference for use by action handlers, e.g. built-in "%track-peer-v1" creates routes with **aux-parent-event** as the new route's **parent-event**.
- **post-actions** – List of actions to take when this event is passed via **event_label** to a dispatch routine such as wolfsentry_route_event_dispatch().
- **insert-actions** – List of actions to take when a route is inserted with this event as **parent-event**.
- **match-actions** – List of actions to take when a route is matched by a dispatch routine, and the route has this event as its **parent-event**.
- **update-actions** – List of actions to take when a route has a status update, such as a change of penalty box status, and has this event as its **parent-event**.
- **delete-actions** – List of actions to take when a route is deleted, and has this event as its **parent-event**.
- **decision-actions** – List of actions to take when dispatch final decision (final value of **action_↩ results**) is determined, and the matched route has this event as its **parent-event**.

default-policies – The global fallthrough default policies for dispatch routines such as wolfsentry_route_event_disp

- **default-policy** – A simple **action_result** flag to set by default, either **accept**, **reject**, or **reset**, the latter of which causes generation of TCP reset and ICMP unreachable reply packets where relevant.

- **default-event** – An event to use when a dispatch routine is called with a null **event_label**.

routes – The list of routes with their respective definitions. This section can appear more than once.

Each route is composed of the following elements, all of which are optional.

- **parent-event** – The event whose attributes determine the dynamics of the route.
- **family** – The address family to match. See `address_family` definition in the ABNF grammar below for permissible values.
- **protocol** – The protocol to match. See `route_protocol` definition in the ABNF grammar below for permissible values.
- **remote** – The attributes to match for the remote endpoint of the traffic.
  - **interface** – Network interface ID, as an arbitrary integer chosen and used consistently by the caller or IP stack integration.
  - **address** – The network address, in idiomatic form. IPv4, IPv6, and MAC addresses shall enumerate all octets. See `route_address` definition in the ABNF grammar below for permissible values.
  - **prefix-bits** – The number of bits in the **address** that traffic must match (mutually exclusive with **bitmask**).
  - **bitmask** – A bitmask to be applied to the traffic address before matching with the route **address** (mutually exclusive with **prefix-bits**).
  - **port** – The port number that traffic must match.
- **local** – The attributes to match for the local endpoint of the traffic. The same nodes are available as for **remote**.
- **direction-in** – If true, match inbound traffic.
- **direction-out** – If true, match outbound traffic.
- **penalty-boxed** – If true, traffic matching the route is penalty boxed (rejected or reset).
- **green-listed** – If true, traffic matching the route is accepted.
- **dont-count-hits** – If true, inhibit statistical bookkeeping (no effect on dynamics).
- **dont-count-current-connections** – If true, inhibit tracking of concurrent connections, so that **max-connection-count** has no effect on traffic matching this route.
- **port-reset** – If true, set the `WOLFSENTRY_ACTION_RES_PORT_RESET` flag in the **action_results** when this route is matched, causing TCP reset or ICMP unreachable reply packet to be generated if IP stack integration is activated (e.g. `wolfsentry_install_lwip_filter_callbacks()`).

user-values – One or more sections of fully user-defined data available to application code for any use. Each key is a label as defined in the ABNF grammar below. The value can be any of:

- **null**
- **true**
- **false**
- an integral number, implicitly a signed 64 bit integer
- a floating point number, as defined in the ABNF grammar below for `number_float`
- a quoted string allowing standard JSON escapes

- any of several explicitly typed constructs, with values as defined in the ABNF grammar below.
  - `{ "uint" : number_uint64 }`
  - `{ "sint" : number_sint64 }`
  - `{ "float" : number_float }`
  - `{ "string" : string_value }`
  - `{ "base64" : base64_value }`
  - `{ "json" : json_value }`

Formal ABNF grammar

Below is the formal ABNF definition of the configuration syntax and permitted values.

This definition uses ABNF syntax as prescribed in RFC 5234 and 7405, except:

- Whitespace is ignored, as provided in RFC 8259.
- a − operator is added, accepting a quoted literal string or a group of literal characters, to provide for omitted character(s) in the target text (here, trailing comma separators) by performing all notional matching operations of the containing group up to that point with the target text notionally extended with the argument to the operator.

The length limits used in the definition assume the default values in wolfsentry_settings.h, 32 octets for labels (`WOLFSENTRY_MAX_LABEL_BYTES`), and 16384 octets for user-defined values (`WOLFSENTRY_KV_MAX_↩VALUE_BYTES`). These values can be overridden at build time with user-supplied values.

```
"{"
    DQUOTE %s"wolfsentry-config-version" DQUOTE ":" uint32
    [ "," DQUOTE %s"config-update" DQUOTE ":" top_config_list "," ]
    *("," DQUOTE %s"events" ":" "["
       event *("," event)
    "]")
    [ "," DQUOTE %s"default-policies" DQUOTE ":" "{"
        default_policy_item *("," default_policy_item)
    "}" ]
    *("," DQUOTE %s"routes" DQUOTE ":" "["
        route *("," route)
    "]")
    *("," DQUOTE %s"user-values" DQUOTE ":" "{"
        user_item *("," user_item)
    "}")
"}"

event = "{" label_clause
        [ "," priority_clause ]
        [ "," event_config_clause ]
        [ "," aux_parent_event_clause ]
        *("," action_list_clause) "}"

default_policy_item =
        (DQUOTE %s"default-policy" DQUOTE ":" default_policy_value) /
        (DQUOTE %s"default-event" DQUOTE ":" label)

default_policy_value = (%s"accept" / %s"reject" / %s"reset")

label_clause = DQUOTE %s"label" DQUOTE ":" label

priority_clause = DQUOTE %s"priority" DQUOTE ":" uint16

event_config_clause = DQUOTE %s"config" DQUOTE ":" event_config_list

aux_parent_event_clause = DQUOTE %s"aux-parent-event" DQUOTE ":" label

action_list_clause = DQUOTE (%s"post-actions" / %s"insert-actions" / %s"match-actions"
            / %s"update-actions" / %s"delete-actions" / %s"decision-actions") DQUOTE
```

```
            ":" action_list

action_list = "[" label *("," label) "]"

event_config_list = "{" event_config_item *("," event_config_item) "}"

top_config_list = "{" top_config_item *("," top_config_item) "}"

top_config_item = event_config_item / max_purgeable_routes_clause / max_purgeable_idle_time_clause

event_config_item =
  (DQUOTE %s"max-connection-count" DQUOTE ":" uint32) /
  (DQUOTE %s"penalty-box-duration" DQUOTE ":" duration) /
  (DQUOTE %s"route-idle-time-for-purge" DQUOTE ":" duration) /
  (DQUOTE %s"derog-thresh-for-penalty-boxing" DQUOTE ":" uint16 /
  (DQUOTE %s"derog-thresh-ignore-commendable" DQUOTE ":" boolean /
  (DQUOTE %s"commendable-clears-derogatory" DQUOTE ":" boolean /
  (DQUOTE (%s"route-flags-to-add-on-insert" / %s"route-flags-to-clear-on-insert") DQUOTE ":"
      route_flag_list) /
  (DQUOTE (%s"action-res-filter-bits-set" / %s"action-res-filter-bits-unset" / %s"action-res-bits-to-add" /
      %s"action-res-bits-to-clear") DQUOTE ":" action_res_flag_list)

duration = number_sint64 / (DQUOTE number_sint64 [ %s"d" / %s"h" / %s"m" / %s"s" ] DQUOTE)

max_purgeable_routes_clause = DQUOTE %s"max-purgeable-routes" DQUOTE ":" uint32

max_purgeable_idle_time_clause = DQUOTE %s"max-purgeable-idle-time" DQUOTE ":" duration

route_flag_list = "[" route_flag *("," route_flag) "]"

action_res_flag_list = "[" action_res_flag *("," action_res_flag) "]"

route = "{"
    [ parent_event_clause "," ]
    *(route_flag_clause ",")
    [ family_clause ","
      [ route_protocol_clause "," ]
    ]
    [ route_remote_endpoint_clause "," ]
    [ route_local_endpoint_clause "," ]
    -","
"}"

parent_event_clause = DQUOTE %s"parent-event" DQUOTE ":" label
route_flag_clause = route_flag ":" boolean
family_clause = DQUOTE %s"family" DQUOTE ":" address_family
route_protocol_clause = DQUOTE %s"protocol" DQUOTE ":" route_protocol

route_remote_endpoint_clause = DQUOTE %s"remote" DQUOTE ":" route_endpoint
route_local_endpoint_clause = DQUOTE %s"local" DQUOTE ":" route_endpoint

route_endpoint = "{"
    [ route_interface_clause "," ]
    [ route_address_clause ","
      [ (route_address_prefix_bits_clause / route_address_bitmask_clause) "," ]
    ]
    [ route_port_clause "," ]
    -","
"}"

route_interface_clause = DQUOTE %s"interface" DQUOTE ":" uint8

route_address_clause = DQUOTE %s"address" DQUOTE ":" route_address

route_address_bitmask_clause = DQUOTE %s"bitmask" DQUOTE ":" route_address

route_address = DQUOTE (route_address_ipv4 / route_address_ipv6 / route_address_mac / route_address_user)
      DQUOTE

route_address_ipv4 = uint8 3*3("." uint8)

route_address_ipv6 = < IPv6address from RFC 5954 section 4.1 >

route_address_mac = 1*2HEXDIG ( 5*5(":" 1*2HEXDIG) / 7*7(":" 1*2HEXDIG) )

route_address_user = < an address in a form recognized by a parser
                       installed with `wolfsentry_addr_family_handler_install()`
                     >

address_family = uint16 / address_family_name

address_family_name = DQUOTE ( "inet" / "inet6" / "link" / < a value recognized by
```

```
      wolfsentry_addr_family_pton() > ) DQUOTE

route_address_prefix_bits_clause = DQUOTE %s"prefix-bits" DQUOTE ":" uint16

route_protocol = uint16 / route_protocol_name

route_protocol_name = DQUOTE < a value recognized by getprotobyname_r(), requiring address family inet or
      inet6 >

route_port_clause = DQUOTE %s"port" DQUOTE ":" endpoint_port

endpoint_port = uint16 / endpoint_port_name

endpoint_port_name = DQUOTE < a value recognized by getservbyname_r() for the previously designated protocol
      > DQUOTE

route_flag = DQUOTE (
  %s"af-wild" /
  %s"raddr-wild" /
  %s"rport-wild" /
  %s"laddr-wild" /
  %s"lport-wild" /
  %s"riface-wild" /
  %s"liface-wild" /
  %s"tcplike-port-numbers" /
  %s"direction-in" /
  %s"direction-out" /
  %s"penalty-boxed" /
  %s"green-listed" /
  %s"dont-count-hits" /
  %s"dont-count-current-connections" /
  %s"port-reset"
) DQUOTE

action_res_flag = DQUOTE (
  %s"none" /
  %s"accept" /
  %s"reject" /
  %s"connect" /
  %s"disconnect" /
  %s"derogatory" /
  %s"commendable" /
  %s"stop" /
  %s"deallocated" /
  %s"inserted" /
  %s"error" /
  %s"fallthrough" /
  %s"update" /
  %s"port-reset" /
  %s"sending" /
  %s"received" /
  %s"binding" /
  %s"listening" /
  %s"stopped-listening" /
  %s"connecting-out" /
  %s"closed" /
  %s"unreachable" /
  %s"sock-error" /
  %s"user+0" /
  %s"user+1" /
  %s"user+2" /
  %s"user+3" /
  %s"user+4" /
  %s"user+5" /
  %s"user+6" /
  %s"user+7"
) DQUOTE

user_item = label ":" ( null / true / false / number_sint64_decimal / number_float / string /
      strongly_typed_user_item )

strongly_typed_user_item =
  ( "{" DQUOTE %s"uint" DQUOTE ":" number_uint64 "}" ) /
  ( "{" DQUOTE %s"sint" DQUOTE ":" number_sint64 "}" ) /
  ( "{" DQUOTE %s"float" DQUOTE ":" number_float "}" ) /
  ( "{" DQUOTE %s"string" DQUOTE ":" string_value "}" ) /
  ( "{" DQUOTE %s"base64" DQUOTE ":" base64_value "}" ) /
```

```
  json_value_clause

json_value_clause = "{" DQUOTE %s"json" DQUOTE ":" json_value "}"

null = %s"null"

true = %s"true"

false = %s"false"

boolean = true / false

number_uint64 = < decimal number in the range 0...18446744073709551615 > /
                ( DQUOTE < hexadecimal number in the range 0x0...0xffffffffffffffff > DQUOTE ) /
                ( DQUOTE < octal number in the range 00...01777777777777777777777 > DQUOTE )

number_sint64_decimal = < decimal number in the range -9223372036854775808...9223372036854775807 >

number_sint64 = number_sint64_decimal /
                ( DQUOTE < hexadecimal number in the range -0x8000000000000000...0x7fffffffffffffff > DQUOTE
      ) /
                ( DQUOTE < octal number in the range -01000000000000000000000...07777777777777777777 >
      DQUOTE )

number_float = < floating point value in a form and range recognized by the linked strtod() implementation >

string_value = DQUOTE < any RFC 8259 JSON-valid string that decodes to at most 16384 octets > DQUOTE

base64_value = DQUOTE < any valid RFC 4648 base64 encoding that decodes to at most 16384 octets > DQUOTE

json_value = < any valid, complete and balanced RFC 8259 JSON expression, with
               keys limited to WOLFSENTRY_MAX_LABEL_BYTES (default 32 bytes),
               overall input length limited to WOLFSENTRY_JSON_VALUE_MAX_BYTES
               if set (default unset), and overall depth limited to
               WOLFSENTRY_MAX_JSON_NESTING (default 16) including the 4 parent
               levels
             >

label = DQUOTE < any RFC 8259 JSON-valid string that decodes to at at least 1 and at most 32 octets > DQUOTE

uint32 = < decimal integral number in the range 0...4294967295 >

uint16 = < decimal integral number in the range 0...65535 >

uint8 = < decimal integral number in the range 0...255 >
```

Chapter 4

wolfSentry Release History and Change Log

wolfSentry Release 1.6.2 (January 2, 2024)

Release 1.6.2 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

Noteworthy Changes and Additions

In scripts and Makefile, interpreters (`bash` and `awk`) now follow search `PATH`. Explicit override paths to `bash` and `awk` can be supplied by passing values for `SHELL` and `AWK` to `make`.

Change type of length argument to `wolfsentry_action_res_assoc_by_name()` to `int`, to allow it to accept `WOLFSENTRY_LENGTH_NULL_TERMINATED` (negative number).

Makefile option `STRIPPED` has been split into `STRIPPED` and `FUNCTION_SECTIONS`, the latter directing the compiler and linker to cull any unused object code (with function granularity) to minimize total size.

Bug Fixes, Cleanups, and Debugging Aids

In `handle_route_endpoint_clause()`, add casts to work around an implicit-promotion bug in gcc-7.5.

In `wolfsentry_route_table_max_purgeable_idle_time_get()` and `_set()`, don't use atomic operations, as the context is already locked and the operand is an `int64_t`. This avoids an inadvertent dependency on software __atomic_load_8() and __atomic_store_8() on 32 bit targets.

Various fixes for benign `cppcheck` reports (`duplicateCondition`, `unsignedLessThanZero`, `unreadVariable`, `invalidPrintfArgType_uint`, `invalidPrintfArgType_sint`, `shadowFunction`, `constVariablePointer`, `preprocessorErrorDirective`).

Self-Test Enhancements

Add `replace_rule_transactionally()`, now used in `test_static_routes()` for a thorough workout.

Enhance `freertos-arm32-build-test` target to do two builds, one with and one without `FUNCTION_SECTIONS`, for more thorough coverage.

In `test_lwip()` (`tests/unittests.c`), pass a trivial JSON config to `activate_wolfsentry_lwip()`, to avoid compiler optimizing away `wolfsentry_config_json_oneshot()` and its reverse dependencies.

Split cppcheck-analyze recipe into cppcheck-library, cppcheck-force-library, cppcheck-extras, and cppcheck-force-extras, with increased coverage. Only cppcheck-library and cppcheck-extras are included in the "check-all" dependency list.

wolfSentry Release 1.6.1 (November 18, 2023)

Release 1.6.1 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

New Features

Dynamic rules with nonzero connection counts are now subject to deferred expiration, to assure traffic over established connections is allowed until all connections are closed, even with pauses in traffic flow exceeding the max idle time configured for the rule.

When a rule with a nonzero connection count is deleted, actual deletion is deferred until all connections are closed or the "`max-purgeable-idle-time`" is reached (see below). New success code `WOLFSENTRY_SUCCESS_ID_DEFERRED` is returned in that case. If an identical rule is inserted before the deferred deletion, the existing rule is unmarked for deletion and the insertion call returns another new success code, `WOLFSENTRY_SUCCESS_ID_ALREADY_OK`.

A "`max-purgeable-idle-time`" JSON configuration option has been added, forcing expiration and purge of a zombie dynamic rule even if its current connection count is nonzero. New related APIs are also added: `wolfsentry_route_table_max_purgeable_idle_time_get()`, `wolfsentry_route_table_max_purgeabl` and `wolfsentry_route_purge_time_set()`.

Noteworthy Changes and Additions

A new `FILT_CLOSE_WAIT` event type is added to the lwIP integration patch, and a corresponding `WOLFSENTRY_ACTION_RES_CLOSE_WAIT` result bit is added. Appropriate callbacks are added to lwIP `tcp_process()` and `tcp_receive()`, and the lwIP glue logic now handles mapping from `FILT_CLOSE_WAIT` to `WOLFSENTRY_ACTION_RES_CLOSE_WAIT`.

The lwIP patch has been rebased on upstream 5e3268cf3e (Oct 14 2023), while maintaining compatibility with lwIP 2.1.3-RELEASE.

Bug Fixes, Cleanups, and Debugging Aids

The lwIP patch includes several fixes:

- In `tcp_process()`, when handling passive close and entering `CLOSE_WAIT`, don't `tcp_filter_dispatch_incoming(FILT_CLOSED, ...)` – this happens later, at deallocation.
- Fix `TCP FILT_CLOSED` callbacks to assure accurate interface ID and local_port are passed.

The route/rule system includes several fixes:

- Add error checking to `meta.connection_count` decrement in `wolfsentry_route_event_dispatch_0()`, so that rule churn can never result in count underflow.
- Mask out internal flags (via new macro `WOLFSENTRY_ROUTE_INTERNAL_FLAGS`) from `route_exports->flags` in `wolfsentry_route_init_by_exports()`.
- In `wolfsentry_route_init_by_exports()`, fix pointer math in `memset()` argument to correctly treat `route_exports->private_data_size` as a byte count.
- In `wolfsentry_route_new_by_exports()`, fix check on `route_exports->private_data_size` to properly reflect `config->route_private_data_padding`.
- Add missing implementation of `wolfsentry_route_insert_by_exports()`.
- In `wolfsentry_route_clone()`, fix allocation to use `WOLFSENTRY_MEMALIGN_1()` when `.route_private_data_alignment` is nonzero.
- In `wolfsentry_route_event_dispatch_0()`, don't increment/decrement counts when `WOLFSENTRY_ACTION_RES_FALLTHROUGH`.

In `src/lwip/packet_filter_glue.c`, add `action_results` and `local.sa.interface` to `WOLFSENTRY_DEBUG_LWIP` messages, and add missing gates for `LWIP_IPV6` in `WOLFSENTRY_DEBUG_LWIP` paths.

In `tcp_filter_with_wolfsentry()`, don't set `WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN` for `FILT_REMOTE_RESET`, and fix typo "&event" in call to `wolfsentry_route_event_dispatch_with_inited_result(`

Remove several incorrect calls to `wolfsentry_table_ent_delete_by_id_1()` immediately following failed calls to `wolfsentry_table_ent_insert()` – the former is implicit to the latter.

Self-Test Enhancements

Add to `test_json()` a workout of `connection_count` and deferred deletion dynamics.

`Makefile.analyzers`: add `sanitize-all-NO_POSIX_MEMALIGN-gcc`; tweak `notification-demo-build-tes` to explicitly use the `master` branch of wolfssl.

`Makefile`,`Makefile.analyzers`: tweaks for MacOS X compatibility.

wolfSentry Release 1.6.0 (October 24, 2023)

Release 1.6.0 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

New Features

This release adds native support for the CAN bus address family, and for bitmask-based address matching. CAN addresses and bitmasks are now handled in configuration JSON, as numbers in decimal, octal, or hexadecimal, supporting both 11 bit (part A) and 29 bit (part B) identifiers.

Noteworthy Changes and Additions

`wolfsentry/wolfsentry.h`:

- Add `WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR_BITMASK` and `WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_BITMASK` to `wolfsentry_route_flags_t`.
- Add `WOLFSENTRY_ACTION_RES_USER0`-`WOLFSENTRY_ACTION_RES_USER6` to `wolfsentry_action_res_t` `enum`, add `WOLFSENTRY_ACTION_RES_USER7` macro, and refactor `WOLFSENTRY_ACTION_RES_USER_BASE` as a macro aliased to `WOLFSENTRY_ACTION_RES_USER0`.
- Remove !`WOLFSENTRY_NO_STDIO` gate around `wolfsentry_kv_render_value()`.

`wolfsentry/wolfsentry_settings.h`:

- Rename `WOLFSENTRY_NO_STDIO` to `WOLFSENTRY_NO_STDIO_STREAMS`.
- Rename `WOLFSENTRY_HAVE_NONGNU_ATOMICS` to `WOLFSENTRY_NO_GNU_ATOMICS`.
- Added handling for `WOLFSENTRY_NO_SEM_BUILTIN`, `WOLFSENTRY_NO_ADDR_BITMASK_MATCHING`, and `WOLFSENTRY_NO_IPV6`.
- Gate inclusion of `stdio.h` on !`WOLFSENTRY_NO_STDIO_H`, formerly !`WOLFSENTRY_NO_STDIO`.
- Added `WOLFSENTRY_CONFIG_FLAG_ADDR_BITMASKS`, and rename `WOLFSENTRY_CONFIG_FLAG_NO_STDIO` to `WOLFSENTRY_CONFIG_FLAG_NO_STDIO_STREAMS`.

`src/addr_families.c` and `wolfsentry/wolfsentry_af.h`: Split `WOLFSENTRY_AF_LINK` into `WOLFSENTRY_AF_LINK48` and `WOLFSENTRY_AF_LINK64`, with `WOLFSENTRY_AF_LINK` aliased to `WOLFSENTRY_AF_LINK48`.

`src/kv.c`: remove !`WOLFSENTRY_NO_STDIO` gate around `wolfsentry_kv_render_value()`.

`src/json/load_config.c`: In `convert_sockaddr_address()`, add separate handling for `WOLFSENTRY_AF_LINK48` and `WOLFSENTRY_AF_LINK64`.

`Makefile`:

- Refactor `NO_STDIO`, `NO_JSON`, `NO_JSON_DOM`, `SINGLETHREADED`, `STATIC`, and `STRIPPED` to pivot on definedness, not oneness.
- Add feature flags `NO_ADDR_BITMASK_MATCHING` and `NO_IPV6`.
- Rename feature flag `NO_STDIO` to `NO_STDIO_STREAMS`.

Performance Improvements

src/routes.c: Added AF-mismatch optimization to wolfsentry_route_lookup_0().

Documentation

Add inline documentation for WOLFSENTRY_NO_GETPROTOBY, WOLFSENTRY_SEMAPHORE_INCLUDE, WOLFSENTRY_THREAD_INCLUDE, WOLFSENTRY_THREAD_ID_T, and WOLFSENTRY_THREAD_GET_↩ ID_HANDLER.

doc/json_configuration.md: add documentation and ABNF grammar for "bitmask" node in route endpoints.

Bug Fixes and Cleanups

Fixes for user settings file handling:

- Don't #include <wolfsentry/wolfsentry_options.h> if defined(WOLFSENTRY_↩ USER_SETTINGS_FILE).
- Generate and install wolfsentry/wolfsentry_options.h only if USER_SETTINGS_FILE is undefined, and if USER_SETTINGS_FILE is defined, depend on it where previously the dependency was unconditionally on wolfsentry/wolfsentry_options.h.
- If USER_SETTINGS_FILE is set search it to derive JSON build settings.

Makefile: Don't add -pthread to LDFLAGS if RUNTIME is FreeRTOS-lwIP.

wolfsentry/wolfsentry_settings.h:

- Eliminate inclusion of errno.h – now included only in source files that need it.
- Fix handling for WOLFSENTRY_SEMAPHORE_INCLUDE to give it effect in all code paths (previously ignored in POSIX and FreeRTOS paths).

src/routes.c:

- in wolfsentry_route_event_dispatch_0(), move update of meta.purge_after inside the mutex.
- in wolfsentry_route_get_metadata(), conditionalize use of 64 bit WOLFSENTRY_ATOMIC_LOAD() on pointer size, to avoid dependency on library implementation of __atomic_load_8().

src/wolfsentry_internal.c: fix use-after-free bug in wolfsentry_table_free_ents(), using new table->coupled_ent_fn mechanism.

src/json/load_config.c: In convert_sockaddr_address(), handle sa->addr_len consistently – don't overwrite nonzero values.

src/json/{centijson_dom.c,centijson_sax.c,centijson_value.c}: eliminate direct calls to heap allocator functions in WOLFSENTRY code paths, i.e. use only wolfsentry_allocator.

src/json/centijson_value.c: fix uninited-variable defect on cmp in json_value_dict_get_or↩ _add_().

Self-Test Enhancements

Makefile.analyzers new and enhanced test targets:

- `user-settings-build-test`: construct a user settings file, then build and self-test using it.
- `library-dependency-singlethreaded-build-test` and `library-dependency-multithreaded-buil` : comprehensive check for unexpected unresolved symbols in the library.
- `no-addr-bitmask-matching-test`, `no-ipv6-test`, `linux-lwip-test-no-ipv6`: tests for new feature gates.
- `freertos-arm32-build-test`: newly refactored to perform a final link of `test_lwip` kernel using lwIP and FreeRTOS kernel files and newlib-nano, followed by a check on the size of the kernel.

Added `wolfsentry/wolfssl_test.h`, containing self-test and example logic relocated from `wolfssl/wolfssl/test.↩ h` verbatim.

`tests/test-config*.json`: added several bitmask-matched routes, added several diagnostic events (`"set-user-0"` through `"set-user-4"`), and added no-bitmasks and no-ipv6 variants. Also removed AF-wildcard route from `tests/test-config-numeric.json` to increase test coverage.

`tests/unittests.c`:

- Additional tweaks for portability to 32 bit FreeRTOS
- Add FreeRTOS-specific implementations of `test_lwip()` and `main()`.
- In `test_json()`, add `wolfsentry_addr_family_handler_install(...,"my_AF2",...)`.
- In `test_json()`, add bitmask tests.
- Added stub implementations for various FreeRTOS/newlib dependencies to support final link in `freertos-arm32-build-test` target.

wolfSentry Release 1.5.0 (September 13, 2023)

Release 1.5.0 of the wolfSentry embedded firewall/IDPS has enhancements, additions, and improvements including:

Noteworthy Changes and Additions

In JSON configuration, recognize "events" as equivalent to legacy "events-insert", and "routes" as equivalent to legacy "static-routes-insert". Legacy keys will continue to be recognized.

In the Makefile, FREERTOS_TOP and LWIP_TOP now refer to actual distribution top – previously, FREERTOS_TOP expected a path to the FreeRTOS/Source subdirectory, and LWIP_TOP expected a path to the src subdirectory.

Added public functions wolfsentry_route_default_policy_set() and wolfsentry_route_default_policy_g implicitly accessing the main route table.

Added public functions wolfsentry_get_object_type() and wolfsentry_object_release(), companions to existing wolfsentry_object_checkout() and wolfsentry_get_object_id().

Added wolfsentry_lock_size() to facilitate caller-allocated wolfsentry_rwlocks.

WOLFSENTRY_CONTEXT_ARGS_OUT is now the first argument to utility routines wolfsentry_object_checkout(), wolfsentry_defaultconfig_get(), and wolfsentry_defaultconfig_update(), rather than a bare wolfsentry context pointer.

ports/Linux-lwIP/include/lwipopts.h: Add core locking code.

Removed unneeded routine wolfsentry_config_json_set_default_config().

Improved wolfsentry_kv_render_value() to use json_dump_string() for _KV_STRING rendering, if available, to get JSON-style escapes in output.

Implemented support for user-supplied semaphore callbacks.

Performance Improvements

The critical paths for traffic evaluation have been streamlined by eliminating ephemeral heap allocations, eliminating redundant internal initializations, adding early shortcircuit paths to avoid frivolous processing, and eliminating redundant time lookups and context locking. This results in a 33%-49% reduction in cycles per wolfsentry_route_event_dispatch() on benchmark-test, and a 29%-61% reduction on benchmark-singlethreaded-test, at under 100 cycles for a simple default-policy scenario on a 64 bit target.

Documentation

Added doc/freertos-lwip-app.md, "Building and Initializing wolfSentry for an application on FreeRTOS/lwIP".

Added doc/json_configuration.md, "Configuring wolfSentry using a JSON document".

Doxygen-based annotations are now included in all wolfSentry header files, covering all functions, macros, types, enums, and structures.

The PDF version of the reference manual is now included in the repository and releases at doc/wolfSentry_refman.pdf.

The Makefile now has targets doc-html, doc-pdf, and related targets for generating and cleaning the documentation artifacts.

Bug Fixes and Cleanups

lwip/LWIP_PACKET_FILTER_API.patch has fixes for -Wconversion and -Wshadow warnings.

src/json/centijson_sax.c: Fix bug in json_dump_double() such that floating point numbers were rendered with an extra decimal place.

In wolfsentry_config_json_init_ex(), error if json_config.max_key_len is greater than WOLFSENTRY_MAX_LABEL_BYTES (required for memory safety).

In wolfsentry_config_json_init_ex(), call wolfsentry_defaultconfig_get() to initialize jps->default_config with settings previously passed to wolfsentry_init().

src/kv.c: Fixed _KV_STRING and _KV_BYTES cases in wolfsentry_kv_value_eq_1() (inadvertently inverted memcmp()), and fixed _KV_NONE case to return true.

Fixed wolfsentry_kv_render_value() for _KV_JSON case to pass JSON_DOM_DUMP_PREFERDICTORDER to json_dom_dump().

src/lwip/packet_filter_glue.c: In wolfsentry_install_lwip_filter_callbacks(), if error encountered, disable all callbacks to assure known state on return.

In wolfsentry_init_ex(), correctly convert user-supplied route_idle_time_for_purge from seconds to wolfsentry_time_t.

Pass route_table->default_event to wolfsentry_route_event_dispatch_0() if caller-supplied trigger event is null (changed in wolfsentry_route_event_dispatch_1(), wolfsentry_↩route_event_dispatch_by_id_1(), and wolfsentry_route_event_dispatch_by_route↩_1()).

In wolfsentry_route_lookup_0(), fixed scoping of WOLFSENTRY_ACTION_RES_EXCLUDE_↩REJECT_ROUTES to only check WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED, not WOLFSENTRY_↩ROUTE_FLAG_PORT_RESET.

In wolfsentry_route_delete_0(), properly set WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE.

In wolfsentry_route_event_dispatch_0() and wolfsentry_route_event_dispatch_1(), properly set WOLFSENTRY_ACTION_RES_ERROR at end if ret < 0.

In wolfsentry_route_event_dispatch_1(), properly set WOLFSENTRY_ACTION_RES_↩FALLTHROUGH when route_table->default_policy is used.

Added missing action_results reset to wolfsentry_route_delete_for_filter().

In wolfsentry_lock_init(), properly forbid all inapplicable flags.

Fixed wolfsentry_eventconfig_update_1() to copy over all relevant elements.

Fixed and updated expression for WOLFSENTRY_USER_DEFINED_TYPES.

Self-Test Enhancements

Makefile.analyzers: Added targets test_lwip, minimal-threaded-build-test, pahole-test, route-holes-test, benchmark-test, benchmark-singlethreaded-test, and doc-check.

Implemented tripwires in benchmark-test and benchmark-singlethreaded-test for unexpectedly high cycles/call.

Enlarged coverage of target notification-demo-build-test to run the applications and check for expected and unexpected output.

tests/unittests.c:

- Add test_lwip() with associated helper functions;
- Add WOLFSENTRY_UNITTEST_BENCHMARKS sections in test_static_routes() and test_↩ json();
- Add to test_init() tests of wolfsentry_errcode_source_string() and wolfsentry_errcode_error_s
- Add to test_static_routes() tests of wolfsentry_route_default_policy_set() and wolfsentry_get_object_type(), wolfsentry_object_checkout(), and wolfsentry_object_releas

wolfSentry Release 1.4.1 (July 20, 2023)

Release 1.4.1 of the wolfSentry embedded firewall/IDPS has bug fixes including:

Bug Fixes and Cleanups

Add inline implementations of WOLFSENTRY_ERROR_DECODE_{ERROR_CODE,SOURCE_ID,LINE_↩ NUMBER}() for portable protection from multiple argument evaluation, and refactor WOLFSENTRY_ERROR_ENCODE() and WOLFSENTRY_SUCCESS_ENCODE() to avoid unnecessary dependence on non-portable (gnu-specific) construct.

Use a local stack variable in WOLFSENTRY_ERROR_ENCODE_1() to assure a single evaluation of the argument.

Add -Wno-inline to CALL_TRACE CFLAGS.

Correct the release date of 1.4.0 in ChangeLog.

Self-Test Enhancements

Add CALL_TRACE-test to Makefile.analyzers, and include it in the check-extra dep list.

wolfSentry Release 1.4.0 (July 19, 2023)

Release 1.4.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

New Features

Routes can now be configured to match traffic with designated `action_results` bit constraints, and can be configured to update `action_results` bits, by inserting the route with a parent event that has the desired configuration. Parent events can now also be configured to add or clear route flags for all routes inserted with that parent event.

Added new `aux_event` mechanism to facilitate distinct configurations for a static generator route and the narrower ephemeral routes dynamically created when it is matched.

Added a new built-in action, `"%track-peer-v1"`, that can be used in combination with the above new facilities to dynamically spawn ephemeral routes, allowing for automatic pinhole routes, automatic adversary tracking, and easy implementation of dynamic blocks and/or notifications for port scanning adversaries.

Noteworthy Changes and Additions

Added new APIs `wolfsentry_event_set_aux_event()` and `wolfsentry_event_get_aux_event()`.

Added flag filters and controls to `struct wolfsentry_eventconfig`, and added corresponding clauses to JSON `"config"` sections:

- `.action_res_filter_bits_set`, "action-res-filter-bits-set"
- `.action_res_filter_bits_unset`, `"action-res-filter-bits-unset"`
- `.action_res_bits_to_add`, `"action-res-bits-to-add"`
- `.action_res_bits_to_clear`, `"action-res-bits-to-clear"`
- `.route_flags_to_add_on_insert`, `"route-flags-to-add-on-insert"`
- `.route_flags_to_clear_on_insert`, `"route-flags-to-clear-on-insert"`

Added new `WOLFSENTRY_ACTION_RES_*` (action result) flags to support filtering matches by generic traffic type:

- `WOLFSENTRY_ACTION_RES_SENDING`
- `WOLFSENTRY_ACTION_RES_RECEIVED`
- `WOLFSENTRY_ACTION_RES_BINDING`
- `WOLFSENTRY_ACTION_RES_LISTENING`
- `WOLFSENTRY_ACTION_RES_STOPPED_LISTENING`
- `WOLFSENTRY_ACTION_RES_CONNECTING_OUT`
- `WOLFSENTRY_ACTION_RES_CLOSED`
- `WOLFSENTRY_ACTION_RES_UNREACHABLE`

- WOLFSENTRY_ACTION_RES_SOCK_ERROR

These flags are now passed by the lwIP integration code in src/lwip/packet_filter_glue.c. Detailed descriptions of these and other _ACTION_RES_ bits are in wolfsentry/wolfsentry.h.

Added wolfsentry_addr_family_max_addr_bits(), to allow programmatic determination of whether a given address is a prefix or fully specified.

Added a family of functions to let routes be inserted directly from a prepared struct wolfsentry_route_exports, and related helper functions to prepare it:

- wolfsentry_route_insert_by_exports_into_table()
- wolfsentry_route_insert_by_exports()
- wolfsentry_route_insert_by_exports_into_table_and_check_out()
- wolfsentry_route_insert_by_exports_and_check_out()
- wolfsentry_route_reset_metadata_exports()

Added convenience accessor/validator functions for routes:

- wolfsentry_route_get_addrs()
- wolfsentry_route_check_flags_sensical()

Refactored the event action list implementation so that the various action lists (WOLFSENTRY_ACTION_TYPE_POST, _INSERT, _MATCH, _UPDATE, _DELETE, and _DECISION) are represented directly in the struct wolfsentry_event, rather than through a "subevent". The related APIs (wolfsentry_event_action_prepend(), wolfsentry_event_action_append(), wolfsentry_event_acti wolfsentry_event_action_delete(), wolfsentry_event_action_list_start()) each gain an additional argument, which_action_list. The old JSON grammar is still supported via internal emulation (still tested by test-config.json). The JSON configuration for the new facility is "post-actions", "insert-actions", "match-actions", "update-actions", "delete-actions", and "decision-actions", each optional, and each expecting an array of zero or more actions.

Added a restriction that user-defined action and event labels can't start with "%", and correspondingly, all built-in actions and events have labels that start with "%". This can be overridden by predefining WOLFSENTRY_BUILTIN_LABEL_PREFIX in user settings.

Removed unused flag WOLFSENTRY_ACTION_RES_CONTINUE, as it was semantically redundant relative to WOLFSENTRY_ACTION_RES_STOP.

Removed flags WOLFSENTRY_ACTION_RES_INSERT and WOLFSENTRY_ACTION_RES_DELETE, as the former is superseded by the new builtin action facility, and the latter will be implemented later with another builtin action.

Added flag WOLFSENTRY_ACTION_RES_INSERTED, to indicate when a side-effect route insertion was performed. This flag is now always set by the route insert routines when they succeed. Action plugins must copy this flag as shown in the new wolfsentry_builtin_action_track_peer() to assure proper internal accounting.

Reduced number of available user-defined _ACTION_RESULT_ bits from 16 to 8, to accommodate new generic traffic bits (see above).

In struct wolfsentry_route_metadata_exports, changed .connection_count, .derogatory_count, and .commendable_count, from wolfsentry_hitcount_t to uint16_t, to match internal representations. Similarly, in struct wolfsentry_route_exports, changed .parent_event_label_len from size_t to int to match label_len arg type.

Added wolfsentry_table_ent_get_by_id() to the public API.

Renamed public API wolfsentry_action_res_decode() as wolfsentry_action_res_assoc_by_flag() for clarity and consistency.

Bug Fixes and Cleanups

Consistently set the WOLFSENTRY_ACTION_RES_FALLTHROUGH flag in action_results when dispatch classification (_ACCEPT/_REJECT) was by fallthrough policy.

Refactored internal code to avoid function pointer casts, previously used to allow implementations with struct pointers where a handler pointer has a type that expects void *. The refactored code has shim implementations with fully conformant signatures, that cast the arguments to pass them to the actual implementations. This works around over-eager analysis by the clang UB sanitizer.

Fix missing default cases in non-enum switch() constructs.

Self-Test Enhancements

Added new clauses to test-config*.json for wolfsentry_builtin_action_track_peer() (events "ephemeral-pinhole-parent", "pinhole-generator-parent", "ephemeral-port-scanner-parent", "port-scanner-generator-parent", and related routes), and added full dynamic workout for them to test_json().

Add unit test coverage:

- wolfsentry_event_set_aux_event()
- wolfsentry_event_get_aux_event()
- wolfsentry_event_get_label()
- wolfsentry_addr_family_max_addr_bits()

wolfSentry Release 1.3.1 (July 5, 2023)

Release 1.3.1 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

Bug Fixes and Cleanups

Updated lwIP patches to fix packet_filter_event_t checking on short-enum targets.

Fixed copying of route table header fields (table config) when cloning or rebuilding (preserve default policy etc when loading with WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT | WOLFSENTRY↩_CONFIG_LOAD_FLAG_NO_FLUSH or WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES).

Implemented proper locking in wolfsentry_route_get_reference(), and corresponding lock assertion in wolfsentry_table_cursor_init().

Fixed logic in address matching to properly match zero-length addresses when peforming subnet matching, even if the corresponding _ADDR_WILDCARD flag bit is clear.

Self-Test Enhancements

`Makefile.analyzers`: add `-fshort-enums` variants to `sanitize-all` and `sanitize-all-gcc` recipes, and add `short-enums-test` recipe.

Added `wolfsentry_route_event_dispatch()` cases to `test_json()`.

Added unit test coverage to confirm correct copying of route table header fields when cloning.

wolfSentry Release 1.3 (May 19, 2023)

Release 1.3 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

New Features

Route dump to JSON

The route (rule) table can now be dumped in conformant JSON format to a byte stream, using wolfSentry intrinsics (no `stdio` dependencies), and subsequently reloaded.

- `wolfsentry_route_table_dump_json_start()`, `_next()`, `_end()`
- Byte streams using new `WOLFSENTRY_BYTE_STREAM_*()` macros, with stack and heap options.
- Retryable rendering on `_BUFFER_TOO_SMALL` error, by flushing the byte stream, calling `WOLFSENTRY_BYTE_STREAM_R` and retrying the `wolfsentry_route_table_dump_json_*()` call.
- New flag `WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES`, to allow reloads that leave all event and key-value configuration intact, and only replace the routes.

Bug Fixes and Cleanups

- Non-threadsafe `get{proto,serv}by{name.number}()` calls (already configuration-gated) have been replaced by their `_r()` counterparts, and gated on compatible glibc.
- Fixed an underread bug in `convert_hex_byte()` that affected parsing of MAC addresses.

Self-Test Enhancements

- Added `__wolfsentry_wur` to `WOLFSENTRY_LOCAL`.
- Added new clauses in `test_json()` to verify bitwise idempotency of route table export-ingest cycles to/from JSON.
- Added new target `notification-demo-build-test`.

wolfSentry Release 1.2.2 (May 4, 2023)

Release 1.2.2 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

Noteworthy Changes and Additions

Added C89 pedantic compatibility in core codebase, including unit tests, via `-DWOLFSENTRY_C89`.

Added error code `IO_FAILED`, returned for various stdio failures that previously returned `SYS_OP_FAILED` or went undetected.

Refined `wolfsentry_lock_unlock()` so that final unlock while holding a promotion reservation is not an error and implicitly drops the reservation.

Bug Fixes and Cleanups

Cleanups guided by `clang-tidy` and `cppcheck`: fixed a misused retval from `posix_memalign()`, fixed overwritten retvals in `wolfsentry_lock_unlock()`, and effected myriad cleanups to improve clarity and portability.

Fixed missing assignment of `new->prev` in `wolfsentry_table_clone()`.

Fixed route metadata coherency in transactional configuration updates: add `wolfsentry_route_copy_↩metadata()`, and call it from `wolfsentry_context_exchange()`.

When `wolfsentry_route_event_dispatch*()` results in a default policy fallback, return `USED_↩FALLBACK` success code.

Properly release lock promotion reservation in `wolfsentry_config_json_init_ex()` if obtained.

Fixed several accounting bugs in the lock kernel related to promotion reservations.

Copy `fallthrough_route` pointer in `wolfsentry_route_table_clone_header()`, rather than improperly trying to clone the fallthrough route.

Self-Test Enhancements

Added new global compiler warnings to `Makefile`:

- `-Wmissing-prototypes`
- `-Wdeclaration-after-statement`
- `-Wnested-externs`
- `-Wlogical-not-parentheses`
- `-Wpacked-not-aligned`

Added new targets to `Makefile.analyzers`:

- clang-tidy-build-test
- cppcheck-analyze
- c89-test
- m32-c89-test
- freertos-arm32-c89-build-test
- freertos-arm32-singlethreaded-build-test
- sanitize-aarch64-be-test
- sanitize-all-no-inline-gcc
- no-inline-test
- no-alloca-test
- release-check

Added WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH coverage and an array of should-fail JSON objects to unittests.c:test_json().

Added more arg-not-null and thread-inited checks to thread/lock routines in src/wolfsentry_util.c, and corresponding unit test coverage for all null/uninited arg permutations.

Added assert in release recipe to assure that wolfsentry.h has a version that matches the tagged version.

wolfSentry Release 1.2.1 (Apr 5, 2023)

Release 1.2.1 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

Noteworthy Changes and Additions

Added API wolfsentry_route_render_flags(), now used in wolfsentry_route_render() and wolfsentry_route_exports_render().

Refactored wolfsentry_route_lookup_0() to consistently return the highest-priority matching route, breaking ties using compare_match_exactness().

Added DEBUG_ROUTE_LOOKUP code paths in wolfsentry_route_lookup_0(), for verbose troubleshooting of configurations and internal logic.

Added to convert_hex_byte() (and therefore to MAC address parsing) tolerance for single-hex-digit byte values, as in a:b:c:1:2:3.

Bug Fixes

Removed several inappropriate wildcard flags on queries in lwIP event handlers, particularly `_SA_LOCAL_PORT_WILDCARD` for `FILT_PORT_UNREACHABLE` and `*_INTERFACE_WILDCARD` for `FILT_BINDING`/`FILT_LISTENING`/`FILT_STOP_LISTENING` and when `event->netif` is null.

Added nullness checks for `laddr` and `raddr` in lwIP event handlers, and if null, set all-zeros address.

Refactored wildcard handling in `wolfsentry_route_init()`, `wolfsentry_route_new()`, and `wolfsentry_route_insert_1()`, to zero out wildcard fields at insert time, rather than at init time, so that routes used as targets contain accurate information for `compare_match_exactness()`, regardless of wildcard bits.

Fixed `WOLFSENTRY_VERSION_*` values, which were inadvertently swapped in release 1.2.0.

wolfSentry Release 1.2.0 (Mar 24, 2023)

Production Release 1.2.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

New Features

lwIP full firewall integration

When wolfSentry is built with make options `LWIP=1 LWIP_TOP=<path-to-lwIP-source>`, the library is built with new APIs `wolfsentry_install_lwip_filter_ethernet_callback()`, `wolfsentry_install_lwip_filter_ip_callbacks()`, `wolfsentry_install_lwip_filter_icmp_callba`
`wolfsentry_install_lwip_filter_tcp_callback()`, `wolfsentry_install_lwip_filter_udp_callbac`
and the all-on-one `wolfsentry_install_lwip_filter_callbacks()`. For each layer/protocol, a simple bitmask, of type `packet_filter_event_mask_t`, allows events to be selectively filtered, with other traffic passed with negligible overhead. For example, TCP connection requests can be fully evaluated by wolfSentry, while traffic within established TCP connections can pass freely.

`wolfSentry LWIP=1` relies on a patchset to lwIP, gated on the macro `LWIP_PACKET_FILTER_API`, that adds generic filter callback APIs to each layer and protocol. See `lwip/README.md` for details.

In addition to `LWIP_DEBUG` instrumentation, the new integration supports `WOLFSENTRY_DEBUG_PACKET_FILTER`, which renders the key attributes and outcome for all callout events.

Noteworthy Changes and Additions

Routes and default actions can now be annotated to return `WOLFSENTRY_ACTION_RES_PORT_RESET` in their `action_results`. This is used in the new lwIP integration to control whether TCP reset and ICMP port-unreachable packets are sent (versus dropping the rejected packet unacknowledged).

A new `ports/` tree is added, and the former FreeRTOS/ tree is moved to `ports/FreeRTOS-lwIP`.

New helper macros are added for managing thread state: `WOLFSENTRY_THREAD_HEADER_DECLS`, `WOLFSENTRY_THREAD_HEADER_INIT()`, `WOLFSENTRY_THREAD_HEADER_INIT_CHECKED()`.

New flags `WOLFSENTRY_ROUTE_FLAG_PORT_RESET` and `WOLFSENTRY_ACTION_RES_EXCLUDE_REJECT_ROUTES` to support firewall functionalities.

Bug Fixes

Wildcard matching in the routes/rules table now works correctly even for non-contiguous wildcard matching.

`struct wolfsentry_sockaddr` now aligns its `addr` member to a 4 byte boundary, for safe casting to `(int *)`, using a new `attr_align_to()` macro.

The route lookup algorithm has been improved for correct results with non-contiguous wildcards, to correctly break ties using the new `compare_match_exactness()`, and to correctly give priority to routes with a matching event.

When matching target routes (e.g. with `wolfsentry_route_event_dispatch()`), ignore failure in `wolfsentry_event_get_reference()` if `WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD` is set in the `flags`.

wolfSentry Release 1.1.0 (Feb 23, 2023)

Production Release 1.1.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

New Features

Internal settings, types, alignments, constants, a complete set of internal shims, and Makefile clauses, for portability to native FreeRTOS with threads on 32 bit gcc targets.

Noteworthy Changes and Additions

rwlock control contexts can now be allocated inside interrupt handlers, and `WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAPHORE` can be supplied to the new `wolfsentry_context_lock_mutex_timed_ex()`, allowing safe trylock followed by automatic lock recursion.

API routines are now marked warn-unused-return by default, subject to user-defined override. This new default warns on untrapped errors, to aid preventing undefined behavior.

API arguments previously accepting "long" ints for counts of seconds now expect `time_t`, for portability to ARM32 and FreeRTOS.

New unit test: `test_json_corpus`, for highly configurable bulk trial runs of the JSON processing subsystem.

New tests in `Makefile.analyzers`: `no-getprotoby-test`, `freertos-arm32-build-test`.

A new guard macro, `WOLFSENTRY_NO_GETPROTOBY`, allows narrow elimination of dependencies on `getprotobyname()` and `getprotobynumber()`.

Recursive JSON DOM tree processing logic was refactored to greatly reduce stack burden.

Substantial enlargement of code coverage by unit tests, guided by `gcov`.

New convenience macros for typical threaded state tracking wrappers: `WOLFSENTRY_THREAD_HEADER_CHECKED()` and `WOLFSENTRY_THREAD_TAILER_CHECKED()`.

Bug Fixes

Cloning of user-defined deep JSON objects is now implemented, as needed for configuration load dry runs and load-then-commit semantics.

JSON processing of UTF-8 surrogate pairs is now fixed.

Fixed retval testing in `wolfsentry_action_list_{append,prepend,insert}_1()`, and added missing `point_action` lookup in `wolfsentry_action_list_insert_after()`.

Fixed potential use-after-free defect in `wolfsentry_event_delete()`.

wolfSentry Release 1.0.0 (Jan 18, 2023)

Production Release 1.0.0 of the wolfSentry embedded firewall/IDPS has bug fixes and improvements including:

Noteworthy Changes and Additions

- Makefile improvements around `wolfsentry_options.h`, and a new com-bundle rule.
- A new macro `WOLFSENTRY_USE_NONPOSIX_THREADS`, separated from `WOLFSENTRY_USE_NONPOSIX_SEMAPHORES`, supporting mixed-model targets, e.g. Mac OS X.

Bug Fixes

- In `examples/notification-demo/log_server/log_server.c`, in `main()`, properly reset `transaction_successful` at top of the accept loop.

wolfSentry Release 0.8.0 (Jan 6, 2023)

Preview Release 0.8.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Features

Multithreaded application support

- Automatic locking on API entry, using a high performance, highly portable semaphore-based readwrite lock facility, with error checking and opportunistic lock sharing.
- Thread-specific deadlines set by the caller, limiting waits for lock acquisition as needed for realtime applications.
- A mechanism for per-thread private data, accessible to user plugins.
- No dependencies on platform-supplied thread-local storage.

Updated Examples

examples/notification-demo

- Add interrupt handling for clean error-checked shutdown in `log_server`.
- Add `/kill-server` admin command to `log_server`.
- Reduce penalty-box-duration in `notify-config.{json,h}` to 10s for demo convenience.

Noteworthy Changes and Additions

- A new first argument to `wolfsentry_init_ex()` and `wolfsentry_init()`, `caller_build_settings`, for runtime error-checking of application/library compatibility. This mechanism will also allow future library changes to be conditionalized on caller version and/or configuration expectations as needed, often avoiding the need for application recompilation.
- `src/util.c` was renamed to `src/wolfsentry_util.c`.
- `wolfsentry/wolfsentry_settings.h` was added, containing setup code previously in `wolfsentry/wolfsentry.h`.
- Error IDs in `enum wolfsentry_error_id` are all now negative, and a new `WOLFSENTRY_SUCCESS_ID_*` namespace was added, with positive values and supporting macros.

New public utility APIs, macros, types, etc.

- `WOLFSENTRY_VERSION_*` macros, for version testing
- `wolfsentry_init_thread_context()`, `wolfsentry_alloc_thread_context()`, `wolfsentry_get_thread_id()`, `wolfsentry_get_thread_user_context()`, `wolfsentry_get_threa` `wolfsentry_get_thread_flags()`, `wolfsentry_destroy_thread_context()`, `wolfsentry_free_th` `wolfsentry_set_deadline_rel_usecs()`, `wolfsentry_set_deadline_abs()`, `wolfsentry_clear_d` `wolfsentry_set_thread_readonly()`, `wolfsentry_set_thread_readwrite()`
- `WOLFSENTRY_DEADLINE_NEVER` and `WOLFSENTRY_DEADLINE_NOW`, used internally and for testing values returned by `wolfsentry_get_thread_deadline()`
- Many new values in the `WOLFSENTRY_LOCK_FLAG_*` set.
- `wolfsentry_lock_*()` APIs now firmed, and new `wolfsentry_context_lock_shared_with_reservation`
- `WOLFSENTRY_CONTEXT_*` helper macros.
- `WOLFSENTRY_UNLOCK_*()`, `WOLFSENTRY_SHARED_*()`, `WOLFSENTRY_MUTEX_*()`, and `WOLFSENTRY_PROMOTABLE_*()` helper macros
- `WOLFSENTRY_ERROR_UNLOCK_AND_RETURN()`, `WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN()`, and related helper macros.

Bug Fixes

- Various fixes, and additional hardening and cleanup, in the readwrite lock kernel.
- Various fixes in `Makefile`, for proper handling and installation of `wolfsentry_options.h`.

wolfSentry Release 0.7.0 (Nov 7, 2022)

Preview Release 0.7.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Features

Support for freeform user-defined JSON objects in the "user-values" (key-value pair) section of the config package.

- Uses syntax `"key" : { "json" : x }` where `x` is any valid standalone JSON expression.
- Key length limited to `WOLFSENTRY_MAX_LABEL_BYTES` by default.
- String length limited to `WOLFSENTRY_KV_MAX_VALUE_BYTES` by default.
- JSON tree depth limited to `WOLFSENTRY_MAX_JSON_NESTING` by default.
- All default limits subject to caller runtime override using the `json_config` arg to the new APIs `wolfsentry_config_json_init_ex()` and `wolfsentry_config_json_oneshot_ex()`, accepting a `JSON_CONFIG *` (accepted as `const`).

New APIs for JSON KVs

- `wolfsentry_user_value_store_json()`
- `wolfsentry_user_value_get_json()`
- `WOLFSENTRY_KV_V_JSON()`
- `wolfsentry_config_json_init_ex()`
- `wolfsentry_config_json_oneshot_ex()`

New config load flags controlling JSON KV parsing

- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST`
- `WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER`

Support for setting a user KV as read-only.

- Read-only KVs can't be deleted or overwritten without first setting them read-write.
- Mechanism can be used to protect user-configured data from dynamic changes by JSON configuration package – JSON cannot change or override the read-only bit.

KV mutability APIs:

- `wolfsentry_user_value_set_mutability()`
- `wolfsentry_user_value_get_mutability()`

Updated Examples

examples/notification-demo

- Update and clean up `udp_to_dbus`, and add `--kv-string` and `--kv-int` command line args for runtime ad hoc config overrides.
- Rename config node controlling the `udp_to_dbus` listen address from "notification-dest-addr" to "notification-listen-addr".

Added examples/notification-demo/log_server

- Toy embedded web server demonstrating HTTPS with dynamic insertion of limited-lifespan wolfSentry rules blocking (penalty boxing) abusive peers.
- Demonstrates mutual authentication using TLS, and role-based authorizations pivoting on client certificate issuer (certificate authority).

Noteworthy Changes and Additions

- JSON strings (natively UTF-8) are now consistently passed in and out with `unsigned char` pointers.
- `wolfsentry_kv_render_value()` now has a `struct wolfsentry_context *` as its first argument (necessitated by addition of freeform JSON rendering).
- Added new API routine `wolfsentry_centijson_errcode_translate()`, allowing conversion of all CentiJSON return codes (e.g. from `json_dom_parse()`, `json_value_path()`, and `json_value_build_path()`) from native CentiJSON to roughly-corresponding native wolfSentry codes.

Cleanup of JSON DOM implementation

- Added `json_` prefix to all JSON functions and types.
- CentiJSON now uses wolfSentry configured allocator for all heap operations.

New utility APIs

- `wolfsentry_get_allocator()`
- `wolfsentry_get_timecbs()`

Bug Fixes

- Fix error-path memory leak in JSON KV handling.
- Fix "echo: write error: Broken pipe" condition in recipe for rule "force"
- Various minor portability fixes.
- Enlarged scope for build-time pedantic warnings – now includes all of CentiJSON.

wolfSentry Release 0.6.0 (Sep 30, 2022)

Preview Release 0.6.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Features

Core support for automatic penalty boxing, with configurable threshold when derogatory count reaches threshold

New APIs for manipulating route derogatory/commendable counts from application/plugin code:

- `wolfsentry_route_increment_derogatory_count()`
- `wolfsentry_route_increment_commendable_count()`
- `wolfsentry_route_reset_derogatory_count()`
- `wolfsentry_route_reset_commendable_count()`

New JSON config nodes:

- `derog-thresh-for-penalty-boxing`
- `derog-thresh-ignore-commendable`
- `commendable-clears-derogatory`

Automatic purging of expired routes:

- constant time garbage collection
- `wolfsentry_route_table_max_purgeable_routes_get()`
- `wolfsentry_route_table_max_purgeable_routes_set()`
- `wolfsentry_route_stale_purge_one()`

Noteworthy Changes and Additions

- New API `wolfsentry_route_insert_and_check_out()`, allowing efficient update of route state after insert; also related new API `wolfsentry_object_checkout()`.
- New APIs `wolfsentry_route_event_dispatch_by_route()` and `wolfsentry_route_event_dispatch_` analogous to the `_by_id()` variants, but accepting a struct wolfsentry_route pointer directly.
- `wolfsentry_route_init()` and `wolfsentry_route_new()` now allow (and ignore) nonzero supplied values in wildcarded wolfsentry_sockaddr members.
- New debugging aid, make CALL_TRACE=1, gives full call stack trace with codepoints and error codes, to aid debugging of library, plugins, and configurations.

Bug Fixes

- src/internal.c: fix wrong constant of iteration in `wolfsentry_table_ent_get_by_id()`.

wolfSentry Release 0.5.0 (Aug 1, 2022)

Preview Release 0.5.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Example

examples/notification-demo

Added examples/notification-demo, demonstrating plugin actions, JSON event representation, and pop-up messages using the D-Bus notification facility and a middleware translation daemon.

Noteworthy Changes

- Added new API `wolfsentry_init_ex()` with `wolfsentry_init_flags_t` argument.
- Added runtime error-checking on lock facility.

Bug Fixes

Fix missing assignment in `wolfsentry_list_ent_insert_after()`.

wolfSentry Release 0.4.0 (May 27, 2022)

Preview Release 0.4.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Features

- User-defined key-value pairs in JSON configuration: allows user plugins to access custom config parameters in the wolfSentry config using the new `wolfsentry_user_value_*()` family of API functions. Binary configuration data can be supplied in the configuration using base64 encoding, and are decoded at parse time and directly available to user plugins in the original raw binary form. The key-value facility also supports a custom validator callback to enforce constraints on user-defined config params in the JSON.
- User-defined address families: allows user plugins for custom address families and formats, using new `wolfsentry_addr_family_*()` API routines. This allows idiomatic formats for non-Internet addresses in the JSON config, useful for various buses and device namespaces.
- Formalization of the concepts of default events and fallthrough rules in the route tables.
- A new subevent action list facility to support logging and notifications around the final decisions of the rule engine, alongside the existing subevents for rule insertions, matches, and deletions.
- The main plugin interface (`wolfsentry_action_callback_t`) now passes two separate routes, a "`trigger_route`" with full attributes of the instant traffic, and a "`rule_route`" that matches that traffic. In dynamic rule scenarios, plugins can manipulate the passed `rule_route` and set the `WOLFSENTRY_↩ ACTION_RES_INSERT` bit in the to define a new rule that will match the traffic thereafter. All actions in the chain retain readonly access to the unmodified trigger route for informational purposes.
- The JSON DOM facility from CentiJSON is now included in the library by default (disabled by make `NO_↩ JSON_DOM=1`), layered on the SAX facility used directly by the wolfSentry core to process the JSON config package. The DOM facility can be used as a helper in user plugins and applications, for convenient JSON parsing, random access, and production.

Noteworthy Changes

- In the JSON config, non-event-specific members of top level node "config-update" node have been moved to the new top level node "default-policies", which must appear after "event-insert". "default-policies" members are "default-policy-static", "default-policy-dynamic", "default-event-static", and "default-event-dynamic".

Bug Fixes

- In `wolfsentry_config_json_init()`, properly copy the load_flags from the caller into the `_json↩ _process_state`.
- The JSON SAX API routines (`wolfsentry/centijson_sax.h`) are now properly exported.

wolfSentry Release 0.3.0 (Dec 30, 2021)

Preview Release 0.3.0 of the wolfSentry embedded firewall/IDPS has bug fixes and new features including:

New Ports and Examples

examples/Linux-LWIP

This demo uses Linux-hosted LWIP in Docker containers to show packet-level and connection-level filtering using wolfSentry. Filtering can be by MAC, IPv4, or IPv6 address. Demos include pre-accept TCP filtering, and filtering of ICMP packets.

See examples/Linux-LWIP/README.md for the installation and usage guide, and examples/Linux-LWIP/echo-config.json for the associated wolfSentry configuration.

FreeRTOS with LWIP on STM32

This demo is similar to Linux-LWIP, but targets the STM32 ARM core and the STM32CubeMX or STM32Cube↩ IDE toolchain, with a FreeRTOS+LWIP runtime. It shows wolfSentry functionality in a fully embedded (bare metal) application.

See examples/STM32/README.md for the installation and usage guide, and examples/STM32/Src/sentry.c for the compiled-in wolfSentry configuration.

New Features

- Autogeneration and inclusion of `wolfsentry_options.h`, synchronizing applications with wolfSentry library options as built.
- New APIs `wolfsentry_route_event_dispatch_[by_id]with_inited_result()`, for easy caller designation of known traffic attributes, e.g. `WOLFSENTRY_ACTION_RES_CONNECT` or `WOLFSENTRY_ACTION_RES_DISCONNECT`.
- Efficient support for aligned heap allocations on targets that don't have a native aligned allocation API: `wolfsentry_free_aligned_cb_t`, `wolfsentry_allocator.free_aligned`, `wolfsentry_builtin_free_aligned()`, `wolfsentry_free_aligned()`, and `WOLFSENTRY↩ _FREE_ALIGNED()`.
- Semaphore wrappers for FreeRTOS, for use by the `wolfsentry_lock_*()` shareable-upgradeable lock facility.

Bug Fixes

- `wolfsentry_route_event_dispatch_1()`: don't impose `config.penaltybox_duration` on routes with `route->meta.last_penaltybox_time == 0`.
- trivial fixes for backward compat with gcc-5.4.0, re `-Wconversion` and `-Winline`.

Please send questions or comments to `douzzer@wolfssl.com`

Chapter 5

Topic Index

5.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 8

Topic Documentation

8.1 Core Types and Macros

Macros

- #define **WOLFSENTRY_NO_ALLOCA**

  Build flag to use only implementations that avoid alloca().
- #define **WOLFSENTRY_C89**

  Build flag to use only constructs that are pedantically legal in C89.
- #define **__attribute_maybe_unused__**

  Attribute abstraction to mark a function or variable (typically a `static`) as possibly unused.
- #define **DO_NOTHING**

  Statement-type abstracted construct that executes no code.
- #define **WOLFSENTRY_NO_POSIX_MEMALIGN**

  Define if `posix_memalign()` is not available.
- #define **WOLFSENTRY_FLEXIBLE_ARRAY_SIZE**

  Value appropriate as a size for an array that will be allocated to a variable size. Built-in value usually works.
- #define **SIZET_FMT**

  printf-style format string appropriate for pairing with `size_t`
- #define **WOLFSENTRY_ENT_ID_FMT**

  printf-style format string appropriate for pairing with wolfsentry_ent_id_t
- #define **WOLFSENTRY_ENT_ID_NONE**

  always-invalid object ID
- #define **WOLFSENTRY_HITCOUNT_FMT**

  printf-style format string appropriate for pairing with wolfsentry_hitcount_t
- #define **__wolfsentry_wur**

  abstracted attribute designating that the return value must be checked to avoid a compiler warning
- #define **wolfsentry_static_assert**(c)

  abstracted static assert – `c` must be true, else `c` is printed
- #define **wolfsentry_static_assert2**(c, m)

  abstracted static assert – `c` must be true, else `m` is printed
- #define **WOLFSENTRY_API_VOID**

  Function attribute for declaring/defining public void API functions.
- #define **WOLFSENTRY_API**

  Function attribute for declaring/defining public API functions with return values.
- #define **WOLFSENTRY_LOCAL_VOID**

Function attribute for declaring/defining private void functions.

- #define **WOLFSENTRY_LOCAL**

  Function attribute for declaring/defining private functions with return values.

- #define **WOLFSENTRY_MAX_ADDR_BYTES** 16

  The maximum size allowed for an address, in bytes. Can be overridden. Note that support for bitmask matching for an address family depends on WOLFSENTRY_MAX_ADDR_BYTES at least twice the max size of a bare address in that family, as the address and mask are internally stored as a single double-length byte vector. Note also that WOLFSENTRY_MAX_ADDR_BYTES entails proportional overhead if wolfSentry is built WOLFSENTRY_NO_ALLOCA or WOLFSENTRY_C89.

- #define **WOLFSENTRY_MAX_ADDR_BITS** (WOLFSENTRY_MAX_ADDR_BYTES∗8)

  The maximum size allowed for an address, in bits. Can be overridden.

- #define **WOLFSENTRY_MAX_LABEL_BYTES** 32

  The maximum size allowed for a label, in bytes. Can be overridden.

- #define **WOLFSENTRY_BUILTIN_LABEL_PREFIX** "%"

  The prefix string reserved for use in names of built-in actions and events.

- #define **WOLFSENTRY_KV_MAX_VALUE_BYTES** 16384

  The maximum size allowed for scalar user-defined values. Can be overridden.

Typedefs

- typedef unsigned char **byte**

  8 bits unsigned

- typedef uint16_t **wolfsentry_addr_family_t**

  integer type for holding address family number

- typedef uint16_t **wolfsentry_proto_t**

  integer type for holding protocol number

- typedef uint16_t **wolfsentry_port_t**

  integer type for holding port number

- typedef uint32_t **wolfsentry_ent_id_t**

  integer type for holding table entry ID

- typedef uint16_t **wolfsentry_addr_bits_t**

  integer type for address prefix lengths (in bits)

- typedef uint32_t **wolfsentry_hitcount_t**

  integer type for holding hit count statistics

- typedef int64_t **wolfsentry_time_t**

  integer type for holding absolute and relative times, using microseconds in built-in implementations.

- typedef uint16_t **wolfsentry_priority_t**

  integer type for holding event priority (smaller number is higher priority)

8.1.1 Detailed Description

8.2 Startup/Configuration/Shutdown Subsystem

Data Structures

- struct wolfsentry_host_platform_interface

  struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores

- struct wolfsentry_build_settings

  struct for passing the build version and configuration

Macros

- #define **WOLFSENTRY_VERSION_MAJOR**

  Macro for major version number of installed headers.
- #define **WOLFSENTRY_VERSION_MINOR**

  Macro for minor version number of installed headers.
- #define **WOLFSENTRY_VERSION_TINY**

  Macro for tiny version number of installed headers.
- #define **WOLFSENTRY_VERSION_ENCODE**(major, minor, tiny)

  Macro to convert a wolfSentry version to a single integer, for comparison to other similarly converted versions.
- #define **WOLFSENTRY_VERSION**

  The version recorded in wolfsentry.h, encoded as an integer.
- #define **WOLFSENTRY_VERSION_GT**(major, minor, tiny)

  Helper macro that is true if the given version is greater than that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_GE**(major, minor, tiny)

  Helper macro that is true if the given version is greater than or equal to that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_EQ**(major, minor, tiny)

  Helper macro that is true if the given version equals that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_LT**(major, minor, tiny)

  Helper macro that is true if the given version is less than that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_LE**(major, minor, tiny)

  Helper macro that is true if the given version is less than or equal to that in wolfsentry.h.
- #define **WOLFSENTRY_MAX_JSON_NESTING** 16

  Can be overridden.
- #define **WOLFSENTRY_USER_SETTINGS_FILE** "the_path"

  Define to the path of a user settings file to be included, containing extra and override definitions and directives. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.
- #define **WOLFSENTRY_NO_INTTYPES_H**

  Define to inhibit inclusion of `inttypes.h` (alternative `typedef`s or `include` must be supplied with WOLFSENTRY_USER_SETTINGS_FILE).
- #define **WOLFSENTRY_NO_STDINT_H**

  Define to inhibit inclusion of `stding.h` (alternative `typedef`s or `include` must be supplied with WOLFSENTRY_USER_SETTINGS_FILE).
- #define **WOLFSENTRY_SINGLETHREADED**

  Define to disable all thread handling and safety in wolfSentry.
- #define **WOLFSENTRY_USE_NONPOSIX_SEMAPHORES**

  Define if POSIX semaphore API is not available. If no non-POSIX builtin implementation is present in wolfsentry_util.c, then WOLFSENTRY_NO_SEM_BUILTIN must be set, and the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include a full semaphore implementation (shim set) in its wolfsentry_semcbs slot.
- #define **WOLFSENTRY_USE_NONPOSIX_THREADS**

  Define if POSIX thread API is not available. `WOLFSENTRY_THREAD_INCLUDE`, `WOLFSENTRY_THREAD_ID_T`, and `WOLFSENTRY_THREAD_GET_ID_HANDLER` will need to be supplied in WOLFSENTRY_USER_SETTINGS_FILE.
- #define **WOLFSENTRY_NO_GNU_ATOMICS**

  Define if gnu-style atomic intrinsics are not available. `WOLFSENTRY_ATOMIC_*()` macro definitions for intrinsics will need to be supplied in WOLFSENTRY_USER_SETTINGS_FILE (see wolfsentry_util.h).
- #define **WOLFSENTRY_NO_CLOCK_BUILTIN**

  If defined, omit built-in time primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_timecbs.
- #define **WOLFSENTRY_NO_SEM_BUILTIN**

  If defined, omit built-in semaphore primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_semcbs.
- #define **WOLFSENTRY_NO_MALLOC_BUILTIN**

If defined, omit built-in heap allocator primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_allocator.

- #define **WOLFSENTRY_NO_ERROR_STRINGS**

  If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
- #define **WOLFSENTRY_NO_PROTOCOL_NAMES**

  If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.
- #define **WOLFSENTRY_NO_ADDR_BITMASK_MATCHING**

  If defined, omit support for bitmask matching of addresses, and support only prefix matching.
- #define **WOLFSENTRY_NO_IPV6**

  If defined, omit support for IPv6.
- #define **WOLFSENTRY_MAX_BITMASK_MATCHED_AFS**

  The maximum number of distinct address families that can use bitmask matching in routes. Default value is 4.
- #define **WOLFSENTRY_NO_GETPROTOBY**

  Define this to gate out calls to getprotobyname_r() and getservbyname_r(), necessitating numeric identification of protocols (e.g. 6 for TCP) and services (e.g. 25 for SMTP) in configuration JSON documents.
- #define **WOLFSENTRY_SEMAPHORE_INCLUDE** "the_path"

  Define to the path of a header file declaring a semaphore API. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.
- #define **WOLFSENTRY_THREAD_INCLUDE** "the_path"

  Define to the path of a header file declaring a threading API. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.
- #define **WOLFSENTRY_THREAD_ID_T** thread_id_type

  Define to the appropriate type analogous to POSIX `pthread_t`.
- #define **WOLFSENTRY_THREAD_GET_ID_HANDLER** pthread_self_ish_function

  Define to the name of a void function analogous to POSIX `pthread_self`, returning a value of type WOLFSENTRY_THREAD_ID_T.
- #define **WOLFSENTRY_CONFIG_SIGNATURE**

  Macro to use as the initializer for wolfsentry_build_settings.config and wolfsentry_host_platform_interface.caller_build_settings.

Typedefs

- typedef void(∗ **wolfsentry_cleanup_callback_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, void ∗cleanup_↩ arg)

  Function type to pass to wolfsentry_cleanup_push()
- typedef uint32_t **wolfsentry_config_load_flags_t**

  Type for holding flag bits from wolfsentry_config_load_flags.

Enumerations

- enum wolfsentry_init_flags_t {
  WOLFSENTRY_INIT_FLAG_NONE ,
  WOLFSENTRY_INIT_FLAG_LOCK_SHARED_ERROR_CHECKING }

  flags to pass to wolfsentry_init_ex(), to be `OR`d together.
- enum wolfsentry_clone_flags_t {
  WOLFSENTRY_CLONE_FLAG_NONE ,
  WOLFSENTRY_CLONE_FLAG_AS_AT_CREATION ,
  WOLFSENTRY_CLONE_FLAG_NO_ROUTES }

  Flags to be `OR`d together to control the dynamics of wolfsentry_context_clone() and other cloning functions.

- enum wolfsentry_config_load_flags {
  WOLFSENTRY_CONFIG_LOAD_FLAG_NONE ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_FINI }

  Flags to be `OR`d together to communicate options to wolfsentry_config_json_init()

Functions

- WOLFSENTRY_API struct wolfsentry_build_settings **wolfsentry_get_build_settings** (void)

  Return the `wolfsentry_build_settings` of the library as built.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_build_settings_compatible** (struct wolfsentry_build_settings caller_build_settings)

  Return success if the application and library were built with mutually compatible wolfSentry version and configuration.
- WOLFSENTRY_API struct wolfsentry_host_platform_interface * **wolfsentry_get_hpi** (struct wolfsentry_↩ context *wolfsentry)

  Return a pointer to the `wolfsentry_host_platform_interface` associated with the supplied `wolfsentry_context`, mainly for passing to `wolfsentry_alloc_thread_context()`, `wolfsentry_free_thread_c` `wolfsentry_lock_init()`, and `wolfsentry_lock_alloc()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_push** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_cleanup_callback_t handler, void *arg)

  Register `handler` to be called at shutdown with arg `arg`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_pop** (WOLFSENTRY_CONTEXT_ARGS_IN, int execute_p)

  Remove the most recently registered and unpopped handler from the cleanup stack, and if `execute_p` is nonzero, call it with the `arg` with which it was registered.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_all** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Iteratively call wolfsentry_cleanup_pop(), executing each handler as it is popped, passing it the `arg` with which it was registered.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_init_ex** (struct wolfsentry_build_settings caller_↩ build_settings, WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi), const struct wolfsentry_eventconfig *config, struct wolfsentry_context **wolfsentry, wolfsentry_init_flags_t flags)

  Variant of wolfsentry_init() that accepts a `flags` argument, for additional control over configuration.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init (struct wolfsentry_build_settings caller_build_↩ settings, WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi), const struct wolfsentry_eventconfig *config, struct wolfsentry_context **wolfsentry)

  Allocates and initializes the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_get (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_eventconfig *config)

  Get the default config from a wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_update (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_eventconfig *config)

  Updates mutable fields of the default config (all but wolfsentry_eventconfig::route_private_data_size and wolfsentry_eventconfig::route_private_data_alignment)
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_flush (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flushes the route, event, and user value tables from the wolfsentry context.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_free (WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry))

  Frees the wolfsentry context and the tables within it. The wolfsentry context will be a pointer to NULL upon success.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_shutdown (WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry))

  Shut down wolfSentry, freeing all resources. Gets an exclusive lock on the context, then calls wolfsentry_context_free().
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_inhibit_actions (WOLFSENTRY_CONTEXT_ARGS_IN)

  Disable automatic dispatch of actions on the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_enable_actions (WOLFSENTRY_CONTEXT_ARGS_IN)

  Re-enable automatic dispatch of actions on the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_clone (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_context **clone, wolfsentry_clone_flags_t flags)

  Clones a wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_exchange (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_context *wolfsentry2)

  Swaps information between two wolfsentry contexts.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_centijson_errcode_translate** (wolfsentry_errcode_t centijson_errcode)

  Convert CentiJSON numeric error code to closest-corresponding wolfSentry error code.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_init** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_config_load_flags_t load_flags, struct wolfsentry_json_process_state **jps)

  Allocate and initialize a `struct wolfsentry_json_process_state` with the designated `load_flags`, to subsequently pass to `wolfsentry_config_json_feed()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_init_ex** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_config_load_flags_t load_flags, const JSON_CONFIG *json_config, struct wolfsentry_json_↩ process_state **jps)

  Variant of `wolfsentry_config_json_init()` with an additional `JSON_CONFIG` argument, `json_↩ config`, for tailoring of JSON parsing dynamics.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_feed** (struct wolfsentry_json_process↩ _state *jps, const unsigned char *json_in, size_t json_in_len, char *err_buf, size_t err_buf_size)

  Pass a segment of JSON configuration into the parsing engine. Segments can be as short or as long as desired, to facilitate incremental read-in.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_centijson_errcode** (struct wolfsentry_json↩ _process_state *jps, int *json_errcode, const char **json_errmsg)

  Copy the current error code and/or human-readable error message from a `struct wolfsentry_json_↩ process_state` allocated by `wolfsentry_config_json_init()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_fini** (struct wolfsentry_json_process↩ _state **jps, char *err_buf, size_t err_buf_size)

  To be called when done iterating `wolfsentry_config_json_feed()`, completing the configuration load.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_oneshot** (WOLFSENTRY_CONTEXT_ARGS_IN, const unsigned char *json_in, size_t json_in_len, wolfsentry_config_load_flags_t load_flags, char *err_buf, size_t err_buf_size)

  Load a complete JSON configuration from an in-memory buffer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_oneshot_ex** (WOLFSENTRY_CONTEXT_ARGS_IN, const unsigned char *json_in, size_t json_in_len, wolfsentry_config_load_flags_t load_flags, const JSON_CONFIG *json_config, char *err_buf, size_t err_buf_size)

  Variant of `wolfsentry_config_json_oneshot()` with an additional `JSON_CONFIG` argument, `json_↩ config`, for tailoring of JSON parsing dynamics.

8.2.1 Detailed Description

8.2.2 Enumeration Type Documentation

8.2.2.1 wolfsentry_clone_flags_t

`enum wolfsentry_clone_flags_t`

Flags to be ORd together to control the dynamics of wolfsentry_context_clone() and other cloning functions.

Enumerator

WOLFSENTRY_CLONE_FLAG_NONE	Default behavior.
WOLFSENTRY_CLONE_FLAG_AS_AT_CREATION	Don't copy routes, events, or user values, and copy default config as it existed upon return from `wolfsentry_init()`. Action and address family tables are copied as usual.
WOLFSENTRY_CLONE_FLAG_NO_ROUTES	Don't copy route table entries. Route table config, default config, and all other tables, are copied as usual.

8.2.2.2 wolfsentry_config_load_flags

`enum wolfsentry_config_load_flags`

Flags to be ORd together to communicate options to wolfsentry_config_json_init()

Enumerator

WOLFSENTRY_CONFIG_LOAD_FLAG_NONE	Default behavior.
WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH	Add to current configuration, rather than replacing it.
WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN	Test the load operation, as modified by other flags, without updating current configuration.
WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_↩ THEN_COMMIT	Test the load operation before replacing the current configuration.
WOLFSENTRY_CONFIG_LOAD_FLAG_NO_↩ ROUTES_OR_EVENTS	Skip routes and events in the supplied configuration.
WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↩ DOM_DUPKEY_ABORT	When loading JSON user values, treat as an error when duplicate keys are found.
WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↩ DOM_DUPKEY_USEFIRST	When loading JSON user values, when duplicate keys are found, keep the first one.
WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↩ DOM_DUPKEY_USELAST	When loading JSON user values, when duplicate keys are found, keep the last one.
WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↩ DOM_MAINTAINDICTORDER	When loading JSON user values, store extra sequence information so that dictionaries are rendered in same sequence by `json_dom_dump()` and `wolfsentry_kv_render_value()`.
WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_↩ ONLY_ROUTES	Don't flush the events or user values, just flush the routes, before loading incremental configuration JSON.
WOLFSENTRY_CONFIG_LOAD_FLAG_FINI	Internal use.

8.2.2.3 wolfsentry_init_flags_t

```
enum wolfsentry_init_flags_t
```

flags to pass to wolfsentry_init_ex(), to be ORd together.

Enumerator

WOLFSENTRY_INIT_FLAG_NONE	Default behavior.
WOLFSENTRY_INIT_FLAG_LOCK_SHARED_↩ ERROR_CHECKING	Enables supplementary error checking on shared lock usage (not currently implemented)

8.2.3 Function Documentation

8.2.3.1 wolfsentry_context_clone()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_clone (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_context ** clone,
            wolfsentry_clone_flags_t flags )
```

Clones a wolfsentry context.

Parameters

clone	the destination wolfsentry context, should be a pointer to a NULL pointer as this function will malloc
flags	set to WOLFSENTRY_CLONE_FLAG_AT_CREATION to use the config at the creation of the original wolfsentry context instead of the current configuration

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.2 wolfsentry_context_enable_actions()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_enable_actions (
            WOLFSENTRY_CONTEXT_ARGS_IN  )
```

Re-enable automatic dispatch of actions on the wolfsentry context.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.3 wolfsentry_context_exchange()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_exchange (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_context * wolfsentry2 )
```

Swaps information between two wolfsentry contexts.

Parameters

wolfsentry2	the new context to swap into the primary context

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.4 wolfsentry_context_flush()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_flush (
            WOLFSENTRY_CONTEXT_ARGS_IN  )
```

Flushes the route, event, and user value tables from the wolfsentry context.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.5 wolfsentry_context_free()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_free (
            WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry)  )
```

Frees the wolfsentry context and the tables within it. The wolfsentry context will be a pointer to NULL upon success.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true, and `*wolfsentry` is `NULL`, on success.

See also

wolfsentry_context_shutdown

WOLFSENTRY_CONTEXT_ARGS_IN_EX

8.2.3.6 wolfsentry_context_inhibit_actions()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_inhibit_actions (
            WOLFSENTRY_CONTEXT_ARGS_IN  )
```

Disable automatic dispatch of actions on the wolfsentry context.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.7 wolfsentry_defaultconfig_get()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_get (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_eventconfig * config )
```

Get the default config from a wolfsentry context.

Parameters

config	a config struct to be loaded with a copy of the config

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.2.3.8 wolfsentry_defaultconfig_update()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_update (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_eventconfig * config )
```

Updates mutable fields of the default config (all but wolfsentry_eventconfig::route_private_data_size and wolfsentry_eventconfig::route_private_data_alignment)

Parameters

config	the config struct to load from

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.2.3.9 wolfsentry_init()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init (
            struct wolfsentry_build_settings caller_build_settings,
            WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface
*hpi) ,
            const struct wolfsentry_eventconfig * config,
            struct wolfsentry_context ** wolfsentry )
```

Allocates and initializes the wolfsentry context.

Parameters

caller_build_settings	Pass wolfsentry_build_settings here (definition is in `wolfsentry_settings.h`)
config	a pointer to a `wolfsentry_eventconfig` to use (can be NULL)
wolfsentry	a pointer to the wolfsentry_context to initialize

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

struct wolfsentry_host_platform_interface

WOLFSENTRY_CONTEXT_ARGS_IN_EX

8.2.3.10 wolfsentry_shutdown()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_shutdown (
            WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry)  )
```

Shut down wolfSentry, freeing all resources. Gets an exclusive lock on the context, then calls wolfsentry_context_free().

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true, and `*wolfsentry` is `NULL`, on success.

See also

wolfsentry_context_free

WOLFSENTRY_CONTEXT_ARGS_IN_EX

8.3 Diagnostics, Control Flow Helpers, and Compiler Attribute Helpers

Macros

- #define **WOLFSENTRY_SOURCE_ID**

  In each source file in the wolfSentry library, `WOLFSENTRY_SOURCE_ID` is defined to a number that is decoded using `enum wolfsentry_source_id`. Application source files that use the below error encoding and rendering macros must also define `WOLFSENTRY_SOURCE_ID` to a number, starting with `WOLFSENTRY_SOURCE_ID_USER_BASE`, and can use `wolfsentry_user_source_string_set()` or `WOLFSENTRY_REGISTER_SOURCE()` to arrange for error and warning messages that render the source code file by name.
- #define **WOLFSENTRY_ERRCODE_FMT**

  String-literal macro for formatting `wolfsentry_errcode_t` using `printf()`-type functions.
- #define **WOLFSENTRY_SOURCE_ID_MAX** 127
- #define **WOLFSENTRY_ERROR_ID_MAX** 255
- #define **WOLFSENTRY_LINE_NUMBER_MAX** 65535
- #define **WOLFSENTRY_ERROR_DECODE_ERROR_CODE**(x)

  Extract the bare error (negative) or success (zero/positive) code from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_DECODE_SOURCE_ID**(x)

  Extract the bare source file ID from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_DECODE_LINE_NUMBER**(x)

  Extract the bare source line number from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_RECODE**(x)

  Take an encoded `wolfsentry_errcode_t` and recode it with the current source ID and line number.
- #define **WOLFSENTRY_ERROR_CODE_IS**(x, name)

  Take an encoded `wolfsentry_errcode_t x` and test if its error code matches short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_CODE_IS**(x, name)

  Take an encoded `wolfsentry_errcode_t x` and test if its error code matches short-form success `name` (e.g. `OK`).
- #define **WOLFSENTRY_IS_FAILURE**(x)

  Evaluates to true if `x` is a `wolfsentry_errcode_t` that encodes a failure.
- #define **WOLFSENTRY_IS_SUCCESS**(x)

  Evaluates to true if `x` is a `wolfsentry_errcode_t` that encodes a success.
- #define **WOLFSENTRY_ERROR_FMT**

  Convenience string-constant macro for formatting a `wolfsentry_errcode_t` for rendering by a `printf`-type function.
- #define **WOLFSENTRY_ERROR_FMT_ARGS**(x)

  Convenience macro supplying args to match the format directives in `WOLFSENTRY_ERROR_FMT`.
- #define **WOLFSENTRY_ERROR_ENCODE**(name)

  Compute a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_ENCODE**(x)

  Compute a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `OK`).
- #define WOLFSENTRY_DEBUG_CALL_TRACE

  Define to build the library or application to output codepoint and error code info at each return point.
- #define **WOLFSENTRY_ERROR_RETURN**(x)

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_RETURN**(x)

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `OK`).

- #define **WOLFSENTRY_ERROR_RETURN_RECODED**(x)

  Take an encoded `wolfsentry_errcode_t`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_ERROR_RERETURN**(x)

  Return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_RETURN_VALUE**(x)

  Return an arbitrary value.
- #define **WOLFSENTRY_RETURN_VOID**

  Return from a void function.
- #define **WOLFSENTRY_SUCCESS_RETURN_RECODED**(x)

  Take an encoded `wolfsentry_errcode_t`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_SUCCESS_RERETURN**(x)

  Return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_UNLOCK_FOR_RETURN_EX**(ctx)

  Unlock a previously locked `wolfsentry_context`, and if the unlock fails, return the error.
- #define **WOLFSENTRY_UNLOCK_FOR_RETURN**()

  Unlock the current context, and if the unlock fails, return the error.
- #define **WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN_EX**(ctx)

  Unlock a previously locked `wolfsentry_context`, and abandon a held promotion reservation if any (see `wolfsentry_lock_unlock()`), and if the operation fails, return the error.
- #define **WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN**()

  Unlock the current context, and abandon a held promotion reservation if any (see `wolfsentry_lock_unlock()`), and if the operation fails, return the error.
- #define **WOLFSENTRY_MUTEX_EX**(ctx)

  Get a mutex on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_MUTEX_OR_RETURN**()

  Get a mutex on the current context, and on failure, return the `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_SHARED_EX**(ctx)

  Get a shared lock on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_SHARED_OR_RETURN**()

  Get a shared lock on the current context, and on failure, return the `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_PROMOTABLE_EX**(ctx)

  Get a mutex on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_PROMOTABLE_OR_RETURN**()

  Get a shared lock with mutex promotion reservation on the current context, and on failure, return the `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_UNLOCK_AND_RETURN**(ret)

  Unlock the current context, and return the supplied `wolfsentry_errcode_t` .
- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN**(name)

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED**(x)

  Unlock the current context, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_EX**(ctx, name)

  Unlock a previously locked `wolfsentry_context ctx`, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED_EX**(ctx, x)

  Unlock a previously locked `wolfsentry_context ctx`, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RERETURN**(x)

  Unlock the current context, and return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_ERROR_RERETURN_AND_UNLOCK**(y)

Calculate the `wolfsentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN**(name)

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN_RECODED**(x)

  Unlock the current context, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RERETURN**(x)

  Unlock the current context, and return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_SUCCESS_RERETURN_AND_UNLOCK**(y)

  Calculate the `wolfsentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_UNLOCK_AND_RETURN_VALUE**(x)

  Unlock the current context, and return a value `x`.
- #define **WOLFSENTRY_UNLOCK_AND_RETURN_VOID**

  Unlock the current context, and return void.
- #define **WOLFSENTRY_RETURN_OK**

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.
- #define **WOLFSENTRY_UNLOCK_AND_RETURN_OK**

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.
- #define **WOLFSENTRY_RERETURN_IF_ERROR**(y)

  If `wolfsentry_errcode_t y` is a failure code, return it.
- #define **WOLFSENTRY_UNLOCK_AND_RERETURN_IF_ERROR**(y)

  If `wolfsentry_errcode_t y` is a failure code, unlock the current context and return the code.
- #define **WOLFSENTRY_WARN**(fmt, ...)

  Render a warning message using `WOLFSENTRY_PRINTF_ERR()`, or if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, `DO_NOTHING`.
- #define **WOLFSENTRY_WARN_ON_FAILURE**(...)

  Evaluate the supplied expression, and if the resulting `wolfsentry_errcode_t` encodes an error, render the expression and the decoded error using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, don't render a warning.
- #define **WOLFSENTRY_WARN_ON_FAILURE_LIBC**(...)

  Evaluate the supplied expression, and if it evaluates to a negative value, render the expression and the decoded `errno` using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, don't render a warning.
- #define **WOLFSENTRY_REGISTER_SOURCE**()

  Helper macro to call `wolfsentry_user_source_string_set()` with appropriate arguments.
- #define **WOLFSENTRY_REGISTER_ERROR**(name, msg)

  Helper macro to call `wolfsentry_user_error_string_set()` with appropriate arguments, given a short-form `name` and freeform string `msg`.
- #define **WOLFSENTRY_PRINTF_ERR**(...)

  printf-like macro, expecting a format as first arg, used for rendering warning and error messages. Can be overridden in WOLFSENTRY_USER_SETTINGS_FILE.

Typedefs

- typedef int32_t **wolfsentry_errcode_t**

  The structured result code type for wolfSentry. It encodes a failure or success code, a source code file ID, and a line number.

Enumerations

- enum **wolfsentry_source_id** {
  WOLFSENTRY_SOURCE_ID_UNSET = 0 ,
  WOLFSENTRY_SOURCE_ID_ACTIONS_C = 1 ,
  WOLFSENTRY_SOURCE_ID_EVENTS_C = 2 ,
  WOLFSENTRY_SOURCE_ID_WOLFSENTRY_INTERNAL_C = 3 ,
  WOLFSENTRY_SOURCE_ID_ROUTES_C = 4 ,
  WOLFSENTRY_SOURCE_ID_WOLFSENTRY_UTIL_C = 5 ,
  WOLFSENTRY_SOURCE_ID_KV_C = 6 ,
  WOLFSENTRY_SOURCE_ID_ADDR_FAMILIES_C = 7 ,
  WOLFSENTRY_SOURCE_ID_JSON_LOAD_CONFIG_C = 8 ,
  WOLFSENTRY_SOURCE_ID_JSON_JSON_UTIL_C = 9 ,
  WOLFSENTRY_SOURCE_ID_LWIP_PACKET_FILTER_GLUE_C = 10 ,
  WOLFSENTRY_SOURCE_ID_ACTION_BUILTINS_C = 11 ,
  WOLFSENTRY_SOURCE_ID_USER_BASE = 112 }
- enum **wolfsentry_error_id** {
  WOLFSENTRY_ERROR_ID_OK = 0 ,
  WOLFSENTRY_ERROR_ID_NOT_OK = -1 ,
  WOLFSENTRY_ERROR_ID_INTERNAL_CHECK_FATAL = -2 ,
  WOLFSENTRY_ERROR_ID_SYS_OP_FATAL = -3 ,
  WOLFSENTRY_ERROR_ID_SYS_OP_FAILED = -4 ,
  WOLFSENTRY_ERROR_ID_SYS_RESOURCE_FAILED = -5 ,
  WOLFSENTRY_ERROR_ID_INCOMPATIBLE_STATE = -6 ,
  WOLFSENTRY_ERROR_ID_TIMED_OUT = -7 ,
  WOLFSENTRY_ERROR_ID_INVALID_ARG = -8 ,
  WOLFSENTRY_ERROR_ID_BUSY = -9 ,
  WOLFSENTRY_ERROR_ID_INTERRUPTED = -10 ,
  WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_BIG = -11 ,
  WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_SMALL = -12 ,
  WOLFSENTRY_ERROR_ID_STRING_ARG_TOO_LONG = -13 ,
  WOLFSENTRY_ERROR_ID_BUFFER_TOO_SMALL = -14 ,
  WOLFSENTRY_ERROR_ID_IMPLEMENTATION_MISSING = -15 ,
  WOLFSENTRY_ERROR_ID_ITEM_NOT_FOUND = -16 ,
  WOLFSENTRY_ERROR_ID_ITEM_ALREADY_PRESENT = -17 ,
  WOLFSENTRY_ERROR_ID_ALREADY_STOPPED = -18 ,
  WOLFSENTRY_ERROR_ID_WRONG_OBJECT = -19 ,
  WOLFSENTRY_ERROR_ID_DATA_MISSING = -20 ,
  WOLFSENTRY_ERROR_ID_NOT_PERMITTED = -21 ,
  WOLFSENTRY_ERROR_ID_ALREADY = -22 ,
  WOLFSENTRY_ERROR_ID_CONFIG_INVALID_KEY = -23 ,
  WOLFSENTRY_ERROR_ID_CONFIG_INVALID_VALUE = -24 ,
  WOLFSENTRY_ERROR_ID_CONFIG_OUT_OF_SEQUENCE = -25 ,
  WOLFSENTRY_ERROR_ID_CONFIG_UNEXPECTED = -26 ,
  WOLFSENTRY_ERROR_ID_CONFIG_MISPLACED_KEY = -27 ,
  WOLFSENTRY_ERROR_ID_CONFIG_PARSER = -28 ,
  WOLFSENTRY_ERROR_ID_CONFIG_MISSING_HANDLER = -29 ,
  WOLFSENTRY_ERROR_ID_CONFIG_JSON_VALUE_SIZE = -30 ,
  WOLFSENTRY_ERROR_ID_OP_NOT_SUPP_FOR_PROTO = -31 ,
  WOLFSENTRY_ERROR_ID_WRONG_TYPE = -32 ,
  WOLFSENTRY_ERROR_ID_BAD_VALUE = -33 ,
  WOLFSENTRY_ERROR_ID_DEADLOCK_AVERTED = -34 ,
  WOLFSENTRY_ERROR_ID_OVERFLOW_AVERTED = -35 ,
  WOLFSENTRY_ERROR_ID_LACKING_MUTEX = -36 ,
  WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK = -37 ,
  WOLFSENTRY_ERROR_ID_LIB_MISMATCH = -38 ,
  WOLFSENTRY_ERROR_ID_LIBCONFIG_MISMATCH = -39 ,
  WOLFSENTRY_ERROR_ID_IO_FAILED = -40 ,
  WOLFSENTRY_ERROR_ID_WRONG_ATTRIBUTES = -41 ,

WOLFSENTRY_ERROR_ID_USER_BASE = -128 ,
WOLFSENTRY_SUCCESS_ID_OK = 0 ,
WOLFSENTRY_SUCCESS_ID_LOCK_OK_AND_GOT_RESV = 1 ,
WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX = 2 ,
WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK = 3 ,
WOLFSENTRY_SUCCESS_ID_USED_FALLBACK = 4 ,
WOLFSENTRY_SUCCESS_ID_YES = 5 ,
WOLFSENTRY_SUCCESS_ID_NO = 6 ,
WOLFSENTRY_SUCCESS_ID_ALREADY_OK = 7 ,
WOLFSENTRY_SUCCESS_ID_DEFERRED = 8 ,
WOLFSENTRY_SUCCESS_ID_USER_BASE = 128 }

Functions

- WOLFSENTRY_API const char * **wolfsentry_errcode_source_string** (wolfsentry_errcode_t e)

  Return the name of the source code file associated with `wolfsentry_errcode_t e`, or "unknown user defined source", or "unknown source".
- WOLFSENTRY_API const char * **wolfsentry_errcode_error_string** (wolfsentry_errcode_t e)

  Return a description of the failure or success code associated with `wolfsentry_errcode_t e`, or various "unknown" strings if not known.
- WOLFSENTRY_API const char * **wolfsentry_errcode_error_name** (wolfsentry_errcode_t e)

  Return the short name of the failure or success code associated with `wolfsentry_errcode_t e`, or `wolfsentry_errcode_error_string(e)` if not known.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_source_string_set** (enum wolfsentry_↩ source_id wolfsentry_source_id, const char *source_string)

  Register a source code file so that `wolfsentry_errcode_source_string()`, and therefore `WOLFSENTRY_ERROR_FMT_ARG` and `WOLFSENTRY_WARN_ON_FAILURE()`, can render it. Note that `source_string` must be a string constant or otherwise remain valid for the duration of runtime.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_error_string_set** (enum wolfsentry_error_↩ id wolfsentry_error_id, const char *message_string)

  Register an error (negative) or success (positive) code, and corresponding message, so that `wolfsentry_errcode_error_strin` and therefore `WOLFSENTRY_ERROR_FMT_ARGS()` and `WOLFSENTRY_WARN_ON_FAILURE()`, can render it in human-readable form. Note that `error_string` must be a string constant or otherwise remain valid for the duration of runtime.

8.3.1 Detailed Description

8.3.2 Macro Definition Documentation

8.3.2.1 WOLFSENTRY_DEBUG_CALL_TRACE

```
#define WOLFSENTRY_DEBUG_CALL_TRACE
```

Define to build the library or application to output codepoint and error code info at each return point.

In the wolfSentry library, and optionally in applications, all returns from functions are through macros, typically `WOLFSENTRY_ERROR_RETURN()`. In normal builds, these macros just `return` as usual. But if `WOLFSENTRY_DEBUG_CALL_TRACE` is defined, then alternative implementations are used that print trace info, using the `WOLFSENTRY_PRINTF_ERR()` macro, which has platform-specific default definitions in `wolfsentry_settings.h`, subject to override.

8.4 Route/Rule Subsystem

Data Structures

- struct wolfsentry_route_endpoint

  struct for exporting socket addresses, with fixed-length fields
- struct wolfsentry_route_metadata_exports

  struct for exporting route metadata for access by applications
- struct wolfsentry_route_exports

  struct for exporting a route for access by applications
- struct wolfsentry_sockaddr

  struct for passing socket addresses into `wolfsentry_route_*()` API routines

Macros

- #define **WOLFSENTRY_ROUTE_DEFAULT_POLICY_MASK** (WOLFSENTRY_ACTION_RES_ACCEPT | WOLFSENTRY_ACTION_RES_REJECT | WOLFSENTRY_ACTION_RES_STOP | WOLFSENTRY_ACTION_RES_ERROR)

  Bit mask spanning the bits allowed by wolfsentry_route_table_default_policy_set()
- #define **WOLFSENTRY_ROUTE_WILDCARD_FLAGS**

  Bit mask for the wildcard bits in a wolfsentry_route_flags_t.
- #define **WOLFSENTRY_ROUTE_IMMUTABLE_FLAGS**

  Bit mask for the bits in a wolfsentry_route_flags_t that can't change after the implicated route has been inserted in the route table.
- #define WOLFSENTRY_ROUTE_INTERNAL_FLAGS
- #define **WOLFSENTRY_SOCKADDR**(n)

  Macro to instantiate a wolfsentry_sockaddr with an `addr` field sized to hold `n` bits of address data. Cast to `struct wolfsentry_sockaddr` to pass as API argument.

Enumerations

- enum wolfsentry_route_flags_t {
  WOLFSENTRY_ROUTE_FLAG_NONE = 0U ,
  WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD ,
  WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS ,
  WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN ,
  WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT ,
  WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR_BITMASK ,
  WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_BITMASK ,
  WOLFSENTRY_ROUTE_FLAG_IN_TABLE ,
  WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE ,
  WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED ,
  WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED ,
  WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED ,
  WOLFSENTRY_ROUTE_FLAG_GREENLISTED ,
  WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_HITS ,
  WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_CURRENT_CONNECTIONS ,
  WOLFSENTRY_ROUTE_FLAG_PORT_RESET }

bit field specifying attributes of a route/rule

- enum wolfsentry_format_flags_t { WOLFSENTRY_FORMAT_FLAG_NONE , WOLFSENTRY_FORMAT_FLAG_ALWAYS_NUMERIC }

  bit field with options for rendering

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_check_flags_sensical** (wolfsentry_route_flags_t flags)

  Check the self-consistency of `flags`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_into_table** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label↩_len, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_into_table** (WOLFSENTRY_CONTEXT_AR struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_route_exports *route↩_exports, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports, and takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Insert a route into the route table.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_route_exports *route_exports, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_into_table_and_check_out** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label↩_len, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that takes an explicit `route_table`, and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_into_table_and_↩check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_route_exports *route_exports, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports, takes an explicit `route_table`, and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_and_check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_and_check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_route_exports *route↩_exports, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_delete_from_table** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, wolfsentry_action_res_t *action_results, int *n_deleted)

  Variant of wolfsentry_route_delete() that takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *trigger_label, int trigger_label_len, wolfsentry_action_res_t *action_results, int *n_deleted)

  Delete route from the route table. The supplied parameters, including the flags, must match the route exactly, else `ITEM_NOT_FOUND` will result. To avoid fidgety parameter matching, use wolfsentry_route_delete_by_id(). The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_by_id (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, wolfsentry_ent_id_t id, const char *trigger_label, int trigger_label_len, wolfsentry_action_res_t *action_results)

  Delete a route from its route table using its ID. The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_main_table (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table **table)

  Get a pointer to the internal route table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_start (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor)

  Open a cursor to interate through a routes table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_head (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor)

  Reset the cursor to the beginning of a table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_tail (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor)

  Move the cursor to the end of a table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_current (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

  Get the current position for the table cursor.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_prev (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

  Get the previous position for the table cursor.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_next (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

  Get the next position for the table cursor.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_end (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor)

  Frees the table cursor. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_set (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t default_policy)

  Set a table's default policy.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_default_policy_set** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t default_policy)

  variant of wolfsentry_route_table_default_policy_set() that uses the main route table implicitly, and takes care of context locking.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_get (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *default_policy)

  Get a table's default policy. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_default_policy_get** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t *default_policy)

  variant of wolfsentry_route_table_default_policy_get() that uses the main route table implicitly. Caller must have a lock on the context at entry.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, int exact_p, wolfsentry_route_flags_t *inexact_matches, struct wolfsentry_route **route)

  Increments a reference counter for a route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_action_res_t *action_results)

  Decrease a reference counter for a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_clear_default_event** (WOLFSENTRY_CONTEXT_ARGS, struct wolfsentry_route_table *table)

  Clear an event previously set by wolfsentry_route_table_set_default_event().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_set_default_event** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *table, const char *event_label, int event_label_len)

  Set an event to be used as a foster parent event for routes with no parent event of their own.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_get_default_event** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *table, char *event_label, int *event_label_len)

  Get the event, if any, set by wolfsentry_route_table_set_default_event()
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_fallthrough_route_get (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *route_table, const struct wolfsentry_route **fallthrough_route)

  Retrieve the default route in a route table, chiefly to pass to wolfsentry_route_update_flags().
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs (const struct wolfsentry_route *route, wolfsentry_addr_family_t *af, wolfsentry_addr_bits_t *local_addr_len, const byte **local_addr, wolfsentry_addr_bits_t *remote_addr_len, const byte **remote_addr)

  Extract numeric address family and binary address pointers from a `wolfsentry_route`
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_export (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *route, struct wolfsentry_route_exports *route_exports)

  Exports a route.
- WOLFSENTRY_API const struct wolfsentry_event * wolfsentry_route_parent_event (const struct wolfsentry_route *route)

  Get a parent event from a given route. Typically used in the wolfsentry_action_callback_t callback. Note: returned wolfsentry_event remains valid only as long as the wolfsentry lock is held (shared or exclusive).
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_table** (WOLFSENTRY_CONTEXT_ARGS struct wolfsentry_route_table *route_table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that accepts an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Submit an event into wolfsentry and pass it through the filters. The action_results are cleared on entry, and can be checked to see what actions wolfsentry took, and what actions the caller should take (most saliently, WOLFSENTRY_ACTION_RES_ACCEPT or WOLFSENTRY_ACTION_RES_REJECT). `action_results` can be filtered with constructs like `WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_REJECT)`
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_table_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that accepts an explicit `route_table`, and doesn't clear `action_results` on entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_inited_result** (WOLFSENTRY_CONTEX const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

Variant of wolfsentry_route_event_dispatch() that doesn't clear `action_results` on entry.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_id** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, mainly for use by application code that tracks ID/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_id_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, and doesn't clear `action_results` on entry, mainly for use by application code that tracks ID/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_route** (WOLFSENTRY_CONTEXT_ARGS_ struct wolfsentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, mainly for use by application code that tracks route/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_route_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, and doesn't clear `action_results` on entry, mainly for use by application code that tracks route/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_routes_get** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *table, wolfsentry_hitcount_t *max_purgeable_routes)

  Retrieve the current limit for ephemeral routes in `table`. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_routes_set** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *table, wolfsentry_hitcount_t max_purgeable_routes)

  Set the limit for ephemeral routes in `table`. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_idle_time_get** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_time_t *max_purgeable_idle_time)

  Retrieve the current absolute maximum idle time for a purgeable route (controls forced purges of routes with nonzero wolfsentry_route_metadata_exports.connection_count). Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_idle_time_set** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_time_t max_purgeable_idle_time)

  Set the maximum idle time for a purgeable route (controls forced purges of routes with nonzero wolfsentry_route_metadata_exports.conne Default is no limit. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_purge_time_set** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_time_t purge_after)

  Set the time after which `route` in `table` is to be subject to automatic purge. `0` sets the route as persistent. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Purges all stale (expired) routes from `table`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_stale_purge_one** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_stale_purge() that purges at most one stale route, to limit time spent working.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_stale_purge_one_opportunistically** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_stale_purge() that purges at most one stale route, and only if the context lock is uncontended.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flush_table (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Flush routes from a given table.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_clear_insert_action_status (WOLFSENTRY_CONTEXT_ARG wolfsentry_action_res_t *action_results)

  Clears the WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED flag on all routes in the table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_insert_actions (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t *action_results)

  Executes the insert actions for all routes in the table that don't have WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED set.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_private_data (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, void **private_data, size_t *private_data_size)

  Gets the private data for a given route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_flags (const struct wolfsentry_route *route, wolfsentry_route_flags_t *flags)

  Gets the flags for a route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_metadata (const struct wolfsentry_route *route, struct wolfsentry_route_metadata_exports *metadata)

  Gets the metadata for a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_metadata_exports** (struct wolfsentry_route_exports *route_exports)

  clear metadata counts (wolfsentry_route_metadata_exports::purge_after, wolfsentry_route_metadata_exports::connection_count, wolfsentry_route_metadata_exports::derogatory_count, and wolfsentry_route_metadata_exports::commendable_count) in wolfsentry_route_exports to prepare for use with wolfsentry_route_insert_by_exports()
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_update_flags (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_route_flags_t flags_to_set, wolfsentry_route_flags_t flags_to_↩ clear, wolfsentry_route_flags_t *flags_before, wolfsentry_route_flags_t *flags_after, wolfsentry_action_res_t *action_results)

  Update the route flags.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_increment_derogatory_count** (WOLFSENTRY_CONTEXT_AR struct wolfsentry_route *route, int count_to_add, int *new_derogatory_count_ptr)

  Increase the derogatory event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_increment_commendable_count** (WOLFSENTRY_CONTEXT_ struct wolfsentry_route *route, int count_to_add, int *new_commendable_count)

  Increase the commendable event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_derogatory_count** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route *route, int *old_derogatory_count_ptr)

  Reset the derogatory event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_commendable_count** (WOLFSENTRY_CONTEXT_ARG struct wolfsentry_route *route, int *old_commendable_count_ptr)

  Reset the commendable event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_set_wildcard (struct wolfsentry_route *route, wolfsentry_route_flags_t wildcards_to_set)

  Set wildcard flags for a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_format_address** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t sa_family, const byte *addr, unsigned int addr_bits, char *buf, int *buflen)

  Render a binary address in human-readable form to a buffer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_flag_assoc_by_flag** (wolfsentry_route_flags_t flag, const char **name)

  Retrieve the name of a route flag, given its numeric value. Note that `flag` must have exactly one bit set, else `ITEM_NOT_FOUND` will be returned.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_flag_assoc_by_name** (const char *name, int len, wolfsentry_route_flags_t *flag)

  Retrieve the numeric value of a route flag, given its name.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_format_json** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *r, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

Render a route to an output buffer, in JSON format, advancing the output buffer pointer by the length of the rendered output.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_start** (WOLFSENTRY_CONTEXT_ARGS_IN const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Start a rendering loop to export the route table contents as a JSON document that is valid input for wolfsentry_config_json_feed() or wolfsentry_config_json_oneshot(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount. Caller must have a shared or exclusive lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_next** (WOLFSENTRY_CONTEXT_ARGS_IN const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Render a route within a loop started with wolfsentry_route_table_dump_json_start(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_end** (WOLFSENTRY_CONTEXT_ARGS_IN const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Finish a rendering loop started with wolfsentry_route_table_dump_json_start(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_render_flags** (wolfsentry_route_flags_t flags, FILE *f)

  Render route flags in human-readable form to a stream.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *r, FILE *f)

  Renders route information to a file pointer.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_exports_render (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_exports *r, FILE *f)

  Renders route exports information to a file pointer.

8.4.1 Detailed Description

8.4.2 Macro Definition Documentation

8.4.2.1 WOLFSENTRY_ROUTE_INTERNAL_FLAGS

```
#define WOLFSENTRY_ROUTE_INTERNAL_FLAGS
```

Value:

```
((wolfsentry_route_flags_t) \
(WOLFSENTRY_ROUTE_FLAG_IN_TABLE | \
 WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE | \
 WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED | \
 WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED))
```

8.4.3 Enumeration Type Documentation

8.4.3.1 wolfsentry_format_flags_t

```
enum wolfsentry_format_flags_t
```

bit field with options for rendering

Enumerator

WOLFSENTRY_FORMAT_FLAG_NONE	Default rendering behavior.
WOLFSENTRY_FORMAT_FLAG_ALWAYS_↩ NUMERIC	When rendering address families and protocols, always render as bare integers. Currently honored by wolfsentry_route_format_json().

8.4.3.2 wolfsentry_route_flags_t

`enum wolfsentry_route_flags_t`

bit field specifying attributes of a route/rule

Enumerator

WOLFSENTRY_ROUTE_FLAG_NONE	No attributes
WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_↩ WILDCARD	Address family is wildcard – match all traffic in specified direction(s), optionally with specified interfaces.
WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_↩ ADDR_WILDCARD	Remote address is wildcard – match any remote address.
WOLFSENTRY_ROUTE_FLAG_SA_PROTO_↩ WILDCARD	Protocol is wildcard – match any protocol.
WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_↩ PORT_WILDCARD	Local port is wildcard – match any local port.
WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_↩ ADDR_WILDCARD	Local address is wildcard – match any local address.
WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_↩ PORT_WILDCARD	Remote port is wildcard – match any remote port.
WOLFSENTRY_ROUTE_FLAG_REMOTE_↩ INTERFACE_WILDCARD	Ingestion interface is wildcard – match any ingestion interface.
WOLFSENTRY_ROUTE_FLAG_LOCAL_↩ INTERFACE_WILDCARD	Local interface (usually same as remote interface) is wildcard – match any local interface.
WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT↩ _WILDCARD	Match regardless of parent event mismatch.
WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT↩ _NUMBERS	Interpret port names using TCP/UDP mappings (available unless build option WOLFSENTRY_NO_GETPROTOBY is defined)
WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN	Match inbound traffic.
WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT	Match outbound traffic (if WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN and WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT are both set, traffic in both directions is matched)
WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR↩ _BITMASK	Supplied remote address consists of an address followed by a bitmask, and its addr_len is the total bit count for the address and mask. The bit count for the address and bitmask must be equal, and each must be a multiple of 8, i.e. aligned to a byte boundary. Matching will be performed by checking that masked addresses are equal.

Enumerator

WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_↩ BITMASK	Supplied local address consists of an address followed by a bitmask, and its addr_len is the total bit count for the address and mask. The bit count for the address and bitmask must be equal, and each must be a multiple of 8, i.e. aligned to a byte boundary. Matching will be performed by checking that masked addresses are equal.
WOLFSENTRY_ROUTE_FLAG_IN_TABLE	Internal use – marks route as resident in table.
WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE	Internal use – marks route as deleted.
WOLFSENTRY_ROUTE_FLAG_INSERT_↩ ACTIONS_CALLED	Internal use – records that route insertion actions have been completed.
WOLFSENTRY_ROUTE_FLAG_DELETE_↩ ACTIONS_CALLED	Internal use – records that route deletion actions have been completed.
WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED	Traffic that matches a route with this flag set will be rejected.
WOLFSENTRY_ROUTE_FLAG_GREENLISTED	Traffic that matches a route with this flag set will be accepted.
WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_↩ HITS	Don't keep traffic statistics for this rule (avoid counting overhead)
WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_↩ CURRENT_CONNECTIONS	Don't keep concurrent connection count for this rule (don't impose connection limit, even if set in the applicable `wolfsentry_eventconfig`)
WOLFSENTRY_ROUTE_FLAG_PORT_RESET	If traffic is rejected by this rule, set WOLFSENTRY_ACTION_RES_PORT_RESET in the returned wolfsentry_action_res_t, prompting generation by the network stack of a TCP reset, ICMP unreachable, or other applicable reply packet.

8.4.4 Function Documentation

8.4.4.1 wolfsentry_route_bulk_clear_insert_action_status()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_clear_insert_action_status (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            wolfsentry_action_res_t * action_results )
```

Clears the WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED flag on all routes in the table.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_route_bulk_insert_actions()

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.2 wolfsentry_route_bulk_insert_actions()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_insert_actions (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            wolfsentry_action_res_t * action_results )
```

Executes the insert actions for all routes in the table that don't have WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED set.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_route_bulk_clear_insert_action_status()

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.3 wolfsentry_route_delete()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            void * caller_arg,
            const struct wolfsentry_sockaddr * remote,
            const struct wolfsentry_sockaddr * local,
            wolfsentry_route_flags_t flags,
            const char * trigger_label,
            int trigger_label_len,
            wolfsentry_action_res_t * action_results,
            int * n_deleted )
```

Delete route from the route table. The supplied parameters, including the flags, must match the route exactly, else ITEM_NOT_FOUND will result. To avoid fidgety parameter matching, use wolfsentry_route_delete_by_id(). The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.

Parameters

caller_arg	an arbitrary pointer to be passed to callbacks
remote	the remote sockaddr for the route
local	the local sockaddr for the route
flags	flags for the route
trigger_label	a label for the trigger event (or null)
trigger_label_len	the length of the trigger_label parameter
action_results	a pointer to results of the insert action – all bits are cleared on entry.
n_deleted	a counter for the number of entries deleted

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.4 wolfsentry_route_delete_by_id()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_by_id (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            void * caller_arg,
            wolfsentry_ent_id_t id,
            const char * trigger_label,
            int trigger_label_len,
            wolfsentry_action_res_t * action_results )
```

Delete a route from its route table using its ID. The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.

Parameters

caller_arg	an arbitrary pointer to be passed to callbacks
id	the object ID, as returned by wolfsentry_route_insert() or wolfsentry_get_object_id()
trigger_label	a label for a trigger event (or null)
trigger_label_len	the length of the trigger_label parameter
action_results	a pointer to results of the insert action – all bits are cleared on entry.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.5 wolfsentry_route_drop_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_drop_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route * route,
            wolfsentry_action_res_t * action_results )
```

Decrease a reference counter for a route.

Parameters

route	the route to drop the reference for
action_results	a pointer to results of the action

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.6 wolfsentry_route_event_dispatch()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_sockaddr * remote,
            const struct wolfsentry_sockaddr * local,
            wolfsentry_route_flags_t flags,
            const char * event_label,
            int event_label_len,
            void * caller_arg,
            wolfsentry_ent_id_t * id,
            wolfsentry_route_flags_t * inexact_matches,
            wolfsentry_action_res_t * action_results )
```

Submit an event into wolfsentry and pass it through the filters. The action_results are cleared on entry, and can be checked to see what actions wolfsentry took, and what actions the caller should take (most saliently, WOLFSENTRY_ACTION_RES_ACCEPT or WOLFSENTRY_ACTION_RES_REJECT). `action_results` can be filtered with constructs like `WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_REJECT)`

Parameters

remote	the remote sockaddr details
local	the local sockaddr details
flags	the flags for the event, set to WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN for an incoming event
event_label	an optional label for a trigger event
event_label_len	the length of event_label
caller_arg	an arbitrary pointer to be passed to action callbacks
id	an optional pointer to a wolfsentry_ent_id_t that will be set to the ID of the matched route, if any
inexact_matches	details for inexact matches
action_results	a pointer to a wolfsentry_action_res_t, which will be used to record actions taken and to be taken

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.7 wolfsentry_route_export()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_export (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route * route,
            struct wolfsentry_route_exports * route_exports )
```

Exports a route.

`route_exports` remains valid only as long as the wolfsentry lock is held (shared or exclusive), unless the route was obtained via wolfsentry_route_get_reference(), in which case it's valid until wolfsentry_route_drop_reference().

Parameters

route	the route to export
route_exports	the struct to export into

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.8 wolfsentry_route_exports_render()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_exports_render (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route_exports * r,
            FILE * f )
```

Renders route exports information to a file pointer.

Parameters

r	the route exports to render
f	the pointer to render to

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.9 wolfsentry_route_flush_table()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flush_table (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table * table,
            wolfsentry_action_res_t * action_results )
```

Flush routes from a given table.

Parameters

table	the table to purge
action_results	the result bit field, pooling results from all constituent operations

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.10 wolfsentry_route_get_addrs()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs (
            const struct wolfsentry_route * route,
            wolfsentry_addr_family_t * af,
            wolfsentry_addr_bits_t * local_addr_len,
            const byte ** local_addr,
            wolfsentry_addr_bits_t * remote_addr_len,
            const byte ** remote_addr )
```

Extract numeric address family and binary address pointers from a `wolfsentry_route`

`local_addr` and `remote_addr` remain valid only as long as the wolfsentry lock is held (shared or exclusive), unless the route was obtained via wolfsentry_route_get_reference(), in which case it's valid until wolfsentry_route_drop_reference().

8.4.4.11 wolfsentry_route_get_flags()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_flags (
            const struct wolfsentry_route * route,
            wolfsentry_route_flags_t * flags )
```

Gets the flags for a route.

Parameters

route	the route to get the flags for
flags	a pointer to receive the flags

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.12 wolfsentry_route_get_main_table()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_main_table (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table ** table )
```

Get a pointer to the internal route table. Caller must have a lock on the context at entry.

Parameters

table	a pointer to a pointer to a table which will be filled

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()

WOLFSENTRY_UNLOCK_AND_RETURN()

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.13 wolfsentry_route_get_metadata()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_metadata (
            const struct wolfsentry_route * route,
            struct wolfsentry_route_metadata_exports * metadata )
```

Gets the metadata for a route.

Parameters

route	the route to get the metadata for
metadata	a pointer to a pointer to receive the metadata

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.14 wolfsentry_route_get_private_data()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_private_data (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
```

```
            struct wolfsentry_route * route,
            void ** private_data,
            size_t * private_data_size )
```

Gets the private data for a given route.

Parameters

route	the route to get the data from
private_data	a pointer to a pointer that will receive the data
private_data_size	a pointer that will recieve the size of the data

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.15 wolfsentry_route_get_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route_table * table,
            const struct wolfsentry_sockaddr * remote,
            const struct wolfsentry_sockaddr * local,
            wolfsentry_route_flags_t flags,
            const char * event_label,
            int event_label_len,
            int exact_p,
            wolfsentry_route_flags_t * inexact_matches,
            struct wolfsentry_route ** route )
```

Increments a reference counter for a route.

Parameters

table	the table to get the route from
remote	the remote sockaddr
local	the local sockaddr
flags	flags for the route
event_label	a label for the event
event_label_len	the length of the event_label parameter
exact_p	set to 1 for exact matches only
inexact_matches	wildcard flags hit
route	the route returned

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.16 wolfsentry_route_insert()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            void * caller_arg,
            const struct wolfsentry_sockaddr * remote,
            const struct wolfsentry_sockaddr * local,
            wolfsentry_route_flags_t flags,
            const char * event_label,
            int event_label_len,
            wolfsentry_ent_id_t * id,
            wolfsentry_action_res_t * action_results )
```

Insert a route into the route table.

Parameters

caller_arg	an arbitrary pointer to be passed to callbacks
remote	the remote sockaddr for the route
local	the local sockaddr for the route
flags	flags for the route
event_label	a label for the route
event_label_len	the length of the event_label parameter
id	the object ID
action_results	a pointer to results of the insert action

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.17 wolfsentry_route_parent_event()

```
WOLFSENTRY_API const struct wolfsentry_event * wolfsentry_route_parent_event (
            const struct wolfsentry_route * route )
```

Get a parent event from a given route. Typically used in the wolfsentry_action_callback_t callback. Note: returned wolfsentry_event remains valid only as long as the wolfsentry lock is held (shared or exclusive).

Parameters

route	a pointer to the route

Returns

a pointer to the parent event

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.18 wolfsentry_route_render()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route * r,
            FILE * f )
```

Renders route information to a file pointer.

Parameters

r	the route to render
f	the pointer to render to

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.19 wolfsentry_route_set_wildcard()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_set_wildcard (
            struct wolfsentry_route * route,
            wolfsentry_route_flags_t wildcards_to_set )
```

Set wildcard flags for a route.

Parameters

route	the route to set the flags for
wildcards_to_set	the wildcards to be set

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.20 wolfsentry_route_stale_purge()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table * table,
            wolfsentry_action_res_t * action_results )
```

Purges all stale (expired) routes from `table`.

Parameters

table	the table to purge from
action_results	the result bit field, pooling results from all constituent operations

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.21 wolfsentry_route_table_default_policy_get()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_get (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table * table,
            wolfsentry_action_res_t * default_policy )
```

Get a table's default policy. Caller must have a lock on the context at entry.

Parameters

table	the table to set the policy for
default_policy	the policy retrieved

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_defaultconfig_update()
WOLFSENTRY_SHARED_OR_RETURN()
WOLFSENTRY_UNLOCK_AND_RETURN()
WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.22 wolfsentry_route_table_default_policy_set()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_set (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table * table,
            wolfsentry_action_res_t default_policy )
```

Set a table's default policy.

Parameters

table	the table to set the policy for
default_policy	the policy to set

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.23 wolfsentry_route_table_fallthrough_route_get()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_fallthrough_route_get (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route_table * route_table,
            const struct wolfsentry_route ** fallthrough_route )
```

Retrieve the default route in a route table, chiefly to pass to wolfsentry_route_update_flags().

Caller must have a shared or mutex lock on the context at entry, but can release the lock on return and safely continue to access or update the route. Caller must drop the route when done, using wolfsentry_route_drop_reference() or wolfsentry_object_release().

See also

WOLFSENTRY_SHARED_OR_RETURN()
WOLFSENTRY_UNLOCK_FOR_RETURN()

8.4.4.24 wolfsentry_route_table_iterate_current()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_current (
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor * cursor,
            struct wolfsentry_route ** route )
```

Get the current position for the table cursor.

Parameters

table	the table for the cursor
cursor	a poiner for the cursor
route	a pointer to a pointer for the returned route

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.25 wolfsentry_route_table_iterate_end()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_end (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor ** cursor )
```

Frees the table cursor. Caller must have a lock on the context at entry.

Parameters

table	the table for the cursor
cursor	a poiner to a pointer for the cursor to free

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()

WOLFSENTRY_UNLOCK_AND_RETURN()

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.26 wolfsentry_route_table_iterate_next()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_next (
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor * cursor,
            struct wolfsentry_route ** route )
```

Get the next position for the table cursor.

Parameters

table	the table for the cursor
cursor	a poiner for the cursor
route	a pointer to a pointer for the returned route

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.27 wolfsentry_route_table_iterate_prev()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_prev (
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor * cursor,
            struct wolfsentry_route ** route )
```

Get the previous position for the table cursor.

Parameters

table	the table for the cursor
cursor	a poiner for the cursor
route	a pointer to a pointer for the returned route

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.28 wolfsentry_route_table_iterate_seek_to_head()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_head (
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor * cursor )
```

Reset the cursor to the beginning of a table.

Parameters

table	the table for the cursor
cursor	a poiner for the cursor

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.29 wolfsentry_route_table_iterate_seek_to_tail()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_tail (
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor * cursor )
```

Move the cursor to the end of a table.

Parameters

table	the table for the cursor
cursor	a poiner for the cursor

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.4.4.30 wolfsentry_route_table_iterate_start()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_start (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const struct wolfsentry_route_table * table,
            struct wolfsentry_cursor ** cursor )
```

Open a cursor to interate through a routes table. Caller must have a lock on the context at entry.

Parameters

table	a pointer to the table to open the cursor on
cursor	a pointer to a pointer for the cursor

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()

WOLFSENTRY_UNLOCK_AND_RETURN()

WOLFSENTRY_CONTEXT_ARGS_IN

8.4.4.31 wolfsentry_route_update_flags()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_update_flags (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_route * route,
            wolfsentry_route_flags_t flags_to_set,
            wolfsentry_route_flags_t flags_to_clear,
            wolfsentry_route_flags_t * flags_before,
            wolfsentry_route_flags_t * flags_after,
            wolfsentry_action_res_t * action_results )
```

Update the route flags.

Parameters

route	the route to update the flags for

Parameters

flags_to_set	new flags to set
flags_to_clear	old flags to clear
flags_before	a pointer that will be filled with the flags before the change
flags_after	a pointer that will be filled with flags after the change
action_results	the results bit field

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5 Action Subsystem

Macros

- #define **WOLFSENTRY_ACTION_RES_USER_SHIFT** 24U

  Bit shift for user-defined bit span in wolfsentry_action_res_t.
- #define **WOLFSENTRY_ACTION_RES_USER7** (1U << 31U)

  user-defined result bit #8 of 8. Defined with a macro to retain ISO C compliance on enum range.

Typedefs

- typedef wolfsentry_errcode_t(* wolfsentry_action_callback_t) (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_action *action, void *handler_arg, void *caller_arg, const struct wolfsentry_event *trigger_event, wolfsentry_action_type_t action_type, const struct wolfsentry_route *trigger_route, struct wolfsentry_route_table *route_table, struct wolfsentry_route *rule_route, wolfsentry_action_res_t *action_results)

  A callback that is triggered when an action is taken.

Enumerations

- enum wolfsentry_action_flags_t {
  WOLFSENTRY_ACTION_FLAG_NONE ,
  WOLFSENTRY_ACTION_FLAG_DISABLED }

  enum for communicating attributes of an action object
- enum wolfsentry_action_type_t {
  WOLFSENTRY_ACTION_TYPE_NONE ,
  WOLFSENTRY_ACTION_TYPE_POST ,
  WOLFSENTRY_ACTION_TYPE_INSERT ,
  WOLFSENTRY_ACTION_TYPE_MATCH ,
  WOLFSENTRY_ACTION_TYPE_UPDATE ,
  WOLFSENTRY_ACTION_TYPE_DELETE ,
  WOLFSENTRY_ACTION_TYPE_DECISION }

  enum communicating (to action handlers and internal logic) what type of action is being evaluated

- enum wolfsentry_action_res_t {
  WOLFSENTRY_ACTION_RES_NONE ,
  WOLFSENTRY_ACTION_RES_ACCEPT ,
  WOLFSENTRY_ACTION_RES_REJECT ,
  WOLFSENTRY_ACTION_RES_CONNECT ,
  WOLFSENTRY_ACTION_RES_DISCONNECT ,
  WOLFSENTRY_ACTION_RES_DEROGATORY ,
  WOLFSENTRY_ACTION_RES_COMMENDABLE ,
  WOLFSENTRY_ACTION_RES_STOP ,
  WOLFSENTRY_ACTION_RES_DEALLOCATED ,
  WOLFSENTRY_ACTION_RES_INSERTED ,
  WOLFSENTRY_ACTION_RES_ERROR ,
  WOLFSENTRY_ACTION_RES_FALLTHROUGH ,
  WOLFSENTRY_ACTION_RES_UPDATE ,
  WOLFSENTRY_ACTION_RES_PORT_RESET ,
  WOLFSENTRY_ACTION_RES_SENDING ,
  WOLFSENTRY_ACTION_RES_RECEIVED ,
  WOLFSENTRY_ACTION_RES_BINDING ,
  WOLFSENTRY_ACTION_RES_LISTENING ,
  WOLFSENTRY_ACTION_RES_STOPPED_LISTENING ,
  WOLFSENTRY_ACTION_RES_CONNECTING_OUT ,
  WOLFSENTRY_ACTION_RES_CLOSED ,
  WOLFSENTRY_ACTION_RES_UNREACHABLE ,
  WOLFSENTRY_ACTION_RES_SOCK_ERROR ,
  WOLFSENTRY_ACTION_RES_CLOSE_WAIT ,
  WOLFSENTRY_ACTION_RES_USER0 ,
  WOLFSENTRY_ACTION_RES_USER1 ,
  WOLFSENTRY_ACTION_RES_USER2 ,
  WOLFSENTRY_ACTION_RES_USER3 ,
  WOLFSENTRY_ACTION_RES_USER4 ,
  WOLFSENTRY_ACTION_RES_USER5 ,
  WOLFSENTRY_ACTION_RES_USER6 }

  bit field used to communicate states and attributes through the evaluation pipeline.

Functions

- WOLFSENTRY_API const char * **wolfsentry_action_res_assoc_by_flag** (wolfsentry_action_res_t res, unsigned int bit)

  Given a `bit` number (from 0 to 31), return the name of that bit if set in `res`, else return a null pointer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_action_res_assoc_by_name** (const char *bit_↩ name, int bit_name_len, wolfsentry_action_res_t *res)

  Given a `bit_name`, set `*res` to the corresponding bit number if known, failing which, return `ITEM_NOT_FOUND`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_insert (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_flags_t flags, wolfsentry_action_callback_t handler, void *handler_arg, wolfsentry_ent_id_t *id)

  Insert a new action into wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_res_t *action_results)

  Delete an action from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_flush_all (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flush all actions from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_action **action)

  Get a reference to an action.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action *action, wolfsentry_action_res_t *action_results)

  Drop a reference to an action.
- WOLFSENTRY_API const char * wolfsentry_action_get_label (const struct wolfsentry_action *action)

  Get the label for an action. This is the internal pointer to the label so should not be freed by the application.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_flags (struct wolfsentry_action *action, wolfsentry_action_flags_t *flags)

  Get the flags for an action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_update_flags (struct wolfsentry_action *action, wolfsentry_action_flags_t flags_to_set, wolfsentry_action_flags_t flags_to_clear, wolfsentry_action_flags_t *flags_before, wolfsentry_action_flags_t *flags_after)

  Update the flags for an action.

8.5.1 Detailed Description

8.5.2 Typedef Documentation

8.5.2.1 wolfsentry_action_callback_t

```
typedef wolfsentry_errcode_t(* wolfsentry_action_callback_t) (WOLFSENTRY_CONTEXT_ARGS_IN,
const struct wolfsentry_action *action, void *handler_arg, void *caller_arg, const struct wolfsentry↩
_event *trigger_event, wolfsentry_action_type_t action_type, const struct wolfsentry_route
*trigger_route, struct wolfsentry_route_table *route_table, struct wolfsentry_route *rule_↩
route, wolfsentry_action_res_t *action_results)
```

A callback that is triggered when an action is taken.

Parameters

action	a pointer to action details
handler_arg	an opaque pointer registered with `wolfsentry_action_insert()`, passed to every invocation of the handler
caller_arg	an opaque pointer supplied by the caller to the dispatching `wolfsentry_route_*()` API
trigger_event	the event which triggered the action, if any
action_type	the action type
trigger_route	a pointer to the subject route, reflecting instantaneous traffic attributes and contents
route_table	a pointer to the implicated route table
rule_route	a pointer to the matched route, reflecting rule logic
action_results	a pointer to the action results, to be read and/or updated by the handler

Returns

WOLFSENTRY_RETURN_OK if there is no error

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.3 Enumeration Type Documentation

8.5.3.1 wolfsentry_action_flags_t

`enum wolfsentry_action_flags_t`

enum for communicating attributes of an action object

Enumerator

WOLFSENTRY_ACTION_FLAG_NONE	Default attributes.
WOLFSENTRY_ACTION_FLAG_DISABLED	Disable this action – while this bit is set, dispatches will not call this action.

8.5.3.2 wolfsentry_action_res_t

`enum wolfsentry_action_res_t`

bit field used to communicate states and attributes through the evaluation pipeline.

Enumerator

WOLFSENTRY_ACTION_RES_NONE	initializer for wolfsentry_action_res_t.
WOLFSENTRY_ACTION_RES_ACCEPT	the route state or an action determined the event should be allowed.
WOLFSENTRY_ACTION_RES_REJECT	the route state or an action determined the event should be forbidden.
WOLFSENTRY_ACTION_RES_CONNECT	caller-preinited bit signaling that a connection was established.
WOLFSENTRY_ACTION_RES_DISCONNECT	caller-preinited bit signaling that a connection was dissolved.
WOLFSENTRY_ACTION_RES_DEROGATORY	the caller or an action designated this event derogatory for the peer.
WOLFSENTRY_ACTION_RES_COMMENDABLE	the caller or an action designated this event commendable for the peer.
WOLFSENTRY_ACTION_RES_STOP	when an action returns this, don't evaluate any more actions in the current action list.
WOLFSENTRY_ACTION_RES_DEALLOCATED	when an API call returns this, an object and its associated ID were deallocated from the system.
WOLFSENTRY_ACTION_RES_INSERTED	a side-effect route insertion was performed.
WOLFSENTRY_ACTION_RES_ERROR	an error occurred while processing actions.
WOLFSENTRY_ACTION_RES_FALLTHROUGH	dispatch classification (ACCEPT/REJECT) was by fallthrough policy.
WOLFSENTRY_ACTION_RES_UPDATE	signals to subsequent actions and the caller that the route state was updated (e.g. penaltyboxed).
WOLFSENTRY_ACTION_RES_PORT_RESET	when an action returns this, send a TCP reset or ICMP port unreachable packet.
WOLFSENTRY_ACTION_RES_SENDING	caller-preinited bit signaling outbound traffic.
WOLFSENTRY_ACTION_RES_RECEIVED	caller-preinited bit signaling inbound traffic.
WOLFSENTRY_ACTION_RES_BINDING	caller-preinited bit signaling that a socket will be bound.

Enumerator

WOLFSENTRY_ACTION_RES_LISTENING	caller-preinited bit signaling that a socket will be listened.
WOLFSENTRY_ACTION_RES_STOPPED_↩ LISTENING	caller-preinited bit signaling that a socket will stop being listened.
WOLFSENTRY_ACTION_RES_CONNECTING_OUT	caller-preinited bit signaling that an outbound connection will be attempted.
WOLFSENTRY_ACTION_RES_CLOSED	caller-preinited bit signaling that an association has closed/ended that wasn't created with _CONNECT.
WOLFSENTRY_ACTION_RES_UNREACHABLE	caller-preinited bit signaling that traffic destination was unreachable (unbound/unlistened).
WOLFSENTRY_ACTION_RES_SOCK_ERROR	caller-preinited bit signaling that a transport error occurred.
WOLFSENTRY_ACTION_RES_CLOSE_WAIT	caller-preinited bit signaling that an association has entered CLOSE_WAIT and will be closed.
WOLFSENTRY_ACTION_RES_USER0	user-defined result bit #1 of 8.
WOLFSENTRY_ACTION_RES_USER1	user-defined result bit #2 of 8.
WOLFSENTRY_ACTION_RES_USER2	user-defined result bit #3 of 8.
WOLFSENTRY_ACTION_RES_USER3	user-defined result bit #4 of 8.
WOLFSENTRY_ACTION_RES_USER4	user-defined result bit #5 of 8.
WOLFSENTRY_ACTION_RES_USER5	user-defined result bit #6 of 8.
WOLFSENTRY_ACTION_RES_USER6	user-defined result bit #7 of 8. start of user-defined results, with user-defined scheme (bit field, sequential, or other). 8 bits are available.

8.5.3.3 wolfsentry_action_type_t

```
enum wolfsentry_action_type_t
```

enum communicating (to action handlers and internal logic) what type of action is being evaluated

Enumerator

WOLFSENTRY_ACTION_TYPE_NONE	no action
WOLFSENTRY_ACTION_TYPE_POST	called when an event is posted.
WOLFSENTRY_ACTION_TYPE_INSERT	called when a route is added to the route table for this event.
WOLFSENTRY_ACTION_TYPE_MATCH	called by wolfsentry_route_dispatch() for a route match.
WOLFSENTRY_ACTION_TYPE_UPDATE	called by wolfsentry_route_dispatch() when the logical state (currently, flags) of an existing route changes.
WOLFSENTRY_ACTION_TYPE_DELETE	called when a route associated with this event expires or is otherwise deleted.
WOLFSENTRY_ACTION_TYPE_DECISION	called after final decision has been made by wolfsentry_route_event_dispatch∗().

8.5.4 Function Documentation

8.5.4.1 wolfsentry_action_delete()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_delete (
```

```
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            wolfsentry_action_res_t * action_results )
```

Delete an action from wolfsentry.

Parameters

label	the label of the action to delete
label_len	the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string
action_results	the returned result of the delete

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.4.2 wolfsentry_action_drop_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_drop_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_action * action,
            wolfsentry_action_res_t * action_results )
```

Drop a reference to an action.

Parameters

action	the action to drop the reference for
action_results	a pointer to the result of the function

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.4.3 wolfsentry_action_flush_all()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_flush_all (
            WOLFSENTRY_CONTEXT_ARGS_IN  )
```

Flush all actions from wolfsentry.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.4.4 wolfsentry_action_get_flags()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_flags (
            struct wolfsentry_action * action,
            wolfsentry_action_flags_t * flags )
```

Get the flags for an action.

Parameters

action	the action to get the flags for
flags	the flags to be returned

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.5.4.5 wolfsentry_action_get_label()

```
WOLFSENTRY_API const char * wolfsentry_action_get_label (
            const struct wolfsentry_action * action )
```

Get the label for an action. This is the internal pointer to the label so should not be freed by the application.

Parameters

action	the action to get the label for

Returns

the label for the action

8.5.4.6 wolfsentry_action_get_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            struct wolfsentry_action ** action )
```

Get a reference to an action.

Parameters

label	the label of the action to get the reference for
label_len	the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string
action	a pointer to a pointer for the action returned

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.4.7 wolfsentry_action_insert()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_insert (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            wolfsentry_action_flags_t flags,
            wolfsentry_action_callback_t handler,
            void * handler_arg,
            wolfsentry_ent_id_t * id )
```

Insert a new action into wolfsentry.

Parameters

label	the label for the action
label_len	the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string
flags	set flags for the action
handler	a callback handler when the action commences
handler_arg	an arbitrary pointer for the handler callback
id	the returned ID for the inserted action

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.5.4.8 wolfsentry_action_update_flags()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_update_flags (
            struct wolfsentry_action * action,
            wolfsentry_action_flags_t flags_to_set,
            wolfsentry_action_flags_t flags_to_clear,
            wolfsentry_action_flags_t * flags_before,
            wolfsentry_action_flags_t * flags_after )
```

Update the flags for an action.

Parameters

action	the action to update
flags_to_set	new flags to set
flags_to_clear	old flags to clear
flags_before	the flags before the change
flags_after	the flags after the change

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.6 Event Subsystem

Data Structures

- struct wolfsentry_eventconfig

  struct for representing event configuration

Enumerations

- enum wolfsentry_event_flags_t {
  WOLFSENTRY_EVENT_FLAG_NONE ,
  WOLFSENTRY_EVENT_FLAG_IS_PARENT_EVENT ,
  WOLFSENTRY_EVENT_FLAG_IS_SUBEVENT }

  bit field with attribute flags for events
- enum wolfsentry_eventconfig_flags_t {
  WOLFSENTRY_EVENTCONFIG_FLAG_NONE ,
  WOLFSENTRY_EVENTCONFIG_FLAG_DEROGATORY_THRESHOLD_IGNORE_COMMENDABLE ,
  WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY ,
  WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_ACTIONS }

  bit field with config flags for events

Functions

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_init (struct wolfsentry_context *wolfsentry, struct wolfsentry_eventconfig *config)

  Initializes a wolfsentry_eventconfig struct with the defaults from the wolfsentry context. If no wolfsentry context is provided this will initialize to zero.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_check (const struct wolfsentry_eventconfig *config)

  Checks the config for self-consistency and validity.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_insert (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_priority_t priority, const struct wolfsentry_eventconfig *config, wolfsentry_event_flags_t flags, wolfsentry_ent_id_t *id)

  Insert an event into wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_res_t *action_results)

  Delete an event from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_flush_all (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flush all events from wolfsentry.
- WOLFSENTRY_API const char * wolfsentry_event_get_label (const struct wolfsentry_event *event)

  Get the label for an event. This is the internal pointer to the label so should not be freed by the application.
- WOLFSENTRY_API wolfsentry_event_flags_t wolfsentry_event_get_flags (const struct wolfsentry_event *event)

  Get the flags for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_config (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_eventconfig *config)

  Get the configuration for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_update_config (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, const struct wolfsentry_eventconfig *config)

  Update the configuration for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_event **event)

  Get a reference to an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_event *event, wolfsentry_action_res_t *action_results)

  Drop a reference to an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_prepend (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

  Prepend an action into an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_append (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

  Append an action into an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_insert_after (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len, const char *point_action_label, int point_action_label_len)

  Insert an action into an event after another action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

  Delete an action from an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_set_aux_event (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, const char *aux_event_label, int aux_event_label_len)

  Set an auxiliary event for an event.

- WOLFSENTRY_API const struct wolfsentry_event * **wolfsentry_event_get_aux_event** (const struct wolfsentry_event *event)

  Retrieve an auxiliary event previously set with wolfsentry_event_set_aux_event().
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_start (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, struct wolfsentry_action_list_ent **cursor)

  Open a cursor for the actions in an event. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_next (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action_list_ent **cursor, const char **action_label, int *action_label_len)

  Get the next action in an event cursor. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_done (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action_list_ent **cursor)

  End iteration started with wolfsentry_event_action_list_start(). Caller must have a lock on the context at entry.

8.6.1 Detailed Description

8.6.2 Enumeration Type Documentation

8.6.2.1 wolfsentry_event_flags_t

```
enum wolfsentry_event_flags_t
```

bit field with attribute flags for events

Enumerator

WOLFSENTRY_EVENT_FLAG_NONE	Default attributes.
WOLFSENTRY_EVENT_FLAG_IS_PARENT_EVENT	Internally set – Event is parent of one or more routes.
WOLFSENTRY_EVENT_FLAG_IS_SUBEVENT	Internally set – Event is subevent of another event.

8.6.2.2 wolfsentry_eventconfig_flags_t

```
enum wolfsentry_eventconfig_flags_t
```

bit field with config flags for events

Enumerator

WOLFSENTRY_EVENTCONFIG_FLAG_NONE	Default config.
WOLFSENTRY_EVENTCONFIG_FLAG_↩ DEROGATORY_THRESHOLD_IGNORE_↩ COMMENDABLE	If set, then counts from `WOLFSENTRY_ACTION_RES_COMMENDABLE` are not subtracted from the derogatory count when checking for automatic penalty boxing.
WOLFSENTRY_EVENTCONFIG_FLAG_↩ COMMENDABLE_CLEARS_DEROGATORY	If set, then each count from `WOLFSENTRY_ACTION_RES_COMMENDABLE` zeroes the derogatory count.
WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_↩ ACTIONS	Internal use – Inhibits dispatch of actions listed in this event.

8.6.3 Function Documentation

8.6.3.1 wolfsentry_event_action_append()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_append (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            wolfsentry_action_type_t which_action_list,
            const char * action_label,
            int action_label_len )
```

Append an action into an event.

Parameters

event_label	the label of the event to append the action into
event_label_len	the length of the event_label
which_action_list	the action list of the event to update
action_label	the label of the action to insert
action_label_len	the length of the action_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.2 wolfsentry_event_action_delete()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_delete (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            wolfsentry_action_type_t which_action_list,
            const char * action_label,
            int action_label_len )
```

Delete an action from an event.

Parameters

event_label	the label of the event to delete the action from
event_label_len	the length of the event_label
which_action_list	the action list of the event to update
action_label	the label of the action to delete
action_label_len	the length of the action_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.3 wolfsentry_event_action_insert_after()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_insert_after (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            wolfsentry_action_type_t which_action_list,
            const char * action_label,
            int action_label_len,
            const char * point_action_label,
            int point_action_label_len )
```

Insert an action into an event after another action.

Parameters

event_label	the label of the event to insert the action into
event_label_len	the length of the event_label
which_action_list	the action list of the event to update
action_label	the label of the action to insert
action_label_len	the length of the action_label
point_action_label	the label of the action to insert after
point_action_label_len	the length of the point_action_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.4 wolfsentry_event_action_list_done()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_done (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_action_list_ent ** cursor )
```

End iteration started with wolfsentry_event_action_list_start(). Caller must have a lock on the context at entry.

Parameters

cursor	a pointer to a pointer for the cursor

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()
WOLFSENTRY_UNLOCK_AND_RETURN()
WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.5 wolfsentry_event_action_list_next()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_next (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_action_list_ent ** cursor,
            const char ** action_label,
            int * action_label_len )
```

Get the next action in an event cursor. Caller must have a lock on the context at entry.

Parameters

cursor	a pointer to a pointer for the cursor
action_label	a pointer to a pointer to the returned action_label
action_label_len	the length of action_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()
WOLFSENTRY_UNLOCK_AND_RETURN()
WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.6 wolfsentry_event_action_list_start()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_start (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            wolfsentry_action_type_t which_action_list,
            struct wolfsentry_action_list_ent ** cursor )
```

Open a cursor for the actions in an event. Caller must have a lock on the context at entry.

Parameters

event_label	the event label to open the iterator for
event_label_len	the length of the event_label
which_action_list	the action list of the event to list
cursor	a pointer to a pointer for the cursor to open

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_SHARED_OR_RETURN()

WOLFSENTRY_UNLOCK_AND_RETURN()

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.7 wolfsentry_event_action_prepend()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_prepend (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            wolfsentry_action_type_t which_action_list,
            const char * action_label,
            int action_label_len )
```

Prepend an action into an event.

Parameters

event_label	the label of the event to prepend the action into
event_label_len	the length of the event_label
which_action_list	the action list of the event to update
action_label	the label of the action to insert
action_label_len	the length of the action_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.8 wolfsentry_event_delete()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_delete (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            wolfsentry_action_res_t * action_results )
```

Delete an event from wolfsentry.

Parameters

label	the label of the even to delete
label_len	the length of the label
action_results	the result of the delete action

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.6.3.9 wolfsentry_event_drop_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_drop_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            struct wolfsentry_event * event,
            wolfsentry_action_res_t * action_results )
```

Drop a reference to an event.

Parameters

event	the event to drop the reference for
action_results	a pointer to the result of the function

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.10 wolfsentry_event_flush_all()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_flush_all (
            WOLFSENTRY_CONTEXT_ARGS_IN  )
```

Flush all events from wolfsentry.

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.11 wolfsentry_event_get_config()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_config (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            struct wolfsentry_eventconfig * config )
```

Get the configuration for an event.

Parameters

label	the label for the event to get the config for
label_len	the length of the label
config	the configuration returned

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.12 wolfsentry_event_get_flags()

```
WOLFSENTRY_API wolfsentry_event_flags_t wolfsentry_event_get_flags (
            const struct wolfsentry_event * event )
```

Get the flags for an event.

Parameters

event	the event to get the flags for

Returns

the current flags of the event

8.6.3.13 wolfsentry_event_get_label()

```
WOLFSENTRY_API const char * wolfsentry_event_get_label (
            const struct wolfsentry_event * event )
```

Get the label for an event. This is the internal pointer to the label so should not be freed by the application.

Parameters

event	the event to get the label for

Returns

the label for the event

8.6.3.14 wolfsentry_event_get_reference()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_reference (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            struct wolfsentry_event ** event )
```

Get a reference to an event.

Parameters

label	the label of the event to get the reference for
label_len	the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string
event	a pointer to a pointer for the event returned

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.15 wolfsentry_event_insert()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_insert (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * label,
            int label_len,
            wolfsentry_priority_t priority,
            const struct wolfsentry_eventconfig * config,
            wolfsentry_event_flags_t flags,
            wolfsentry_ent_id_t * id )
```

Insert an event into wolfsentry.

Parameters

label	the label for the event
label_len	the length of the label
priority	the priorty of the event
config	event configuration details
flags	the flags for the event
id	the returned ID for the event

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.16 wolfsentry_event_set_aux_event()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_set_aux_event (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * event_label,
            int event_label_len,
            const char * aux_event_label,
            int aux_event_label_len )
```

Set an auxiliary event for an event.

Parameters

event_label	the parent event label
event_label_len	the length of the event_label
aux_event_label	the aux event label
aux_event_label_len	the length of the aux event_label

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.17 wolfsentry_event_update_config()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_update_config (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
```

```
    const char * label,
    int label_len,
    const struct wolfsentry_eventconfig * config )
```

Update the configuration for an event.

Parameters

label	the label for the event to get the config for
label_len	the length of the label
config	the updated configuration

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_CONTEXT_ARGS_IN

8.6.3.18 wolfsentry_eventconfig_check()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_check (
            const struct wolfsentry_eventconfig * config )
```

Checks the config for self-consistency and validity.

Parameters

config	the pointer to the config to check

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.6.3.19 wolfsentry_eventconfig_init()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_init (
            struct wolfsentry_context * wolfsentry,
            struct wolfsentry_eventconfig * config )
```

Initializes a wolfsentry_eventconfig struct with the defaults from the wolfsentry context. If no wolfsentry context is provided this will initialize to zero.

Parameters

wolfsentry	the wolfsentry context
config	the pointer to the config to initialize

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

8.7 Address Family Subsystem

Macros

- #define **WOLFSENTRY_AF_UNSPEC** 0
- #define **WOLFSENTRY_AF_UNIX** 1

  Unix domain sockets.
- #define **WOLFSENTRY_AF_LOCAL** 1

  POSIX name for WOLFSENTRY_AF_UNIX.
- #define **WOLFSENTRY_AF_INET** 2

  Internet IP Protocol.
- #define **WOLFSENTRY_AF_AX25** 3

  Amateur Radio AX.25.
- #define **WOLFSENTRY_AF_IPX** 4

  Novell IPX.
- #define **WOLFSENTRY_AF_APPLETALK** 5

  AppleTalk DDP.
- #define **WOLFSENTRY_AF_NETROM** 6

  Amateur Radio NET/ROM.
- #define **WOLFSENTRY_AF_BRIDGE** 7

  Multiprotocol bridge.
- #define **WOLFSENTRY_AF_ATMPVC** 8

  ATM PVCs.
- #define **WOLFSENTRY_AF_X25** 9

  Reserved for X.25 project.
- #define **WOLFSENTRY_AF_INET6** 10

  IP version 6.
- #define **WOLFSENTRY_AF_ROSE** 11

  Amateur Radio X.25 PLP.
- #define **WOLFSENTRY_AF_DECnet** 12

  Reserved for DECnet project.
- #define **WOLFSENTRY_AF_NETBEUI** 13

  Reserved for 802.2LLC project.
- #define **WOLFSENTRY_AF_SECURITY** 14

  Security callback pseudo AF.
- #define **WOLFSENTRY_AF_KEY** 15

  PF_KEY key management API.
- #define **WOLFSENTRY_AF_NETLINK** 16
- #define **WOLFSENTRY_AF_ROUTE** WOLFSENTRY_AF_NETLINK

  Alias to emulate 4.4BSD.
- #define **WOLFSENTRY_AF_PACKET** 17

  Packet family.
- #define **WOLFSENTRY_AF_ASH** 18

  Ash.
- #define **WOLFSENTRY_AF_ECONET** 19

  Acorn Econet.
- #define **WOLFSENTRY_AF_ATMSVC** 20

  ATM SVCs.
- #define **WOLFSENTRY_AF_RDS** 21

  RDS sockets.
- #define **WOLFSENTRY_AF_SNA** 22

Linux SNA Project (nutters!)

- #define **WOLFSENTRY_AF_IRDA** 23

  IRDA sockets.
- #define **WOLFSENTRY_AF_PPPOX** 24

  PPPoX sockets.
- #define **WOLFSENTRY_AF_WANPIPE** 25

  Wanpipe API Sockets.
- #define **WOLFSENTRY_AF_LLC** 26

  Linux LLC.
- #define **WOLFSENTRY_AF_IB** 27

  Native InfiniBand address.
- #define **WOLFSENTRY_AF_MPLS** 28

  MPLS.
- #define **WOLFSENTRY_AF_CAN** 29

  Controller Area Network.
- #define **WOLFSENTRY_AF_TIPC** 30

  TIPC sockets.
- #define **WOLFSENTRY_AF_BLUETOOTH** 31

  Bluetooth sockets.
- #define **WOLFSENTRY_AF_IUCV** 32

  IUCV sockets.
- #define **WOLFSENTRY_AF_RXRPC** 33

  RxRPC sockets.
- #define **WOLFSENTRY_AF_ISDN** 34

  mISDN sockets
- #define **WOLFSENTRY_AF_PHONET** 35

  Phonet sockets.
- #define **WOLFSENTRY_AF_IEEE802154** 36

  IEEE802154 sockets.
- #define **WOLFSENTRY_AF_CAIF** 37

  CAIF sockets.
- #define **WOLFSENTRY_AF_ALG** 38

  Algorithm sockets.
- #define **WOLFSENTRY_AF_NFC** 39

  NFC sockets.
- #define **WOLFSENTRY_AF_VSOCK** 40

  vSockets
- #define **WOLFSENTRY_AF_KCM** 41

  Kernel Connection Multiplexor.
- #define **WOLFSENTRY_AF_QIPCRTR** 42

  Qualcomm IPC Router.
- #define **WOLFSENTRY_AF_SMC** 43

  smc sockets: reserve number for PF_SMC protocol family that reuses WOLFSENTRY_AF_INET address family
- #define **WOLFSENTRY_AF_XDP** 44

  XDP sockets.
- #define **WOLFSENTRY_AF_BSD_OFFSET** 100

  from FreeBSD at commit a56e5ad6, except WOLFSENTRY_AF_LINK64, added here.
- #define **WOLFSENTRY_AF_IMPLINK** (WOLFSENTRY_AF_BSD_OFFSET + 3)

  arpanet imp addresses
- #define **WOLFSENTRY_AF_PUP** (WOLFSENTRY_AF_BSD_OFFSET + 4)

  pup protocols: e.g. BSP

- #define **WOLFSENTRY_AF_CHAOS** (WOLFSENTRY_AF_BSD_OFFSET + 5)

  mit CHAOS protocols
- #define **WOLFSENTRY_AF_NETBIOS** (WOLFSENTRY_AF_BSD_OFFSET + 6)

  SMB protocols.
- #define **WOLFSENTRY_AF_ISO** (WOLFSENTRY_AF_BSD_OFFSET + 7)

  ISO protocols.
- #define **WOLFSENTRY_AF_OSI** WOLFSENTRY_AF_ISO
- #define **WOLFSENTRY_AF_ECMA** (WOLFSENTRY_AF_BSD_OFFSET + 8)

  European computer manufacturers.
- #define **WOLFSENTRY_AF_DATAKIT** (WOLFSENTRY_AF_BSD_OFFSET + 9)

  datakit protocols
- #define **WOLFSENTRY_AF_DLI** (WOLFSENTRY_AF_BSD_OFFSET + 13)

  DEC Direct data link interface.
- #define **WOLFSENTRY_AF_LAT** (WOLFSENTRY_AF_BSD_OFFSET + 14)

  LAT.
- #define **WOLFSENTRY_AF_HYLINK** (WOLFSENTRY_AF_BSD_OFFSET + 15)

  NSC Hyperchannel.
- #define **WOLFSENTRY_AF_LINK48** (WOLFSENTRY_AF_BSD_OFFSET + 18)

  Link layer interface, explicit EUI-48.
- #define **WOLFSENTRY_AF_LINK** WOLFSENTRY_AF_LINK48

  Link layer interface, implicit EUI-48.
- #define **WOLFSENTRY_AF_LINK64** (WOLFSENTRY_AF_BSD_OFFSET + 19)

  Link layer interface, explicit EUI-64.
- #define **WOLFSENTRY_AF_COIP** (WOLFSENTRY_AF_BSD_OFFSET + 20)

  connection-oriented IP, aka ST II
- #define **WOLFSENTRY_AF_CNT** (WOLFSENTRY_AF_BSD_OFFSET + 21)

  Computer Network Technology.
- #define **WOLFSENTRY_AF_SIP** (WOLFSENTRY_AF_BSD_OFFSET + 24)

  Simple Internet Protocol.
- #define **WOLFSENTRY_AF_SLOW** (WOLFSENTRY_AF_BSD_OFFSET + 33)

  802.3ad slow protocol
- #define **WOLFSENTRY_AF_SCLUSTER** (WOLFSENTRY_AF_BSD_OFFSET + 34)

  Sitara cluster protocol.
- #define **WOLFSENTRY_AF_ARP** (WOLFSENTRY_AF_BSD_OFFSET + 35)
- #define **WOLFSENTRY_AF_IEEE80211** (WOLFSENTRY_AF_BSD_OFFSET + 37)

  IEEE 802.11 protocol.
- #define **WOLFSENTRY_AF_INET_SDP** (WOLFSENTRY_AF_BSD_OFFSET + 40)

  OFED Socket Direct Protocol ipv4.
- #define **WOLFSENTRY_AF_INET6_SDP** (WOLFSENTRY_AF_BSD_OFFSET + 42)

  OFED Socket Direct Protocol ipv6.
- #define **WOLFSENTRY_AF_HYPERV** (WOLFSENTRY_AF_BSD_OFFSET + 43)

  HyperV sockets.
- #define **WOLFSENTRY_AF_USER_OFFSET** 256

Typedefs

- typedef wolfsentry_errcode_t(∗ **wolfsentry_addr_family_parser_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, const char ∗addr_text, int addr_text_len, byte ∗addr_internal, wolfsentry_addr_bits_t ∗addr_internal_bits)

  Function type for parsing handler, to pass to wolfsentry_addr_family_handler_install()
- typedef wolfsentry_errcode_t(∗ **wolfsentry_addr_family_formatter_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, const byte ∗addr_internal, unsigned int addr_internal_bits, char ∗addr_text, int ∗addr_text_len)

  Function type for formatting handler, to pass to wolfsentry_addr_family_handler_install()

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_install** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family_bynumber, const char *family_byname, int family_byname_len, wolfsentry_addr_family_parser parser, wolfsentry_addr_family_formatter_t formatter, int max_addr_bits)

  Install handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_get_parser** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_family_parser_t *parser)

  Retrieve the parsing handler for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_get_formatter** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_family_formatter_t *formatter)

  Retrieve the formatting handler for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_remove_bynumber** (WOLFSENTRY_CONTEX wolfsentry_addr_family_t family_bynumber, wolfsentry_action_res_t *action_results)

  Remove the handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_drop_reference** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_addr_family_bynumber *family_bynumber, wolfsentry_action_res_t *action_results)

  Release an address family record previously returned by wolfsentry_addr_family_ntop()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_remove_byname** (WOLFSENTRY_CONTEXT_ const char *family_byname, int family_byname_len, wolfsentry_action_res_t *action_results)

  Remove the handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_pton** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *family_name, int family_name_len, wolfsentry_addr_family_t *family_number)

  Look up an address family by name, returning its number.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_ntop** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, struct wolfsentry_addr_family_bynumber **addr_family, const char **family_name)

  Look up an address family by number, returning a pointer to its name. The caller must release `addr_family`, using wolfsentry_addr_family_drop_reference(), when done accessing `family_name`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_max_addr_bits** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_bits_t *bits)

  Look up the max address size for an address family identified by number.

8.7.1 Detailed Description

8.8 User-Defined Value Subsystem

Data Structures

- struct wolfsentry_kv_pair

  public structure for passing user-defined values in/out of wolfSentry

Macros

- #define **WOLFSENTRY_KV_FLAG_MASK**

  A bit mask to retain only the flag bits in a `wolfsentry_kv_type_t`.
- #define **WOLFSENTRY_KV_KEY_LEN**(kv)

  Evaluates to the length of the key of a `wolfsentry_kv_pair`.
- #define **WOLFSENTRY_KV_KEY**(kv)

  Evaluates to the key of a `wolfsentry_kv_pair`.

- #define **WOLFSENTRY_KV_TYPE**(kv)

  Evaluates to the type of a `wolfsentry_kv_pair`, with flag bits masked out.
- #define **WOLFSENTRY_KV_V_UINT**(kv)

  Evaluates to the `uint64_t` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_UINT`.
- #define **WOLFSENTRY_KV_V_SINT**(kv)

  Evaluates to the `int64_t` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_INT`.
- #define **WOLFSENTRY_KV_V_FLOAT**(kv)

  Evaluates to the `double` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_FLOAT`.
- #define **WOLFSENTRY_KV_V_STRING_LEN**(kv)

  Evaluates to the `size_t` length of the value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_STRING`.
- #define **WOLFSENTRY_KV_V_STRING**(kv)

  Evaluates to the `char *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_STRING`.
- #define **WOLFSENTRY_KV_V_BYTES_LEN**(kv)

  Evaluates to the `size_t` length of the value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.
- #define **WOLFSENTRY_KV_V_BYTES**(kv)

  Evaluates to the `byte *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.
- #define **WOLFSENTRY_KV_V_JSON**(kv)

  Evaluates to the `JSON_VALUE *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_JSON`.
- #define **WOLFSENTRY_BASE64_DECODED_BUFSPC**(buf, len)

  Given valid base64 string `buf` of length `len`, evaluates to the exact decoded length.

Typedefs

- typedef wolfsentry_errcode_t(* wolfsentry_kv_validator_t) (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair *kv)

Enumerations

- enum wolfsentry_kv_type_t {
  WOLFSENTRY_KV_NONE = 0 ,
  WOLFSENTRY_KV_NULL ,
  WOLFSENTRY_KV_TRUE ,
  WOLFSENTRY_KV_FALSE ,
  WOLFSENTRY_KV_UINT ,
  WOLFSENTRY_KV_SINT ,
  WOLFSENTRY_KV_FLOAT ,
  WOLFSENTRY_KV_STRING ,
  WOLFSENTRY_KV_BYTES ,
  WOLFSENTRY_KV_JSON ,
  WOLFSENTRY_KV_FLAG_READONLY = 1<<30 }

  enum to represent the type of a user-defined value

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_set_validator** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_kv_validator_t validator, wolfsentry_action_res_t *action_results)

  Install a supplied `wolfsentry_kv_validator_t` to validate all user values before inserting them into the value table.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_set_mutability** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int mutable)

Set the user-defined value with the designated `key` as readwrite (`mutable=1`) or readonly (`mutable=0`). A read-only value cannot be changed or deleted.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_mutability** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int *mutable)

  Query the mutability of the user-defined value with the designated `key`. Readonly value cannot be changed or deleted.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_type** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t *type)

  Returns the type of the value with the designated `key`, using `WOLFSENTRY_KV_TYPE()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_delete** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len)

  Deletes the value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_null** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_NULL` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bool** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_TRUE` or `WOLFSENTRY_KV_FALSE` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_bool** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t *value)

  Gets a `WOLFSENTRY_KV_TRUE` or `WOLFSENTRY_KV_FALSE` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_uint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, uint64_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_uint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, uint64_t *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_sint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int64_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_SINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_sint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int64_t *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_double** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, double value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_FLOAT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_float** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, double *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_string** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const char *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_STRING` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_string (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const char **value, int *value_len, struct wolfsentry_kv_pair_internal **user↩ _value_record)

  Gets a `WOLFSENTRY_KV_STRING` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bytes** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const byte *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_BYTES` value with the designated `key` and a binary-clean `value`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bytes_base64** (WOLFSENTRY_CONTEXT_ARGS const char *key, int key_len, const char *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_BYTES` value with the designated `key` and a base64-encoded `value`.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bytes (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const byte **value, int *value_len, struct wolfsentry_kv_pair_internal **user_value_record)

  Gets a `WOLFSENTRY_KV_BYTES` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_json** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, JSON_VALUE *value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_JSON` value with the designated `key` and a `value` from `json_dom_parse()` (or built up programmatically with the `centijson_value.h` API).
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_json (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, JSON_VALUE **value, struct wolfsentry_kv_pair_internal **user_value_record)

  Gets a `WOLFSENTRY_KV_JSON` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_release_record** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair_internal **user_value_record)

  Release a `user_value_record` from `wolfsentry_user_value_get_string()`, `wolfsentry_user_value_get_by` or `wolfsentry_user_value_get_json()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_pair_export** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair_internal *kv, const struct wolfsentry_kv_pair **kv_exports)

  Extract the `struct wolfsentry_kv_pair` from a `struct wolfsentry_kv_pair_internal`. Caller must have a shared or exclusive lock on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_type_to_string** (wolfsentry_kv_type_t type, const char **out)

  Return a human-readable rendering of a `wolfsentry_kv_type_t`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_render_value** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_kv_pair *kv, char *out, int *out_len)

  Render `kv` in human-readable form to caller-preallocated buffer `out`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_start** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor **cursor)

  Start an iteration loop on the user values table of this context. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_seek_to_head** (WOLFSENTRY_CONTEXT_ARG struct wolfsentry_cursor *cursor)

  Move the cursor to point to the start of the user values table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_seek_to_tail** (WOLFSENTRY_CONTEXT_ARGS struct wolfsentry_cursor *cursor)

  Move the cursor to point to the end of the user values table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_current** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Return the item to which the cursor currently points, without moving the cursor. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_prev** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Move the cursor to the previous item, and return it. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_next** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Move the cursor to the next item, and return it. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_end** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor **cursor)

  End an iteration loop started with wolfsentry_user_values_iterate_start(). Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_base64_decode** (const char *src, size_t src_len, byte *dest, size_t *dest_spc, int ignore_junk_p)

  Convert base64-encoded input `src` to binary output `dest`, optionally ignoring (with nonzero `ignore_junk_p`) non-base64 characters in `src`.

8.8.1 Detailed Description

8.8.2 Typedef Documentation

8.8.2.1 wolfsentry_kv_validator_t

```
typedef wolfsentry_errcode_t(* wolfsentry_kv_validator_t) (WOLFSENTRY_CONTEXT_ARGS_IN, struct
wolfsentry_kv_pair *kv)
```

Function type for user-supplied value validators.

8.8.3 Function Documentation

8.8.3.1 wolfsentry_user_value_get_bytes()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bytes (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * key,
            int key_len,
            const byte ** value,
            int * value_len,
            struct wolfsentry_kv_pair_internal ** user_value_record )
```

Gets a `WOLFSENTRY_KV_BYTES` value with the designated `key`.

The `user_value_record` will be used to store a pointer to an internal structure, which acts as a lease on the `value`. This must be released with `wolfsentry_user_value_release_record()` when done.

8.8.3.2 wolfsentry_user_value_get_json()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_json (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * key,
            int key_len,
            JSON_VALUE ** value,
            struct wolfsentry_kv_pair_internal ** user_value_record )
```

Gets a `WOLFSENTRY_KV_JSON` value with the designated `key`.

The `user_value_record` will be used to store a pointer to an internal structure, which acts as a lease on the `value`. This must be released with `wolfsentry_user_value_release_record()` when done.

8.8.3.3 wolfsentry_user_value_get_string()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_string (
            WOLFSENTRY_CONTEXT_ARGS_IN ,
            const char * key,
            int key_len,
            const char ** value,
            int * value_len,
            struct wolfsentry_kv_pair_internal ** user_value_record )
```

Gets a `WOLFSENTRY_KV_STRING` value with the designated `key`.

The `user_value_record` will be used to store a pointer to an internal structure, which acts as a lease on the `value`. This must be released with `wolfsentry_user_value_release_record()` when done.

8.9 Object Subsystem

Typedefs

- typedef wolfsentry_errcode_t(∗ **wolfsentry_make_id_cb_t**) (void ∗context, wolfsentry_ent_id_t ∗id)

Enumerations

- enum wolfsentry_object_type_t {
  WOLFSENTRY_OBJECT_TYPE_UNINITED ,
  WOLFSENTRY_OBJECT_TYPE_TABLE ,
  WOLFSENTRY_OBJECT_TYPE_ACTION ,
  WOLFSENTRY_OBJECT_TYPE_EVENT ,
  WOLFSENTRY_OBJECT_TYPE_ROUTE ,
  WOLFSENTRY_OBJECT_TYPE_KV ,
  WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER ,
  WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME }

  enum for communicating the type of an object.

Functions

- WOLFSENTRY_API wolfsentry_object_type_t wolfsentry_get_object_type (const void ∗object)

  Get the object type from a wolfsentry object pointer.
- WOLFSENTRY_API wolfsentry_ent_id_t wolfsentry_get_object_id (const void ∗object)

  Get the ID from a wolfsentry object pointer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_table_ent_get_by_id** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, struct wolfsentry_table_ent_header ∗∗ent)

  Retrieve an object pointer given its ID. Lock must be obtained before entry, and ent is only valid while lock is held, or if wolfsentry_object_checkout() is called for the object.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_object_checkout** (WOLFSENTRY_CONTEXT_ARGS_IN, void ∗object)

  Increment the refcount for an object, making it safe from deallocation until wolfsentry_object_release(). Caller must have a context lock on entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_object_release** (WOLFSENTRY_CONTEXT_ARGS_IN, void ∗object, wolfsentry_action_res_t ∗action_results)

  Decrement the refcount for an object, deallocating it if no references remain. Caller does not need to have a context lock on entry.
- WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_inserts (struct wolfsentry_table_header ∗table)

  Get the number of inserts into a table.
- WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_deletes (struct wolfsentry_table_header ∗table)

  Get the number of deletes from a table.

8.9.1 Detailed Description

8.9.2 Enumeration Type Documentation

8.9.2.1 wolfsentry_object_type_t

```
enum wolfsentry_object_type_t
```

enum for communicating the type of an object.

Enumerator

WOLFSENTRY_OBJECT_TYPE_UNINITED	Object is null or uninitialized.
WOLFSENTRY_OBJECT_TYPE_TABLE	Not currently used.
WOLFSENTRY_OBJECT_TYPE_ACTION	Object is a `struct wolfsentry_action`.
WOLFSENTRY_OBJECT_TYPE_EVENT	Object is a `struct wolfsentry_event`.
WOLFSENTRY_OBJECT_TYPE_ROUTE	Object is a `struct wolfsentry_route`.
WOLFSENTRY_OBJECT_TYPE_KV	Object is a `struct wolfsentry_kv_pair_internal`.
WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER	Object is a `struct wolfsentry_addr_family_bynumber`.
WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME	Object is a `struct wolfsentry_addr_family_byname`.

8.9.3 Function Documentation

8.9.3.1 wolfsentry_get_object_id()

```
WOLFSENTRY_API wolfsentry_ent_id_t wolfsentry_get_object_id (
            const void * object )
```

Get the ID from a wolfsentry object pointer.

Parameters

object	a pointer to the object

Returns

the object ID, or WOLFSENTRY_OBJECT_TYPE_UNINITED on error.

8.9.3.2 wolfsentry_get_object_type()

```
WOLFSENTRY_API wolfsentry_object_type_t wolfsentry_get_object_type (
            const void * object )
```

Get the object type from a wolfsentry object pointer.

Parameters

object	a pointer to the object

Returns

the object type, or WOLFSENTRY_OBJECT_TYPE_UNINITED on error.

8.9.3.3 wolfsentry_table_n_deletes()

```
WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_deletes (
            struct wolfsentry_table_header * table )
```

Get the number of deletes from a table.

Parameters

table	the table to get the deletes for

Returns

the total delete count

8.9.3.4 wolfsentry_table_n_inserts()

```
WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_inserts (
            struct wolfsentry_table_header * table )
```

Get the number of inserts into a table.

Parameters

table	the table to get the inserts for

Returns

the total insert count

8.10 Thread Synchronization Subsystem

Data Structures

- struct wolfsentry_thread_context_public

  Right-sized, right-aligned opaque container for thread state.

Macros

- #define **WOLFSENTRY_CONTEXT_ARGS_IN**

  Common context argument generator for use at the beginning of arg lists in function prototypes and definitions. Pair with `WOLFSENTRY_CONTEXT_ARGS_OUT` in the caller argument list.
- #define **WOLFSENTRY_CONTEXT_ARGS_IN_EX**(ctx)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` that allows a fully type-qualified `context` to be supplied explicitly (allowing contexts other than `struct wolfsentry_context`)
- #define **WOLFSENTRY_CONTEXT_ARGS_IN_EX4**(ctx, thr)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` that allows the identifiers for `context` and `thread` pointers to be supplied explicitly.

- #define **WOLFSENTRY_CONTEXT_ELEMENTS**

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` for constructing `struct`s.
- #define **WOLFSENTRY_CONTEXT_SET_ELEMENTS**(s)

  Counterpart to `WOLFSENTRY_CONTEXT_ELEMENTS` to access the `wolfsentry` context.
- #define **WOLFSENTRY_CONTEXT_GET_ELEMENTS**(s)

  Counterpart to `WOLFSENTRY_CONTEXT_ELEMENTS` to access the `thread` context (exists only if `defined(WOLFSENTRY_THREADSAFE)`)
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT**

  Common context argument generator to use in calls to functions taking `WOLFSENTRY_CONTEXT_ARGS_IN`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX**(ctx)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` that allows passing an explicitly identified context argument generator to use in calls to functions taking `WOLFSENTRY_CONTEXT_ARGS_IN_EX`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX2**(x)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` corresponding to `WOLFSENTRY_CONTEXT_ELEMENTS`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX3**(x, y)

  Special-purpose variant of `WOLFSENTRY_CONTEXT_ARGS_OUT_EX` for accessing context element `y` in structure pointer `x`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX4**(x, y)

  Special-purpose variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` that simply expands to `x` or `x, y` depending on `WOLFSENTRY_THREADSAFE`
- #define **WOLFSENTRY_CONTEXT_ARGS_NOT_USED**

  Helper macro for function implementations that need to accept `WOLFSENTRY_CONTEXT_ARGS_IN` for API conformance, but don't actually use the arguments.
- #define **WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED**

  Helper macro for function implementations that need to accept `WOLFSENTRY_CONTEXT_ARGS_IN` for API conformance, but don't actually use the `thread` argument.
- #define **WOLFSENTRY_THREAD_HEADER_DECLS**

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack.
- #define **WOLFSENTRY_THREAD_HEADER_INIT**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this performs the required thread context initialization, with options from its `wolfsentry_thread_flags_t flags` arg.
- #define **WOLFSENTRY_THREAD_HEADER_INIT_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this performs the required thread context initialization, with options from its `wolfsentry_thread_flags_t flags` arg, and returns on failure.
- #define **WOLFSENTRY_THREAD_HEADER**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack, and initializes it with options from its `wolfsentry_thread_flags_t flags` arg.
- #define **WOLFSENTRY_THREAD_HEADER_CHECK**()

  For `WOLFSENTRY_THREADSAFE` applications, checks if thread context initialization succeeded, and returns on failure.
- #define **WOLFSENTRY_THREAD_HEADER_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack, and initializes it with options from its `wolfsentry_thread_flags_t flags` arg, returning on failure.
- #define **WOLFSENTRY_THREAD_TAILER**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this cleans up a thread context allocated with `WOLFSENTRY_THREAD_HEADER*`, with options from its `wolfsentry_thread_flags_t flags` arg, storing the result.
- #define **WOLFSENTRY_THREAD_TAILER_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this cleans up a thread context allocated with `WOLFSENTRY_THREAD_HEADER*`, with options from its `wolfsentry_thread_flags_t flags` arg, returning on error.
- #define **WOLFSENTRY_THREAD_GET_ERROR**

  For `WOLFSENTRY_THREADSAFE` applications, this evaluates to the most recent result from `WOLFSENTRY_THREAD_HEADER_INIT` or `WOLFSENTRY_THREAD_TAILER()`
- #define **WOLFSENTRY_DEADLINE_NEVER** (-1)

Value returned in `deadline->tv_sec` and `deadline->tv_nsec` by wolfsentry_get_thread_deadline() when `thread` has no deadline set. Not allowed as explicit values passed to wolfsentry_set_deadline_abs() – use wolfsentry_clear_deadline() to clear any deadline. Can be overridden with user settings.

- #define **WOLFSENTRY_DEADLINE_NOW** (-2)

  Value returned in `deadline->tv_sec` and `deadline->tv_nsec` by wolfsentry_get_thread_deadline() when `thread` is in non-blocking mode. Not allowed as explicit values passed to wolfsentry_set_deadline_abs() – use wolfsentry_set_deadline_rel_usecs(WOLFSENTRY_CONTEXT_ARGS_OUT, 0) to put thread in non-blocking mode. Can be overridden with user settings.
- #define **WOLFSENTRY_THREAD_NO_ID** 0
- #define **WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER** {0}

Enumerations

- enum wolfsentry_thread_flags_t {
  WOLFSENTRY_THREAD_FLAG_NONE ,
  WOLFSENTRY_THREAD_FLAG_DEADLINE ,
  WOLFSENTRY_THREAD_FLAG_READONLY }

  `wolfsentry_thread_flags_t` flags are to be `OR`ed together.
- enum wolfsentry_lock_flags_t {
  WOLFSENTRY_LOCK_FLAG_NONE ,
  WOLFSENTRY_LOCK_FLAG_PSHARED ,
  WOLFSENTRY_LOCK_FLAG_SHARED_ERROR_CHECKING ,
  WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_MUTEX ,
  WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_SHARED ,
  WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_TRY_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_AUTO_DOWNGRADE ,
  WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAPHORE }

  flags to pass to `wolfsentry_lock_*()` functions, to be `OR`d together

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_init_thread_context** (struct wolfsentry_thread_context *thread_context, wolfsentry_thread_flags_t init_thread_flags, void *user_context)

  Initialize `thread_context` according to `init_thread_flags`, storing `user_context` for later retrieval with wolfsentry_get_thread_user_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_alloc_thread_context** (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context, wolfsentry_thread_flags_t init_thread_flags, void *user_context)

  Allocate space for `thread_context` using the allocator in `hpi`, then call wolfsentry_init_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_id** (struct wolfsentry_thread_context *thread, wolfsentry_thread_id_t *id)

  Write the `wolfsentry_thread_id_t` of `thread` to `id`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_user_context** (struct wolfsentry_thread_context *thread, void **user_context)

  Store to `user_context` the pointer previously passed to wolfsentry_init_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_deadline** (struct wolfsentry_thread_context *thread, struct timespec *deadline)

  Store the deadline for `thread` to `deadline`, or if the thread has no deadline set, store WOLFSENTRY_DEADLINE_NEVER to `deadline->tv_sec` and `deadline->tv_nsec`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_flags** (struct wolfsentry_thread_context *thread, wolfsentry_thread_flags_t *thread_flags)

  Store the flags of `thread` to `thread_flags`.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_destroy_thread_context** (struct wolfsentry_thread↩ _context *thread_context, wolfsentry_thread_flags_t thread_flags)

  Perform final integrity checking on the thread state, and deallocate its ID.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_free_thread_context** (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context, wolfsentry_thread_flags_t thread_flags)

  Call `wolfsentry_destroy_thread_context()` on `*thread_context`, and if that succeeds, deallocate the thread object previously allocated by wolfsentry_alloc_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_deadline_rel_usecs** (WOLFSENTRY_CONTEXT_ARGS_IN, int usecs)

  Set the thread deadline to `usecs` in the future. The thread will not wait for a lock beyond that deadline.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_deadline_abs** (WOLFSENTRY_CONTEXT_ARGS_IN, time_t epoch_secs, long epoch_nsecs)

  Set the thread deadline to the time identified by `epoch_secs` and `epoch_nsecs`. The thread will not wait for a lock beyond that deadline.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_clear_deadline** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Clear any thread deadline previously set. On time-unbounded calls such as wolfsentry_lock_shared() and wolfsentry_lock_mutex(), the thread will sleep until the lock is available.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_thread_readonly** (struct wolfsentry_thread_↩ context *thread_context)

  Set the thread state to allow only readonly locks to be gotten, allowing multiple shared locks to be concurrently held. If any mutexes or reservations are currently held, the call will fail.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_thread_readwrite** (struct wolfsentry_thread_↩ context *thread_context)

  Set the thread state to allow both readonly and mutex locks to be gotten. If multiple shared locks are currently held, the call will fail.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_init (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock *lock, wolfsentry_lock_flags_t flags)

  This initializes a semaphore lock structure created by the user.
- WOLFSENTRY_API size_t **wolfsentry_lock_size** (void)
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_alloc (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock **lock, wolfsentry_lock_flags_t flags)

  Allocates and initializes a semaphore lock structure for use with wolfSentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Requests a shared lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Requests a shared lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Requests a shared lock with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex2shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Downgrade an exclusive lock to a shared lock.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Upgrade a shared lock to an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_reserve (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Attempt to reserve a upgrade of a shared lock to an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abandon (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Abandon a reservation of a lock upgrade from shared to exclusive.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in shared state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in exclusive state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_either (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in either shared or exclusive state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared2mutex_reservation (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if an upgrade reservation is held on the lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_get_flags (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t *flags)

  Extract the current flags from the lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_unlock (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Unlock a lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_destroy (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Destroy a lock that was created with wolfsentry_lock_init()
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_free (struct wolfsentry_rwlock **lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Destroy and free a lock that was created with wolfsentry_lock_alloc(). The lock's pointer will also be set to NULL.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_mutex() on the context.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_abstimed** (WOLFSENTRY_CONTEXT_ARGS_I const struct timespec *abs_timeout)

  Calls wolfsentry_lock_mutex_abstimed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_abstimed_ex** (WOLFSENTRY_CONTEXT_ARG const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  variant of wolfsentry_context_lock_mutex_abstimed() with a `flags` arg.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_mutex_timed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_timed_ex** (WOLFSENTRY_CONTEXT_ARGS_I wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  variant of wolfsentry_context_lock_mutex_timed() with a `flags` arg.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_shared() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_abstimed** (WOLFSENTRY_CONTEXT_ARGS_ const struct timespec *abs_timeout)

  Calls wolfsentry_lock_shared_abstimed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_with_reservation_abstimed** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct timespec *abs_timeout)

  Calls wolfsentry_lock_shared_abstimed() on the context, with the `WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO` flag.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_shared_timed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_with_reservation_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_shared_timed() on the context, with the `WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO` flag.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_unlock** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_unlock() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_unlock_and_abandon_reservation** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_unlock() on the context, with the `WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO` flag.

8.10.1 Detailed Description

8.10.2 Enumeration Type Documentation

8.10.2.1 wolfsentry_lock_flags_t

`enum wolfsentry_lock_flags_t`

flags to pass to `wolfsentry_lock_*()` functions, to be `OR`d together

Enumerator

WOLFSENTRY_LOCK_FLAG_NONE	Default lock behavior.
WOLFSENTRY_LOCK_FLAG_PSHARED	Initialize lock to be shared between processes (currently not used, only allowed by wolfsentry_lock_init(), and only functional on POSIX targets)

Enumerator

WOLFSENTRY_LOCK_FLAG_SHARED_ERROR_↩ _CHECKING	Enables supplementary error checking on shared lock usage (not currently implemented)
WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_↩ MUTEX	Don't allow recursive mutex locking in this call.
WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_↩ SHARED	Don't allow recursive shared locking in this call.
WOLFSENTRY_LOCK_FLAG_GET_↩ RESERVATION_TOO	If a shared lock is gotten in this call, require that a mutex upgrade reservation also be gotten.
WOLFSENTRY_LOCK_FLAG_TRY_↩ RESERVATION_TOO	If a shared lock is gotten in this call, try to get a mutex upgrade reservation too.
WOLFSENTRY_LOCK_FLAG_ABANDON_↩ RESERVATION_TOO	In a call to wolfsentry_lock_unlock(), if a shared lock is released and a mutex upgrade reservation is held, drop it too.
WOLFSENTRY_LOCK_FLAG_AUTO_DOWNGRADE	In a call to wolfsentry_lock_unlock(), if a held mutex was previously gotten by an upgrade, and this release will restore the recursion depth at which the upgrade was gotten, downgrade to a shared lock.
WOLFSENTRY_LOCK_FLAG_RETAIN_↩ SEMAPHORE	For use in an interrupt handler: get an async-signal-safe mutex on the lock. Implicitly has `try` dynamics (immediate return).

8.10.2.2 wolfsentry_thread_flags_t

```
enum wolfsentry_thread_flags_t
```

`wolfsentry_thread_flags_t` flags are to be `OR`ed together.

Enumerator

WOLFSENTRY_THREAD_FLAG_NONE	Default and normal thread state.
WOLFSENTRY_THREAD_FLAG_DEADLINE	This thread currently has a deadline associated with it, and will not wait for a lock beyond that deadline.
WOLFSENTRY_THREAD_FLAG_READONLY	This thread can only get and hold shared locks.

8.10.3 Function Documentation

8.10.3.1 wolfsentry_lock_alloc()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_alloc (
            struct wolfsentry_host_platform_interface * hpi,
            struct wolfsentry_thread_context * thread,
            struct wolfsentry_rwlock ** lock,
            wolfsentry_lock_flags_t flags )
```

Allocates and initializes a semaphore lock structure for use with wolfSentry.

Parameters

hpi	the wolfsentry_host_platform_interface
thread	pointer to the wolfsentry_thread_context
lock	a pointer to a pointer to a lock structure to be allocated and initialized
flags	the initial wolfsentry_lock_flags_t

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_lock_init

wolfsentry_lock_free

WOLFSENTRY_ERROR_DECODE_ERROR_CODE()

8.10.3.2 wolfsentry_lock_destroy()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_destroy (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Destroy a lock that was created with wolfsentry_lock_init()

Parameters

lock	a pointer to the lock
thread	pointer to the wolfsentry_thread_context
flags	optional wolfsentry_lock_flags_t

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_lock_init

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.3 wolfsentry_lock_free()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_free (
            struct wolfsentry_rwlock ** lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Destroy and free a lock that was created with wolfsentry_lock_alloc(). The lock's pointer will also be set to NULL.

Parameters

lock	a pointer to a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_lock_alloc

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.4 wolfsentry_lock_get_flags()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_get_flags (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t * flags )
```

Extract the current flags from the lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.5 wolfsentry_lock_have_either()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_either (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Check if the lock is held in either shared or exclusive state.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

When decoded using `WOLFSENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX` if it is a held mutex lock, `WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK` if it is a held shared lock, `WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK` if the lock is valid but not held by the designated `thread`, or `WOLFSENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.6 wolfsentry_lock_have_mutex()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_mutex (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Check if the lock is held in exclusive state.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

When decoded using `WOLFSENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX` if it is a held mutex lock, `WOLFSENTRY_ERROR_ID_LACKING_MUTEX` if the lock is not in mutex state, `WOLFSENTRY_ERROR_ID_NOT_PERMITTED` if the mutex is held by another thread, or `WOLFSENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.7 wolfsentry_lock_have_shared()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Check if the lock is held in shared state.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

When decoded using `WOLFSENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK` if it is a held shared lock, `WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK` if the lock is valid but not held by the designated `thread`, or `WOLFSENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.8 wolfsentry_lock_have_shared2mutex_reservation()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared2mutex_reservation (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Check if an upgrade reservation is held on the lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

When decoded using WOLFSENTRY_ERROR_DECODE_ERROR_CODE(), WOLFSENTRY_ERROR_ID_OK if it is shared lock. Or WOLFSENTRY_ERROR_ID_NOT_OK if it is not a shared lock.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.9 wolfsentry_lock_init()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_init (
            struct wolfsentry_host_platform_interface * hpi,
            struct wolfsentry_thread_context * thread,
            struct wolfsentry_rwlock * lock,
            wolfsentry_lock_flags_t flags )
```

This initializes a semaphore lock structure created by the user.

Parameters

hpi	the wolfsentry_host_platform_interface
thread	pointer to the wolfsentry_thread_context
lock	a pointer to a lock structure to be initialized
flags	the initial wolfsentry_lock_flags_t

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_lock_alloc

wolfsentry_lock_destroy

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.10 wolfsentry_lock_mutex()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Requests an exclusive lock.

Parameters

lock	a pointer to the lock
thread	pointer to the wolfsentry_thread_context
flags	optional wolfsentry_lock_flags_t

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.11 wolfsentry_lock_mutex2shared()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex2shared (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Downgrade an exclusive lock to a shared lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.12 wolfsentry_lock_mutex_abstimed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_abstimed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            const struct timespec * abs_timeout,
            wolfsentry_lock_flags_t flags )
```

Requests an exclusive lock with an absolute timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
abs_timeout	the absolute timeout for the lock
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.13 wolfsentry_lock_mutex_timed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_timed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_time_t max_wait,
            wolfsentry_lock_flags_t flags )
```

Requests an exclusive lock with a relative timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
max_wait	how long to wait for the timeout
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.14 wolfsentry_lock_shared()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Requests a shared lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.15 wolfsentry_lock_shared2mutex()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Upgrade a shared lock to an exclusive lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.16 wolfsentry_lock_shared2mutex_abandon()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abandon (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Abandon a reservation of a lock upgrade from shared to exclusive.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.17 wolfsentry_lock_shared2mutex_abstimed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abstimed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            const struct timespec * abs_timeout,
            wolfsentry_lock_flags_t flags )
```

Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
abs_timeout	the absolute timeout for the lock
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.18 wolfsentry_lock_shared2mutex_redeem()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.19 wolfsentry_lock_shared2mutex_redeem_abstimed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_abstimed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            const struct timespec * abs_timeout,
            wolfsentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive with an absolute timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
abs_timeout	the absolute timeout for the lock
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.20 wolfsentry_lock_shared2mutex_redeem_timed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_timed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_time_t max_wait,
            wolfsentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive with a relative timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
max_wait	how long to wait for the timeout
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.21 wolfsentry_lock_shared2mutex_reserve()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_reserve (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Attempt to reserve a upgrade of a shared lock to an exclusive lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

wolfsentry_lock_shared2mutex_redeem
wolfsentry_lock_shared2mutex_redeem_abstimed
wolfsentry_lock_shared2mutex_redeem_timed
wolfsentry_lock_shared2mutex_abandon
WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.22 wolfsentry_lock_shared2mutex_timed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_timed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_time_t max_wait,
            wolfsentry_lock_flags_t flags )
```

Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
max_wait	how long to wait for the timeout
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.23 wolfsentry_lock_shared_abstimed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_abstimed (
            struct wolfsentry_rwlock * lock,
```

```
            struct wolfsentry_thread_context * thread,
            const struct timespec * abs_timeout,
            wolfsentry_lock_flags_t flags )
```

Requests a shared lock with an absolute timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
abs_timeout	the absolute timeout for the lock
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.24 wolfsentry_lock_shared_timed()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_timed (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_time_t max_wait,
            wolfsentry_lock_flags_t flags )
```

Requests a shared lock with a relative timeout.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
max_wait	how long to wait for the timeout
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.10.3.25 wolfsentry_lock_unlock()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_unlock (
            struct wolfsentry_rwlock * lock,
            struct wolfsentry_thread_context * thread,
            wolfsentry_lock_flags_t flags )
```

Unlock a lock.

Parameters

lock	a pointer to the lock
thread	pointer to the `wolfsentry_thread_context`
flags	optional `wolfsentry_lock_flags_t`

Returns

WOLFSENTRY_IS_SUCCESS(ret) is true on success.

See also

WOLFSENTRY_ERROR_DECODE_ERROR_CODE

8.11 Allocator (Heap) Functions and Callbacks

Data Structures

- struct wolfsentry_allocator

  Struct for passing shims that abstract the native implementation of the heap allocator.

Typedefs

- typedef void ∗(∗ **wolfsentry_malloc_cb_t**) (void ∗context, struct wolfsentry_thread_context ∗thread, size_t size)

  Pointer to malloc-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void(∗ **wolfsentry_free_cb_t**) (void ∗context, struct wolfsentry_thread_context ∗thread, void ∗ptr)

  Pointer to free-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void ∗(∗ **wolfsentry_realloc_cb_t**) (void ∗context, struct wolfsentry_thread_context ∗thread, void ∗ptr, size_t size)

  Pointer to realloc-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void ∗(∗ **wolfsentry_memalign_cb_t**) (void ∗context, struct wolfsentry_thread_context ∗thread, size_t alignment, size_t size)

  Pointer to memalign-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void(∗ **wolfsentry_free_aligned_cb_t**) (void ∗context, struct wolfsentry_thread_context ∗thread, void ∗ptr)

  Pointer to special-purpose free-like function, needed only if the `memalign` pointer in a `struct wolfsentry_allocator` is non-null. Can be same as routine supplied as `wolfsentry_free_cb_t`, or can be a separate routine, e.g. with special handling for pad bytes. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.

Functions

- WOLFSENTRY_API void * **wolfsentry_malloc** (WOLFSENTRY_CONTEXT_ARGS_IN, size_t size)

  Allocate `size` bytes using the `malloc` configured in the wolfSentry context.
- WOLFSENTRY_API_VOID **wolfsentry_free** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr)

  Free `ptr` using the `free` configured in the wolfSentry context.
- WOLFSENTRY_API void * **wolfsentry_realloc** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr, size_t size)

  Reallocate `ptr` to `size` bytes using the `realloc` configured in the wolfSentry context.
- WOLFSENTRY_API void * **wolfsentry_memalign** (WOLFSENTRY_CONTEXT_ARGS_IN, size_t alignment, size_t size)

  Allocate `size` bytes, aligned to `alignment`, using the `memalign` configured in the wolfSentry context.
- WOLFSENTRY_API_VOID **wolfsentry_free_aligned** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr)

  Free `ptr`, previously allocated with `wolfsentry_memalign()`, using the `free_aligned` configured in the wolfSentry context.
- WOLFSENTRY_API int **_wolfsentry_get_n_mallocs** (void)

  In library builds with `WOLFSENTRY_MALLOC_BUILTINS` and `WOLFSENTRY_MALLOC_DEBUG` defined, this returns the net number of allocations performed as of time of call. I.e., it returns zero iff all allocations have been freed.
- WOLFSENTRY_API struct wolfsentry_allocator * **wolfsentry_get_allocator** (struct wolfsentry_context *wolfsentry)

  Return a pointer to the `wolfsentry_allocator` associated with the supplied `wolfsentry_context`, mainly for passing to `json_init()`, `json_parse()`, `json_value_*()`, and `json_dom_*()`.

8.11.1 Detailed Description

8.12 Time Functions and Callbacks

Data Structures

- struct wolfsentry_timecbs

  Struct for passing shims that abstract the native implementation of time functions.

Typedefs

- typedef wolfsentry_errcode_t(* **wolfsentry_get_time_cb_t**) (void *context, wolfsentry_time_t *ts)

  Pointer to function that returns time denominated in `wolfsentry_time_t`. Takes an initial `context` arg, which can be ignored.
- typedef wolfsentry_time_t(* **wolfsentry_diff_time_cb_t**) (wolfsentry_time_t earlier, wolfsentry_time_t later)

  Pointer to function that subtracts `earlier` from `later`, returning the result.
- typedef wolfsentry_time_t(* **wolfsentry_add_time_cb_t**) (wolfsentry_time_t start_time, wolfsentry_time_t time_interval)

  Pointer to function that adds two `wolfsentry_time_t` times, returning the result.
- typedef wolfsentry_errcode_t(* **wolfsentry_to_epoch_time_cb_t**) (wolfsentry_time_t when, time_t *epoch_secs, long *epoch_nsecs)

  Pointer to function that converts a `wolfsentry_time_t` to seconds and nanoseconds since midnight UTC, 1970-Jan-1.
- typedef wolfsentry_errcode_t(* **wolfsentry_from_epoch_time_cb_t**) (time_t epoch_secs, long epoch_nsecs, wolfsentry_time_t *when)

  Pointer to function that converts seconds and nanoseconds since midnight UTC, 1970-Jan-1, to a `wolfsentry_time_t`.

- typedef wolfsentry_errcode_t(* **wolfsentry_interval_to_seconds_cb_t**) (wolfsentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs)

  Pointer to function that converts a `wolfsentry_time_t` expressing an interval to the corresponding seconds and nanoseconds.
- typedef wolfsentry_errcode_t(* **wolfsentry_interval_from_seconds_cb_t**) (time_t howlong_secs, long howlong_nsecs, wolfsentry_time_t *howlong)

  Pointer to function that converts seconds and nanoseconds expressing an interval to the corresponding `wolfsentry_time_t`.

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_now_plus_delta** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t td, wolfsentry_time_t *res)

  Generate a wolfsentry_time_t at a given offset from current time.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_to_timespec** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t t, struct timespec *ts)

  Convert a wolfsentry_time_t to a `struct timespec`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_now_plus_delta_timespec** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t td, struct timespec *ts)

  Generate a `struct timespec` at a given offset, supplied as wolfsentry_time_t, from current time.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t *time_p)

  Get current time as wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_time_t **wolfsentry_diff_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t later, wolfsentry_time_t earlier)

  Compute the interval between `later` and `earlier`, using wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_time_t **wolfsentry_add_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t start_time, wolfsentry_time_t time_interval)

  Compute the time `time_interval` after `start_time`, using wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_to_epoch_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t when, time_t *epoch_secs, long *epoch_nsecs)

  Convert a wolfsentry_time_t to seconds and nanoseconds since 1970-Jan-1 0:00 UTC.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_from_epoch_time** (struct wolfsentry_context *wolfsentry, time_t epoch_secs, long epoch_nsecs, wolfsentry_time_t *when)

  Convert seconds and nanoseconds since 1970-Jan-1 0:00 UTC to a wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_interval_to_seconds** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs)

  Convert an interval in wolfsentry_time_t to seconds and nanoseconds.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_interval_from_seconds** (struct wolfsentry_context *wolfsentry, time_t howlong_secs, long howlong_nsecs, wolfsentry_time_t *howlong)

  Convert an interval in seconds and nanoseconds to wolfsentry_time_t.
- WOLFSENTRY_API struct wolfsentry_timecbs * **wolfsentry_get_timecbs** (struct wolfsentry_context *wolfsentry)

  Return the active time handlers from the supplied context.

8.12.1 Detailed Description

8.13 Semaphore Function Callbacks

Data Structures

- struct wolfsentry_semcbs

  Struct for passing shims that abstract the native implementation of counting semaphores.

Typedefs

- typedef int(∗ sem_init_cb_t) (sem_t ∗sem, int pshared, unsigned int value)
- typedef int(∗ sem_post_cb_t) (sem_t ∗sem)
- typedef int(∗ sem_wait_cb_t) (sem_t ∗sem)
- typedef int(∗ sem_timedwait_cb_t) (sem_t ∗sem, const struct timespec ∗abs_timeout)
- typedef int(∗ sem_trywait_cb_t) (sem_t ∗sem)
- typedef int(∗ sem_destroy_cb_t) (sem_t ∗sem)

8.13.1 Detailed Description

8.13.2 Typedef Documentation

8.13.2.1 sem_destroy_cb_t

```
typedef int(* sem_destroy_cb_t) (sem_t *sem)
```

Pointer to function with arguments and semantics of POSIX `sem_destroy()`

8.13.2.2 sem_init_cb_t

```
typedef int(* sem_init_cb_t) (sem_t *sem, int pshared, unsigned int value)
```

Pointer to function with arguments and semantics of POSIX `sem_init()`. Currently, `pshared` and `value` are always zero as called by wolfSentry, so implementations can ignore them.

8.13.2.3 sem_post_cb_t

```
typedef int(* sem_post_cb_t) (sem_t *sem)
```

Pointer to function with arguments and semantics of POSIX `sem_post()`

8.13.2.4 sem_timedwait_cb_t

```
typedef int(* sem_timedwait_cb_t) (sem_t *sem, const struct timespec *abs_timeout)
```

Pointer to function with arguments and semantics of POSIX `sem_timedwait()`

8.13.2.5 sem_trywait_cb_t

```
typedef int(* sem_trywait_cb_t) (sem_t *sem)
```

Pointer to function with arguments and semantics of POSIX `sem_trywait()`

8.13.2.6 sem_wait_cb_t

```
typedef int(* sem_wait_cb_t) (sem_t *sem)
```

Pointer to function with arguments and semantics of POSIX `sem_wait()`

8.14 lwIP Callback Activation Functions

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_ethernet_callback** (WOLFSENTRY_CONTEXT_AR packet_filter_event_mask_t ethernet_mask)

  Install wolfSentry callbacks into lwIP for ethernet (layer 2) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_ip_callbacks** (WOLFSENTRY_CONTEXT_ARGS_ packet_filter_event_mask_t ip_mask)

  Install wolfSentry callbacks into lwIP for IPv4/IPv6 (layer 3) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_icmp_callbacks** (WOLFSENTRY_CONTEXT_ARG packet_filter_event_mask_t icmp_mask)

  Install wolfSentry callbacks into lwIP for ICMP filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_tcp_callback** (WOLFSENTRY_CONTEXT_ARGS_ packet_filter_event_mask_t tcp_mask)

  Install wolfSentry callbacks into lwIP for TCP (layer 4) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_udp_callback** (WOLFSENTRY_CONTEXT_ARGS_ packet_filter_event_mask_t udp_mask)

  Install wolfSentry callbacks into lwIP for UDP (layer 4) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_callbacks** (WOLFSENTRY_CONTEXT_ARGS_IN, packet_filter_event_mask_t ethernet_mask, packet_filter_event_mask_t ip_mask, packet_filter_event_↩ mask_t icmp_mask, packet_filter_event_mask_t tcp_mask, packet_filter_event_mask_t udp_mask)

  Install wolfSentry callbacks for all layers/protocols enabled by the supplied masks.
- WOLFSENTRY_API_VOID **wolfsentry_cleanup_lwip_filter_callbacks** (WOLFSENTRY_CONTEXT_ARGS_IN, void *arg)

  Disables any wolfSentry callbacks previously installed in lwIP.

8.14.1 Detailed Description

Chapter 9

Data Structure Documentation

9.1 JSON_CALLBACKS Struct Reference

Data Fields

- int(∗ **process**)(JSON_TYPE, const unsigned char ∗, size_t, void ∗)

9.2 JSON_CONFIG Struct Reference

Data Fields

- size_t **max_total_len**
- size_t **max_total_values**
- size_t **max_number_len**
- size_t **max_string_len**
- size_t **max_key_len**
- unsigned **max_nesting_level**
- unsigned **flags**

9.3 JSON_DOM_PARSER Struct Reference

Data Fields

- JSON_PARSER **parser**
- JSON_VALUE ∗∗ **path**
- size_t **path_size**
- size_t **path_alloc**
- JSON_VALUE **root**
- JSON_VALUE **key**
- unsigned **flags**
- unsigned **dict_flags**

9.4 JSON_INPUT_POS Struct Reference

Data Fields

- size_t **offset**
- unsigned **line_number**
- unsigned **column_number**

9.5 JSON_PARSER Struct Reference

Public Types

- enum **centijson_automaton** {
  AUTOMATON_MAIN = 0 ,
  AUTOMATON_NULL = 1 ,
  AUTOMATON_FALSE = 2 ,
  AUTOMATON_TRUE = 3 ,
  AUTOMATON_NUMBER = 4 ,
  AUTOMATON_STRING = 6 ,
  AUTOMATON_KEY = 7 }

Data Fields

- JSON_CALLBACKS **callbacks**
- JSON_CONFIG **config**
- void * **user_data**
- JSON_INPUT_POS **pos**
- JSON_INPUT_POS **value_pos**
- JSON_INPUT_POS **err_pos**
- int **errcode**
- size_t **value_counter**
- unsigned char * **nesting_stack**
- size_t **nesting_level**
- size_t **nesting_stack_size**
- enum JSON_PARSER::centijson_automaton **automaton**
- unsigned **state**
- unsigned **substate**
- uint32_t **codepoint** [2]
- unsigned char * **buf**
- size_t **buf_used**
- size_t **buf_alloced**
- size_t **last_cl_offset**

9.6 JSON_VALUE Struct Reference

Data Fields

- union {
    uint8_t **data_bytes** [16]
    void * **data_ptrs** [16/sizeof(void *)]
  } **data**

9.7 wolfsentry_allocator Struct Reference

Struct for passing shims that abstract the native implementation of the heap allocator.

`#include <wolfsentry.h>`

Data Fields

- void * **context**

  A user-supplied opaque handle to be passed as the first arg to all callbacks. Can be null.
- wolfsentry_malloc_cb_t **malloc**

  Required pointer.
- wolfsentry_free_cb_t **free**

  Required pointer.
- wolfsentry_realloc_cb_t **realloc**

  Required pointer.
- wolfsentry_memalign_cb_t **memalign**

  Optional pointer. Required only if a `struct wolfsentry_eventconfig` is passed in (e.g. to wolfsentry_init()) with a nonzeroroute_private_data_alignment`.
- wolfsentry_free_aligned_cb_t **free_aligned**

  Optional pointer. Required (and allowed) only if `memalign` pointer is non-null.

9.7.1 Detailed Description

Struct for passing shims that abstract the native implementation of the heap allocator.

9.8 wolfsentry_build_settings Struct Reference

struct for passing the build version and configuration

`#include <wolfsentry_settings.h>`

Data Fields

- uint32_t version
- uint32_t config

9.8.1 Detailed Description

struct for passing the build version and configuration

9.8.2 Field Documentation

9.8.2.1 config

`uint32_t wolfsentry_build_settings::config`

Must be initialized to WOLFSENTRY_CONFIG_SIGNATURE.

9.8.2.2 version

`uint32_t wolfsentry_build_settings::version`

Must be initialized to WOLFSENTRY_VERSION.

9.9 wolfsentry_data Struct Reference

Public Member Functions

- **WOLFSENTRY_SOCKADDR** (128) remote
- **WOLFSENTRY_SOCKADDR** (128) local

Data Fields

- wolfsentry_route_flags_t **flags**
- void * **heap**
- int **alloctype**

9.10 wolfsentry_eventconfig Struct Reference

struct for representing event configuration

`#include <wolfsentry.h>`

Data Fields

- size_t **route_private_data_size**

  bytes to allocate for private use for application data
- size_t **route_private_data_alignment**

  alignment for private data allocation
- uint32_t **max_connection_count**

  If nonzero, the concurrent connection limit, beyond which additional connection requests are rejected.
- wolfsentry_hitcount_t **derogatory_threshold_for_penaltybox**

  If nonzero, the threshold at which accumulated derogatory counts (from `WOLFSENTRY_ACTION_RES_↩DEROGATORY` incidents) automatically penalty boxes a route.
- wolfsentry_time_t **penaltybox_duration**

  The duration that a route stays in penalty box status before automatic release. Zero means time-unbounded.
- wolfsentry_time_t **route_idle_time_for_purge**

  The time after the most recent dispatch match for a route to be garbage-collected. Zero means no automatic purge.
- wolfsentry_eventconfig_flags_t **flags**

  Config flags.
- wolfsentry_route_flags_t **route_flags_to_add_on_insert**

  List of route flags to set on new routes upon insertion.
- wolfsentry_route_flags_t **route_flags_to_clear_on_insert**

  List of route flags to clear on new routes upon insertion.
- wolfsentry_action_res_t **action_res_filter_bits_set**

  List of result flags that must be set at lookup time (dispatch) for referring routes to match.
- wolfsentry_action_res_t **action_res_filter_bits_unset**

  List of result flags that must be clear at lookup time (dispatch) for referring routes to match.
- wolfsentry_action_res_t **action_res_bits_to_add**

  List of result flags to be set upon match.
- wolfsentry_action_res_t **action_res_bits_to_clear**

  List of result flags to be cleared upon match.

9.10.1 Detailed Description

struct for representing event configuration

9.11 wolfsentry_host_platform_interface Struct Reference

struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores

`#include <wolfsentry.h>`

Data Fields

- struct wolfsentry_build_settings caller_build_settings
- struct wolfsentry_allocator allocator
- struct wolfsentry_timecbs timecbs
- struct wolfsentry_semcbs semcbs

9.11.1 Detailed Description

struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores

9.11.2 Field Documentation

9.11.2.1 allocator

`struct wolfsentry_allocator wolfsentry_host_platform_interface::allocator`

Either all-null, or initialized as described for `wolfsentry_allocator`.

9.11.2.2 caller_build_settings

`struct wolfsentry_build_settings wolfsentry_host_platform_interface::caller_build_settings`

Must be initialized as described for `wolfsentry_build_settings`.

9.11.2.3 semcbs

`struct wolfsentry_semcbs wolfsentry_host_platform_interface::semcbs`

Either all-null, or initialized as described for `wolfsentry_semcbs`.

9.11.2.4 timecbs

`struct wolfsentry_timecbs wolfsentry_host_platform_interface::timecbs`

Either all-null, or initialized as described for `wolfsentry_timecbs`.

9.12 wolfsentry_kv_pair Struct Reference

public structure for passing user-defined values in/out of wolfSentry

`#include <wolfsentry.h>`

Data Fields

- int **key_len**
  the length of the key, not including the terminating null
- wolfsentry_kv_type_t **v_type**
  the type of value
- union {
  uint64_t **v_uint**
  The value when `v_type` *is* `WOLFSENTRY_KV_UINT`
  int64_t **v_sint**
  The value when `v_type` *is* `WOLFSENTRY_KV_SINT`
  double **v_float**
  The value when `v_type` *is* `WOLFSENTRY_KV_FLOAT`
  size_t **string_len**
  The length of the value when `v_type` *is* `WOLFSENTRY_KV_STRING`
  size_t **bytes_len**
  The length of the value when `v_type` *is* `WOLFSENTRY_KV_BYTES`
  JSON_VALUE **v_json**
  The value when `v_type` *is* `WOLFSENTRY_KV_JSON`
  } **a**

- byte b [ ]
  A flexible-length buffer to hold the key, and for strings and bytes, the data.

9.12.1 Detailed Description

public structure for passing user-defined values in/out of wolfSentry

9.12.2 Field Documentation

9.12.2.1 b

`byte wolfsentry_kv_pair::b[]`

A flexible-length buffer to hold the key, and for strings and bytes, the data.

For atomic values and `WOLFSENTRY_KV_JSON`, this is just the key, with a terminating null at the end. For `WOLFSENTRY_KV_STRING` and `WOLFSENTRY_KV_BYTES`, the value itself appears right after the key with its terminating null.

9.13 wolfsentry_route_endpoint Struct Reference

struct for exporting socket addresses, with fixed-length fields

`#include <wolfsentry.h>`

Data Fields

- wolfsentry_port_t **sa_port**

  The port number – only treated as a TCP/IP port number if the route has the WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS flag set.
- wolfsentry_addr_bits_t **addr_len**

  The number of significant bits in the address. The address data itself is in the parent `wolfsentry_route_exports` struct.
- byte **extra_port_count**

  The number of extra ports in the route – not currently supported.
- byte **interface**

  The interface ID of the route.

9.13.1 Detailed Description

struct for exporting socket addresses, with fixed-length fields

9.14 wolfsentry_route_exports Struct Reference

struct for exporting a route for access by applications

```
#include <wolfsentry.h>
```

Data Fields

- const char * **parent_event_label**

  Label of the parent event, or null if none.
- int **parent_event_label_len**

  Length (not including terminating null) of label of the parent event, if any.
- wolfsentry_route_flags_t **flags**

  Current route flags (mutable bits are informational/approximate)
- wolfsentry_addr_family_t **sa_family**

  Address family for this route.
- wolfsentry_proto_t **sa_proto**

  Protocol for this route.
- struct wolfsentry_route_endpoint **remote**

  Remote socket address for this route.
- struct wolfsentry_route_endpoint **local**

  Local socket address for this route.
- const byte * **remote_address**

  Binary address data for the remote end of this route.
- const byte * **local_address**

  Binary address data for the local end of this route.
- const wolfsentry_port_t * **remote_extra_ports**

  array of extra remote ports that match this route – not yet implemented
- const wolfsentry_port_t * **local_extra_ports**

  array of extra local ports that match this route – not yet implemented
- struct wolfsentry_route_metadata_exports **meta**

  The current route metadata.
- void * **private_data**

  The private data segment (application-defined), if any.
- size_t **private_data_size**

  The size of the private data segment, if any, or zero.

9.14.1 Detailed Description

struct for exporting a route for access by applications

9.15 wolfsentry_route_metadata_exports Struct Reference

struct for exporting route metadata for access by applications

`#include <wolfsentry.h>`

Data Fields

- wolfsentry_time_t **insert_time**

  The time the route was inserted.
- wolfsentry_time_t **last_hit_time**

  The most recent time the route was matched.
- wolfsentry_time_t **last_penaltybox_time**

  The most recent time the route had its WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED flag set.
- wolfsentry_time_t **purge_after**

  The expiration time of the route, if any. Persistent routes have `0` here, and the setting can be modified with `wolfsentry_route_purge_time_set()`.
- uint16_t **connection_count**

  The current connection count (informational/approximate)
- uint16_t **derogatory_count**

  The current derogatory event count (informational/approximate)
- uint16_t **commendable_count**

  The current commendable event count (informational/approximate)
- wolfsentry_hitcount_t **hit_count**

  The lifetime match count (informational/approximate, and only maintained if the WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_HITS flag is clear)

9.15.1 Detailed Description

struct for exporting route metadata for access by applications

9.16 wolfsentry_semcbs Struct Reference

Struct for passing shims that abstract the native implementation of counting semaphores.

`#include <wolfsentry.h>`

Data Fields

- sem_init_cb_t **sem_init**
  Required pointer.
- sem_post_cb_t **sem_post**
  Required pointer.
- sem_wait_cb_t **sem_wait**
  Required pointer.
- sem_timedwait_cb_t **sem_timedwait**
  Required pointer.
- sem_trywait_cb_t **sem_trywait**
  Required pointer.
- sem_destroy_cb_t **sem_destroy**
  Required pointer.

9.16.1 Detailed Description

Struct for passing shims that abstract the native implementation of counting semaphores.

9.17 wolfsentry_sockaddr Struct Reference

struct for passing socket addresses into `wolfsentry_route_*()` API routines

`#include <wolfsentry.h>`

Data Fields

- wolfsentry_addr_family_t **sa_family**
  Address family number.
- wolfsentry_proto_t **sa_proto**
  Protocol number.
- wolfsentry_port_t **sa_port**
  Port number.
- wolfsentry_addr_bits_t **addr_len**
  Significant bits in address.
- byte **interface**
  Interface ID number.
- byte **addr** [ ]
  Binary big-endian address data.

9.17.1 Detailed Description

struct for passing socket addresses into `wolfsentry_route_*()` API routines

9.18 wolfsentry_thread_context_public Struct Reference

Right-sized, right-aligned opaque container for thread state.

`#include <wolfsentry_settings.h>`

Data Fields

- uint64_t **opaque** [8]

9.18.1 Detailed Description

Right-sized, right-aligned opaque container for thread state.

9.19 wolfsentry_timecbs Struct Reference

Struct for passing shims that abstract the native implementation of time functions.

`#include <wolfsentry.h>`

Data Fields

- void * **context**

  A user-supplied opaque handle to be passed as the first arg to the `get_time` callback. Can be null.
- wolfsentry_get_time_cb_t **get_time**

  Required pointer.
- wolfsentry_diff_time_cb_t **diff_time**

  Required pointer.
- wolfsentry_add_time_cb_t **add_time**

  Required pointer.
- wolfsentry_to_epoch_time_cb_t **to_epoch_time**

  Required pointer.
- wolfsentry_from_epoch_time_cb_t **from_epoch_time**

  Required pointer.
- wolfsentry_interval_to_seconds_cb_t **interval_to_seconds**

  Required pointer.
- wolfsentry_interval_from_seconds_cb_t **interval_from_seconds**

  Required pointer.

9.19.1 Detailed Description

Struct for passing shims that abstract the native implementation of time functions.

Chapter 10

File Documentation

10.1 centijson_dom.h

```
00001 /*
00002  * centijson_dom.h
00003  *
00004  * Copyright (C) 2022-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00023 /*
00024  * CentiJSON
00025  * <http://github.com/mity/centijson>
00026  *
00027  * Copyright (c) 2018 Martin Mitas
00028  *
00029  * Permission is hereby granted, free of charge, to any person obtaining a
00030  * copy of this software and associated documentation files (the "Software"),
00031  * to deal in the Software without restriction, including without limitation
00032  * the rights to use, copy, modify, merge, publish, distribute, sublicense,
00033  * and/or sell copies of the Software, and to permit persons to whom the
00034  * Software is furnished to do so, subject to the following conditions:
00035  *
00036  * The above copyright notice and this permission notice shall be included in
00037  * all copies or substantial portions of the Software.
00038  *
00039  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
00040  * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00041  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00042  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00043  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
00044  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
00045  * IN THE SOFTWARE.
00046  */
00047
00048 #ifndef JSON_DOM_H
00049 #define JSON_DOM_H
00050
00051 #include "wolfsentry/centijson_sax.h"
00052 #include "wolfsentry/centijson_value.h"
00053
00054 #ifdef __cplusplus
00055 extern "C" {
00056 #endif
00057
00058
```

```
00059 /* DOM-specific error codes
00060  *
00061  * The DOM paring functions can return any from json.h and additionally these.
00062  */
00063 #define JSON_DOM_ERR_DUPKEY             (-1000)
00064
00065
00066 /* Flags for json_dom_init()
00067  */
00068
00069 /* Policy how to deal if the JSON contains object with duplicate key: */
00070 #define JSON_DOM_DUPKEY_ABORT           0x0000U
00071 #define JSON_DOM_DUPKEY_USEFIRST        0x0001U
00072 #define JSON_DOM_DUPKEY_USELAST         0x0002U
00073
00074 #define JSON_DOM_DUPKEY_MASK                                            \
00075             (JSON_DOM_DUPKEY_ABORT | JSON_DOM_DUPKEY_USEFIRST | JSON_DOM_DUPKEY_USELAST)
00076
00077 /* When creating JSON_VALUE_DICT (for JSON_OBJECT), use flag JSON_VALUE_DICT_MAINTAINORDER. */
00078 #define JSON_DOM_MAINTAINDICTORDER      0x0010U
00079
00080 /* Internal use */
00081 #define JSON_DOM_FLAG_INITED            0x8000U
00082
00083 /* Structure holding parsing state. Do not access it directly.
00084  */
00085 typedef struct JSON_DOM_PARSER {
00086     JSON_PARSER parser;
00087     JSON_VALUE** path;
00088     size_t path_size;
00089     size_t path_alloc;
00090     JSON_VALUE root;
00091     JSON_VALUE key;
00092     unsigned flags;
00093     unsigned dict_flags;
00094 } JSON_DOM_PARSER;
00095
00096
00097 /* Used internally by load_config.c:handle_user_value_clause() */
00098 int json_dom_init_1(
00099 #ifdef WOLFSENTRY
00100     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00101 #endif
00102     JSON_DOM_PARSER* dom_parser, unsigned dom_flags);
00103
00104 /* Used internally by load_config.c:handle_user_value_clause() */
00105 int json_dom_process(JSON_TYPE type, const unsigned char* data, size_t data_size, void* user_data);
00106
00107 /* Used internally by load_config.c:handle_user_value_clause() */
00108 int json_dom_fini_aux(JSON_DOM_PARSER* dom_parser, JSON_VALUE* p_root);
00109
00110 int json_dom_clean(JSON_DOM_PARSER* dom_parser);
00111
00112 /* Initialize the DOM parser structure.
00113  *
00114  * The parameter `config` is propagated into json_init().
00115  */
00116 WOLFSENTRY_API int json_dom_init(
00117 #ifdef WOLFSENTRY
00118     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00119 #endif
00120     JSON_DOM_PARSER* dom_parser, const JSON_CONFIG* config, unsigned dom_flags);
00121
00122 /* Feed the parser with more input.
00123  */
00124 WOLFSENTRY_API int json_dom_feed(JSON_DOM_PARSER* dom_parser, const unsigned char* input, size_t
      size);
00125
00126 /* Finish the parsing and free any resources associated with the parser.
00127  *
00128  * On success, zero is returned and the JSON_VALUE pointed by `p_dom` is initialized
00129  * accordingly to the root of the data in the JSON input (typically array or
00130  * object), and it contains all the data from the JSON input.
00131  *
00132  * On failure, the error code is returned; info about position of the issue in
00133  * the input is filled in the structure pointed by `p_pos` (if `p_pos` is not
00134  * NULL and if it is a parsing kind of error); and the value pointed by `p_dom`
00135  * is initialized to JSON_VALUE_NULL.
00136  */
00137 WOLFSENTRY_API int json_dom_fini(JSON_DOM_PARSER* dom_parser, JSON_VALUE* p_dom, JSON_INPUT_POS*
      p_pos);
00138
00139
00140 /* Simple wrapper for json_dom_init() + json_dom_feed() + json_dom_fini(),
00141  * usable when the provided input contains complete JSON document.
00142  */
00143 WOLFSENTRY_API int json_dom_parse(
```

```
00144 #ifdef WOLFSENTRY
00145     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00146 #endif
00147                     const unsigned char* input, size_t size, const JSON_CONFIG* config,
00148                     unsigned dom_flags, JSON_VALUE* p_root, JSON_INPUT_POS* p_pos);
00149
00150
00151 /* Dump recursively all the DOM hierarchy out, via the provided writing
00152  * callback.
00153  *
00154  * The provided writing function must write all the data provided to it
00155  * and return zero to indicate success, or non-zero to indicate an error
00156  * and abort the operation.
00157  *
00158  * Returns zero on success, JSON_ERR_OUTOFMEMORY, or an error the code returned
00159  * from writing callback.
00160  */
00161 #define JSON_DOM_DUMP_MINIMIZE          0x0001  /* Do not indent, do not use no extra whitespace
      including new lines. */
00162 #define JSON_DOM_DUMP_FORCECLRF         0x0002  /* Use "\r\n" instead of just "\n". */
00163 #define JSON_DOM_DUMP_INDENTWITHSPACES  0x0004  /* Indent with `tab_width' spaces instead of with
      '\t'. */
00164 #define JSON_DOM_DUMP_PREFERDICTORDER   0x0008  /* Prefer original dictionary order, if available. */
00165
00166 WOLFSENTRY_API int json_dom_dump(
00167 #ifdef WOLFSENTRY
00168     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00169 #endif
00170                     const JSON_VALUE* root,
00171                     JSON_DUMP_CALLBACK write_func, void* user_data,
00172                     unsigned tab_width, unsigned flags);
00173
00174 WOLFSENTRY_API const char* json_dom_error_str(int err_code);
00175
00176 #ifdef __cplusplus
00177 }  /* extern "C" { */
00178 #endif
00179
00180 #endif  /* JSON_DOM_H */
```

10.2 centijson_sax.h

```
00001 /*
00002  * centijson_sax.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00023 /*
00024  * CentiJSON
00025  * <http://github.com/mity/centijson>
00026  *
00027  * Copyright (c) 2018 Martin Mitas
00028  *
00029  * Permission is hereby granted, free of charge, to any person obtaining a
00030  * copy of this software and associated documentation files (the "Software"),
00031  * to deal in the Software without restriction, including without limitation
00032  * the rights to use, copy, modify, merge, publish, distribute, sublicense,
00033  * and/or sell copies of the Software, and to permit persons to whom the
00034  * Software is furnished to do so, subject to the following conditions:
00035  *
00036  * The above copyright notice and this permission notice shall be included in
00037  * all copies or substantial portions of the Software.
00038  *
00039  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
00040  * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00041  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
```

```
00042  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00043  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
00044  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
00045  * IN THE SOFTWARE.
00046  */
00047
00048 #ifndef CENTIJSON_SAX_H
00049 #define CENTIJSON_SAX_H
00050
00051 #if !defined(WOLFSENTRY) && !defined(WOLFSENTRY_API)
00052     #define WOLFSENTRY_API
00053 #endif
00054
00055 #ifndef WOLFSENTRY
00056 #include <stdint.h>
00057 #include <sys/types.h>
00058 #endif
00059
00060 #ifdef __cplusplus
00061 extern "C" {
00062 #endif
00063
00064
00065 /* JSON data types.
00066  *
00067  * Note that we distinguish beginning/end of the arrays and objects for
00068  * the purposes of the processing.
00069  */
00070 typedef enum JSON_TYPE {
00071     JSON_NULL,
00072     JSON_FALSE,
00073     JSON_TRUE,
00074     JSON_NUMBER,
00075     JSON_STRING,
00076     JSON_KEY,     /* String in the specific role of an object key. */
00077     JSON_ARRAY_BEG,
00078     JSON_ARRAY_END,
00079     JSON_OBJECT_BEG,
00080     JSON_OBJECT_END
00081 } JSON_TYPE;
00082
00083
00084 /* Error codes.
00085  */
00086 #define JSON_ERR_SUCCESS                0
00087 #define JSON_ERR_INTERNAL               (-1)    /* This should never happen. If you see it, report bug
      ;-) */
00088 #define JSON_ERR_OUTOFMEMORY            (-2)
00089 #define JSON_ERR_SYNTAX                 (-4)    /* Generic syntax error. (More specific error codes
      are preferred.) */
00090 #define JSON_ERR_BADCLOSER              (-5)    /* Mismatch in brackets (e.g. "{ ]" or "[ }") */
00091 #define JSON_ERR_BADROOTTYPE            (-6)    /* Root type not allowed by CONFIG::flags. */
00092 #define JSON_ERR_EXPECTEDVALUE          (-7)    /* Something unexpected where value has to be. */
00093 #define JSON_ERR_EXPECTEDKEY            (-8)    /* Something unexpected where key has to be. */
00094 #define JSON_ERR_EXPECTEDVALUEORCLOSER  (-9)    /* Something unexpected where value or array/object
      closer has to be. */
00095 #define JSON_ERR_EXPECTEDKEYORCLOSER    (-10)   /* Something unexpected where key or array/object
      closer has to be. */
00096 #define JSON_ERR_EXPECTEDCOLON          (-11)   /* Something unexpected where colon has to be. */
00097 #define JSON_ERR_EXPECTEDCOMMAORCLOSER  (-12)   /* Something unexpected where comma or array/object
      has to be. */
00098 #define JSON_ERR_EXPECTEDEOF            (-13)   /* Something unexpected where end-of-file has to be.
      */
00099 #define JSON_ERR_MAXTOTALLEN            (-14)   /* Reached JSON_CONFIG::max_total_len */
00100 #define JSON_ERR_MAXTOTALVALUES         (-15)   /* Reached JSON_CONFIG::max_total_values */
00101 #define JSON_ERR_MAXNESTINGLEVEL        (-16)   /* Reached JSON_CONFIG::max_nesting_level */
00102 #define JSON_ERR_MAXNUMBERLEN           (-17)   /* Reached JSON_CONFIG::max_number_len */
00103 #define JSON_ERR_MAXSTRINGLEN           (-18)   /* Reached JSON_CONFIG::max_string_len */
00104 #define JSON_ERR_MAXKEYLEN              (-19)   /* Reached JSON_CONFIG::max_key_len */
00105 #define JSON_ERR_UNCLOSEDSTRING         (-20)   /* Unclosed string */
00106 #define JSON_ERR_UNESCAPEDCONTROL       (-21)   /* Unescaped control character (in a string) */
00107 #define JSON_ERR_INVALIDESCAPE          (-22)   /* Invalid/unknown escape sequence (in a string) */
00108 #define JSON_ERR_INVALIDUTF8            (-23)   /* Invalid UTF-8 (in a string) */
00109 #define JSON_ERR_NOT_INITED             (-24)   /* Attempt to access an uninited JSON_PARSER or
      JSON_DOM_PARSER. */
00110
00111
00112 /* Bits for JSON_CONFIG::flags.
00113  */
00114 #define JSON_NONULLASROOT           0x0001U  /* Disallow null to be root value */
00115 #define JSON_NOBOOLASROOT           0x0002U  /* Disallow false or true to be root value */
00116 #define JSON_NONUMBERASROOT         0x0004U  /* Disallow number to be root value */
00117 #define JSON_NOSTRINGASROOT         0x0008U  /* Disallow string to be root value */
00118 #define JSON_NOARRAYASROOT          0x0010U  /* Disallow array to be root value */
00119 #define JSON_NOOBJECTASROOT         0x0020U  /* Disallow object  to be root value */
00120
00121 #define JSON_NOSCALARROOT           (JSON_NONULLASROOT | JSON_NOBOOLASROOT |  \
```

```
00122                                    JSON_NONUMBERASROOT | JSON_NOSTRINGASROOT)
00123 #define JSON_NOVECTORROOT           (JSON_NOARRAYASROOT | JSON_NOOBJECTASROOT)
00124
00125 #define JSON_IGNOREILLUTF8KEY       0x0100U  /* Ignore ill-formed UTF-8 (for keys). */
00126 #define JSON_FIXILLUTF8KEY          0x0200U  /* Replace ill-formed UTF-8 char with replacement char
      (for keys). */
00127 #define JSON_IGNOREILLUTF8VALUE     0x0400U  /* Ignore ill-formed UTF-8 (for string values). */
00128 #define JSON_FIXILLUTF8VALUE        0x0800U  /* Replace ill-formed UTF-8 char with replacement char
      (for string values). */
00129
00130
00131
00132 /* Parser options, passed into json_init().
00133  *
00134  * If NULL is passed to json_init(), default values are used.
00135  */
00136 typedef struct JSON_CONFIG {
00137     size_t max_total_len;       /* zero means no limit; default: 10 MB */
00138     size_t max_total_values;    /* zero means no limit; default: 0 */
00139     size_t max_number_len;      /* zero means no limit; default: 512 */
00140     size_t max_string_len;      /* zero means no limit; default: 65536 */
00141     size_t max_key_len;         /* zero means no limit; default: 512 */
00142     unsigned max_nesting_level; /* zero means no limit; default: 512 */
00143     unsigned flags;             /* default: 0 */
00144 } JSON_CONFIG;
00145
00146
00147 /* Helper structure describing position in the input.
00148  *
00149  * It is used to specify where in the input a parsing error occurred for
00150  * better diagnostics.
00151  */
00152 typedef struct JSON_INPUT_POS {
00153     size_t offset;
00154     unsigned line_number;
00155     unsigned column_number;
00156 } JSON_INPUT_POS;
00157
00158
00159 /* Callbacks the application has to implement, to process the parsed data.
00160  */
00161 typedef struct JSON_CALLBACKS {
00162     /* Data processing callback. For now (and maybe forever) the only callback.
00163      *
00164      * Note that `data` and `data_size` are set only for JSON_KEY, JSON_STRING
00165      * and JSON_NUMBER. (For the other types the callback always gets NULL and
00166      * 0).
00167      *
00168      * Inside an object, the application is guaranteed to get keys and their
00169      * corresponding values in the alternating fashion (i.e. in the order
00170      * as they are in the JSON input.).
00171      *
00172      * Application can abort the parsing operation by returning a non-zero.
00173      * Note the non-zero return value of the callback is propagated to
00174      * json_feed() and json_fini().
00175      */
00176     int (*process)(JSON_TYPE /*type*/, const unsigned char* /*data*/,
00177                    size_t /*data_size*/, void* /*user_data*/);
00178 } JSON_CALLBACKS;
00179
00180
00181 /* Internal parser state. Use pointer to this structure as an opaque handle.
00182  */
00183 typedef struct JSON_PARSER {
00184 #ifdef WOLFSENTRY
00185     struct wolfsentry_allocator *allocator;
00186 #ifdef WOLFSENTRY_THREADSAFE
00187     struct wolfsentry_thread_context *thread;
00188 #endif
00189 #endif
00190     JSON_CALLBACKS callbacks;
00191     JSON_CONFIG config;
00192     void* user_data;
00193
00194     JSON_INPUT_POS pos;
00195     JSON_INPUT_POS value_pos;
00196     JSON_INPUT_POS err_pos;
00197
00198     int errcode;
00199
00200     size_t value_counter;
00201
00202     unsigned char* nesting_stack;
00203     size_t nesting_level;
00204     size_t nesting_stack_size;
00205
00206     enum centijson_automaton {
```

```
00207         AUTOMATON_MAIN = 0,
00208         AUTOMATON_NULL = 1,
00209         AUTOMATON_FALSE = 2,
00210         AUTOMATON_TRUE = 3,
00211         AUTOMATON_NUMBER = 4,
00212         AUTOMATON_STRING = 6,
00213         AUTOMATON_KEY = 7
00214     } automaton;
00215
00216     unsigned state;
00217     unsigned substate;
00218
00219     uint32_t codepoint[2];
00220
00221     unsigned char* buf;
00222     size_t buf_used;
00223     size_t buf_alloced;
00224
00225     size_t last_cl_offset;  /* Offset of most recently seen '\r' */
00226 } JSON_PARSER;
00227
00228
00229
00230 /* Fill `config' with options used by default.
00231  */
00232 WOLFSENTRY_API_VOID json_default_config(JSON_CONFIG* config);
00233
00234
00235 /* Initialize the parser, associate it with the given callbacks and
00236  * configuration. Returns zero on success, non-zero on an error.
00237  *
00238  * If `config' is NULL, default values are used.
00239  */
00240 WOLFSENTRY_API int json_init(
00241 #ifdef WOLFSENTRY
00242     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00243 #endif
00244               JSON_PARSER* parser,
00245               const JSON_CALLBACKS* callbacks,
00246               const JSON_CONFIG* config,
00247               void* user_data);
00248
00249 /* Feed the parser with more input.
00250  *
00251  * Returns zero on success.
00252  *
00253  * If an error occurs it returns non-zero and any attempt to call json_feed()
00254  * again shall just fail with the same error code. Note the application should
00255  * still  call json_fini() to release all resources allocated by the parser.
00256  */
00257 WOLFSENTRY_API int json_feed(JSON_PARSER* parser, const unsigned char* input, size_t size);
00258
00259 /* Finish parsing of the document (note it can still call some callbacks); and
00260  * release any resources held by the parser.
00261  *
00262  * Returns zero on success, or non-zero on failure.
00263  *
00264  * If `p_pos' is not NULL, it is filled with info about reached position in the
00265  * input. It can help in diagnostics if the parsing failed.
00266  *
00267  * Note that if the preceding call to json_feed() failed, the error status also
00268  * propagates into json_fini().
00269  *
00270  * Also note this function may still fail even when all preceding calls to
00271  * json_feed() succeeded. This typically happens when the parser was fed with
00272  * an incomplete JSON document.
00273  */
00274 WOLFSENTRY_API int json_fini(JSON_PARSER* parser, JSON_INPUT_POS* p_pos);
00275
00276
00277 /* Simple wrapper function for json_init() + json_feed() + json_fini(), usable
00278  * when the provided input contains complete JSON document.
00279  */
00280 WOLFSENTRY_API int json_parse(
00281 #ifdef WOLFSENTRY
00282     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00283 #endif
00284                const unsigned char* input, size_t size,
00285                const JSON_CALLBACKS* callbacks, const JSON_CONFIG* config,
00286                void* user_data, JSON_INPUT_POS* p_pos);
00287
00288
00289 /* Converts error code to human readable error message
00290  */
00291 WOLFSENTRY_API const char* json_error_str(int err_code);
00292
00293 WOLFSENTRY_API const char* json_type_str(JSON_TYPE type);
```

```
00294
00295
00296 /*****************
00297  *** Utilities ***
00298  *****************/
00299
00300 /* When implementing the callback processing the parsed data, these utilities
00301  * below may come handy.
00302  */
00303
00304 /* Analyze the string holding a JSON number, and analyze whether it can
00305  * fit into integer types.
00306  *
00307  * (Note it says "no" in cases the number string contains any fraction or
00308  * exponent part.)
00309  */
00310 WOLFSENTRY_API int json_analyze_number(const unsigned char* num, size_t num_size,
00311                          int* p_is_int32_compatible,
00312                          int* p_is_uint32_compatible,
00313                          int* p_is_int64_compatible,
00314                          int* p_is_uint64_compatible);
00315
00316 /* Convert the string holding JSON number to the given C type.
00317  *
00318  * Note the conversion to any of the integer types is undefined unless
00319  * json_analyze_number() says it's fine.
00320  *
00321  * Also note that json_number_to_double() can fail with JSON_ERR_OUTOFMEMORY.
00322  * Hence its prototype differs.
00323  */
00324 WOLFSENTRY_API int32_t json_number_to_int32(const unsigned char* num, size_t num_size);
00325 WOLFSENTRY_API uint32_t json_number_to_uint32(const unsigned char* num, size_t num_size);
00326 WOLFSENTRY_API int64_t json_number_to_int64(const unsigned char* num, size_t num_size);
00327 WOLFSENTRY_API uint64_t json_number_to_uint64(const unsigned char* num, size_t num_size);
00328 WOLFSENTRY_API int json_number_to_double(const unsigned char* num, size_t num_size, double* p_result);
00329
00330
00331 typedef int (*JSON_DUMP_CALLBACK)(const unsigned char* /*str*/, size_t /*size*/, void* /*user_data*/);
00332
00333 /* Helpers for writing numbers and strings in JSON-compatible format.
00334  *
00335  * Note that json_dump_string() assumes the string is a well-formed UTF-8
00336  * string which needs no additional Unicode validation. The function "only"
00337  * handles proper escaping of control characters.
00338  *
00339  * The provided writer callback must write all the data provided to it and
00340  * return zero to indicate success, or non-zero to indicate an error and abort
00341  * the operation.
00342  *
00343  * All these return zero on success, JSON_ERR_OUTOFMEMORY, or an error code
00344  * propagated from the writer callback.
00345  *
00346  * (Given that all the other JSON stuff is trivial to output, the application
00347  * is supposed to implement that manually.)
00348  */
00349 WOLFSENTRY_API int json_dump_int32(int32_t i32, JSON_DUMP_CALLBACK write_func, void* user_data);
00350 WOLFSENTRY_API int json_dump_uint32(uint32_t u32, JSON_DUMP_CALLBACK write_func, void* user_data);
00351 WOLFSENTRY_API int json_dump_int64(int64_t i64, JSON_DUMP_CALLBACK write_func, void* user_data);
00352 WOLFSENTRY_API int json_dump_uint64(uint64_t u64, JSON_DUMP_CALLBACK write_func, void* user_data);
00353 WOLFSENTRY_API int json_dump_double(double dbl, JSON_DUMP_CALLBACK write_func, void* user_data);
00354 WOLFSENTRY_API int json_dump_string(const unsigned char* str, size_t size, JSON_DUMP_CALLBACK
      write_func, void* user_data);
00355
00356
00357 #ifdef __cplusplus
00358 }  /* extern "C" { */
00359 #endif
00360
00361 #endif  /* CENTIJSON_SAX_H */
```

10.3 centijson_value.h

```
00001 /*
00002  * centijson_value.h
00003  *
00004  * Copyright (C) 2022-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
```

```
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00023 /*
00024  * C Reusables
00025  * <http://github.com/mity/c-reusables>
00026  *
00027  * Copyright (c) 2018 Martin Mitas
00028  *
00029  * Permission is hereby granted, free of charge, to any person obtaining a
00030  * copy of this software and associated documentation files (the "Software"),
00031  * to deal in the Software without restriction, including without limitation
00032  * the rights to use, copy, modify, merge, publish, distribute, sublicense,
00033  * and/or sell copies of the Software, and to permit persons to whom the
00034  * Software is furnished to do so, subject to the following conditions:
00035  *
00036  * The above copyright notice and this permission notice shall be included in
00037  * all copies or substantial portions of the Software.
00038  *
00039  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
00040  * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00041  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00042  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00043  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
00044  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
00045  * IN THE SOFTWARE.
00046  */
00047
00048 #ifndef CENTIJSON_VALUE_H
00049 #define CENTIJSON_VALUE_H
00050
00051 #ifdef __cplusplus
00052 extern "C" {
00053 #endif
00054
00055 #ifdef WOLFSENTRY
00056 #include "wolfsentry.h"
00057 #endif
00058 #ifndef WOLFSENTRY_API
00059 #define WOLFSENTRY_API
00060 #endif
00061
00062 #ifndef WOLFSENTRY
00063 #include <stdint.h>
00064 #endif
00065
00066 /* The value structure.
00067  * Use as opaque.
00068  */
00069 typedef struct JSON_VALUE {
00070     /* We need at least 2 * sizeof(void*). Sixteen bytes covers that on 64-bit
00071      * platforms and it seems as a good compromise allowing to "inline" all
00072      * numeric types as well as short strings; which is good idea: most dict
00073      * keys as well as many string values are in practice quite short. */
00074     union {
00075         uint8_t data_bytes[16];
00076         void *data_ptrs[16 / sizeof(void *)];
00077     } data;
00078 } JSON_VALUE;
00079
00080
00081 /* Value types.
00082  */
00083 typedef enum JSON_VALUE_TYPE {
00084     JSON_VALUE_NULL = 0,
00085     JSON_VALUE_BOOL,
00086     JSON_VALUE_INT32,
00087     JSON_VALUE_UINT32,
00088     JSON_VALUE_INT64,
00089     JSON_VALUE_UINT64,
00090     JSON_VALUE_FLOAT,
00091     JSON_VALUE_DOUBLE,
00092     JSON_VALUE_STRING,
00093     JSON_VALUE_ARRAY,
00094     JSON_VALUE_DICT
00095 } JSON_VALUE_TYPE;
00096
00097
00098 /* Free any resources the value holds.
```

```
00099  * For ARRAY and DICT it is recursive.
00100  */
00101 WOLFSENTRY_API int json_value_fini(
00102 #ifdef WOLFSENTRY
00103     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00104 #endif
00105     JSON_VALUE* v);
00106
00107 /* Get value type.
00108  */
00109 WOLFSENTRY_API JSON_VALUE_TYPE json_value_type(const JSON_VALUE* v);
00110
00111 /* Check whether the value is "compatible" with the given type.
00112  *
00113  * This is especially useful for determining whether a numeric value can be
00114  * "casted" to other numeric type. The function does some basic checking
00115  * whether such conversion looses substantial information.
00116  *
00117  * For example, value initialized with init_float(&v, 1.0f) is considered
00118  * compatible with INT32, because 1.0f has zero fraction and 1 fits between
00119  * INT32_MIN and INT32_MAX. Therefore calling int32_value(&v) gets sensible
00120  * result.
00121  */
00122 WOLFSENTRY_API int json_value_is_compatible(const JSON_VALUE* v, JSON_VALUE_TYPE type);
00123
00124 /* Values newly added into array or dictionary are of type VALUE_NULL.
00125  *
00126  * Additionally, for such newly created values, an internal flag is used to
00127  * mark that the value was never explicitly initialized by the application.
00128  *
00129  * This function checks value of the flag, and allows thus the caller to
00130  * distinguish whether the value was just added; or whether the value was
00131  * explicitly initialized as VALUE_NULL with value_init_null().
00132  *
00133  * Caller is supposed to initialize all such newly added value with any of the
00134  * value_init_XXX() functions, and hence reset the flag.
00135  */
00136 WOLFSENTRY_API int json_value_is_new(const JSON_VALUE* v);
00137
00138 /* Simple recursive getter, capable to get a value dwelling deep in the
00139  * hierarchy formed by nested arrays and dictionaries.
00140  *
00141  * Limitations: The function is not capable to deal with object keys which
00142  * contain zero byte '\0', slash '/' or brackets '[' ']' because those are
00143  * interpreted by the function as special characters:
00144  *
00145  *  -- '/' delimits dictionary keys (and optionally also array indexes;
00146  *     paths "foo/[4]" and "foo[4]" are treated as equivalent.)
00147  *  -- '[' ']' enclose array indexes (for distinguishing from numbered
00148  *     dictionary keys). Note that negative indexes are supported here;
00149  *     '[-1]' refers to the last element in the array, '[-2]' to the element
00150  *     before the last element etc.
00151  *  -- '\0' terminates the whole path (as is normal with C strings).
00152  *
00153  * Examples:
00154  *
00155  *  (1) value_path(root, "") gets directly the root.
00156  *
00157  *  (2) value_path(root, "foo") gets value keyed with 'foo' if root is a
00158  *      dictionary having such value, or NULL otherwise.
00159  *
00160  *  (3) value_path(root, "[4]") gets value with index 4 if root is an array
00161  *      having so many members, or NULL otherwise.
00162  *
00163  *  (4) value_path(root, "foo[2]/bar/baz[3]") walks deeper and deeper and
00164  *      returns a value stored there assuming these all conditions are true:
00165  *       -- root is dictionary having the key "foo";
00166  *       -- that value is a nested list having the index [2];
00167  *       -- that value is a nested dictionary having the key "bar";
00168  *       -- that value is a nested dictionary having the key "baz";
00169  *       -- and finally, that is a list having the index [3].
00170  *      If any of those is not fulfilled, then NULL is returned.
00171  */
00172 WOLFSENTRY_API JSON_VALUE* json_value_path(JSON_VALUE* root, const char* path);
00173
00174 /* value_build_path() is similar to value_path(); but allows easy populating
00175  * of value hierarchies.
00176  *
00177  * If all values along the path already exist, the behavior is exactly the same
00178  * as value_path().
00179  *
00180  * But when a value corresponding to any component of the path does not exist
00181  * then, instead of returning NULL, new value is added into the parent
00182  * container (assuming the parent existing container has correct type as
00183  * assumed by the path.)
00184  *
00185  * Caller may use empty "[]" to always enforce appending a new value into an
```

```
00186  * array. E.g. value_build_path(root, "multiple_values/[]/name") makes sure the
00187  * root contains an array under the key "multiple_values", and a new dictionary
00188  * is appended at the end of the array. This new dictionary gets a new value
00189  * under the key "name". Assuming the function succeeds, the caller can now be
00190  * sure the "name" is initialized as VALUE_NULL because the new dictionary has
00191  * been just created and added as the last element if the list.
00192  *
00193  * If such new value does not correspond to the last path component, the new
00194  * value gets initialized as the right type so subsequent path component can
00195  * be treated the same way.
00196  *
00197  * If the function creates the value corresponding to the last component of the
00198  * path, it is initialized as VALUE_NULL and the "new flag" is set for it, so
00199  * caller can test this condition with value_is_new().
00200  *
00201  * Returns NULL if the path cannot be resolved because any existing value
00202  * has a type incompatible with the path; if creation of any value along the
00203  * path fails; or if an array index is out of bounds.
00204  */
00205 /* missing implementation */
00206 /* WOLFSENTRY_API JSON_VALUE* json_value_build_path(JSON_VALUE* root, const char* path); */
00207
00208
00209 /******************
00210  *** VALUE_NULL ***
00211  ******************/
00212
00213 /* Note it is guaranteed that VALUE_NULL does not need any explicit clean-up;
00214  * i.e. application may avoid calling value_fini().
00215  *
00216  * But it is allowed to. value_fini() for VALUE_NULL is a noop.
00217  */
00218
00219
00220 /* Static initializer.
00221  */
00222 #define JSON_VALUE_NULL_INITIALIZER    { { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } }
00223
00224 WOLFSENTRY_API_VOID json_value_init_null(JSON_VALUE* v);
00225
00226
00227 /******************
00228  *** VALUE_BOOL ***
00229  ******************/
00230
00231 WOLFSENTRY_API int json_value_init_bool(JSON_VALUE* v, int b);
00232
00233 WOLFSENTRY_API int json_value_bool(const JSON_VALUE* v);
00234
00235
00236 /*********************
00237  *** Numeric types ***
00238  *********************/
00239
00240
00241 /* Initializers.
00242  */
00243 WOLFSENTRY_API int json_value_init_int32(
00244 #ifdef WOLFSENTRY
00245     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00246 #endif
00247     JSON_VALUE* v, int32_t i32);
00248 WOLFSENTRY_API int json_value_init_uint32(
00249 #ifdef WOLFSENTRY
00250     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00251 #endif
00252     JSON_VALUE* v, uint32_t u32);
00253 WOLFSENTRY_API int json_value_init_int64(
00254 #ifdef WOLFSENTRY
00255     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00256 #endif
00257     JSON_VALUE* v, int64_t i64);
00258 WOLFSENTRY_API int json_value_init_uint64(
00259 #ifdef WOLFSENTRY
00260     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00261 #endif
00262     JSON_VALUE* v, uint64_t u64);
00263 WOLFSENTRY_API int json_value_init_float(
00264 #ifdef WOLFSENTRY
00265     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00266 #endif
00267     JSON_VALUE* v, float f);
00268 WOLFSENTRY_API int json_value_init_double(
00269 #ifdef WOLFSENTRY
00270     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00271 #endif
00272     JSON_VALUE* v, double d);
```

```
00273
00274 /* Getters.
00275  *
00276  * Note you may use any of the getter function for any numeric value. These
00277  * functions perform required conversions under the hood. The conversion may
00278  * have have the same side/limitations as C casting.
00279  *
00280  * However application may use json_value_is_compatible() to verify whether the
00281  * conversion should provide a reasonable result.
00282  */
00283 WOLFSENTRY_API int32_t json_value_int32(const JSON_VALUE* v);
00284 WOLFSENTRY_API uint32_t json_value_uint32(const JSON_VALUE* v);
00285 WOLFSENTRY_API int64_t json_value_int64(const JSON_VALUE* v);
00286 WOLFSENTRY_API uint64_t json_value_uint64(const JSON_VALUE* v);
00287 WOLFSENTRY_API float json_value_float(const JSON_VALUE* v);
00288 WOLFSENTRY_API double json_value_double(const JSON_VALUE* v);
00289
00290
00291 /********************
00292  *** JSON_VALUE_STRING ***
00293  ********************/
00294
00295 /* Note JSON_VALUE_STRING allows to store any sequences of any bytes, even a binary
00296  * data. No particular encoding of the string is assumed. Even zero bytes are
00297  * allowed (but then the caller has to use json_value_init_string_() and specify
00298  * the string length explicitly).
00299  */
00300
00301 /* The function json_value_init_string_() initializes the JSON_VALUE_STRING with any
00302  * sequence of bytes, of any length. It also adds automatically one zero byte
00303  * (not counted in the length of the string).
00304  *
00305  * The function json_value_init_string() is equivalent to calling directly
00306  * json_value_init_string_(str, strlen(str)).
00307  *
00308  * The parameter str is allowed to be NULL (then the functions behave the same
00309  * way as if it is points to an empty string).
00310  */
00311 WOLFSENTRY_API int json_value_init_string_(
00312 #ifdef WOLFSENTRY
00313     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00314 #endif
00315     JSON_VALUE* v, const unsigned char* str, size_t len);
00316 WOLFSENTRY_API int json_value_init_string(
00317 #ifdef WOLFSENTRY
00318     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00319 #endif
00320     JSON_VALUE* v, const unsigned char* str);
00321
00322 /* Get pointer to the internal buffer holding the string. The caller may assume
00323  * the returned string is always zero-terminated.
00324  */
00325 WOLFSENTRY_API const unsigned char* json_value_string(const JSON_VALUE* v);
00326
00327 /* Get length of the string. (The implicit zero terminator does not count.)
00328  */
00329 WOLFSENTRY_API size_t json_value_string_length(const JSON_VALUE* v);
00330
00331
00332 /********************
00333  *** JSON_VALUE_ARRAY ***
00334  ********************/
00335
00336 /* Array of values.
00337  *
00338  * Note that any new value added into the array with json_value_array_append() or
00339  * json_value_array_insert() is initially of the type JSON_VALUE_NULL and that it has
00340  * an internal flag marking the value as new (so that json_value_is_new() returns
00341  * non-zero for it). Application is supposed to initialize the newly added
00342  * value by any of the value initialization functions.
00343  *
00344  * WARNING: Modifying contents of an array (i.e. inserting, appending and also
00345  * removing a value)  can lead to reallocation of internal array buffer.
00346  * Hence, consider all JSON_VALUE* pointers invalid after modifying the array.
00347  * That includes the return values of json_value_array_get(), json_value_array_get_all(),
00348  * but also preceding calls of json_value_array_append() and json_value_array_insert().
00349  */
00350 WOLFSENTRY_API int json_value_init_array(
00351 #ifdef WOLFSENTRY
00352     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00353 #endif
00354     JSON_VALUE* v);
00355
00356 /* Get count of items in the array.
00357  */
00358 WOLFSENTRY_API size_t json_value_array_size(const JSON_VALUE* v);
00359
```

```
00360 /* Get the specified item.
00361  */
00362 WOLFSENTRY_API JSON_VALUE* json_value_array_get(const JSON_VALUE* v, size_t index);
00363
00364 /* Get pointer to internal C array of all items.
00365  */
00366 WOLFSENTRY_API JSON_VALUE* json_value_array_get_all(const JSON_VALUE* v);
00367
00368 /* Append/insert new item.
00369  */
00370 WOLFSENTRY_API JSON_VALUE* json_value_array_append(
00371 #ifdef WOLFSENTRY
00372     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00373 #endif
00374     JSON_VALUE* v);
00375 WOLFSENTRY_API JSON_VALUE* json_value_array_insert(
00376 #ifdef WOLFSENTRY
00377     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00378 #endif
00379     JSON_VALUE* v, size_t index);
00380
00381 /* Remove an item (or range of items).
00382  */
00383 WOLFSENTRY_API int json_value_array_remove(
00384 #ifdef WOLFSENTRY
00385     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00386 #endif
00387     JSON_VALUE* v, size_t index);
00388 WOLFSENTRY_API int json_value_array_remove_range(
00389 #ifdef WOLFSENTRY
00390     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00391 #endif
00392     JSON_VALUE* v, size_t index, size_t count);
00393
00394 /* Remove and destroy all members (recursively).
00395  */
00396 WOLFSENTRY_API int json_value_array_clean(
00397 #ifdef WOLFSENTRY
00398     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00399 #endif
00400     JSON_VALUE* v);
00401
00402
00403 /******************
00404  *** JSON_VALUE_DICT ***
00405  ******************/
00406
00407 /* Dictionary of values. (Internally implemented as red-black tree.)
00408  *
00409  * Note that any new value added into the dictionary is initially of the type
00410  * JSON_VALUE_NULL and that it has  an internal flag marking the value as new
00411  * (so that json_value_is_new() returns non-zero for it). Application is supposed
00412  * to initialize the newly added value by any of the value initialization
00413  * functions.
00414  *
00415  * Note that all the functions adding/removing any items may invalidate all
00416  * pointers into the dictionary.
00417  */
00418
00419
00420 /* Flag for init_dict_ex() asking to maintain the order in which the dictionary
00421  * is populated and enabling dict_walk_ordered().
00422  *
00423  * If used, the dictionary consumes more memory.
00424  */
00425 #define JSON_VALUE_DICT_MAINTAINORDER      0x0001
00426
00427 /* Initialize the value as a (empty) dictionary.
00428  *
00429  * json_value_init_dict_ex() allows to specify custom comparer function (may be NULL)
00430  * or flags changing the default behavior of the dictionary.
00431  */
00432 WOLFSENTRY_API int json_value_init_dict(
00433 #ifdef WOLFSENTRY
00434     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00435 #endif
00436     JSON_VALUE* v);
00437 WOLFSENTRY_API int json_value_init_dict_ex(
00438 #ifdef WOLFSENTRY
00439                         WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00440 #endif
00441                         JSON_VALUE* v,
00442                         int (*custom_cmp_func)(const unsigned char* /*key1*/, size_t /*len1*/,
00443                                                const unsigned char* /*key2*/, size_t /*len2*/),
00444                         unsigned flags);
00445
00446 /* Get flags of the dictionary.
```

```
00447  */
00448 WOLFSENTRY_API unsigned json_value_dict_flags(const JSON_VALUE* v);
00449
00450 /* Get count of items in the dictionary.
00451  */
00452 WOLFSENTRY_API size_t json_value_dict_size(const JSON_VALUE* v);
00453
00454 /* Get all keys.
00455  *
00456  * If the buffer provided by the caller is too small, only subset of keys shall
00457  * be retrieved.
00458  *
00459  * Returns count of retrieved keys.
00460  */
00461 WOLFSENTRY_API size_t json_value_dict_keys_sorted(const JSON_VALUE* v, const JSON_VALUE** buffer,
      size_t buffer_size);
00462 WOLFSENTRY_API size_t json_value_dict_keys_ordered(const JSON_VALUE* v, const JSON_VALUE** buffer,
      size_t buffer_size);
00463
00464 /* Find an item with the given key, or return NULL of no such item exists.
00465  */
00466 WOLFSENTRY_API JSON_VALUE* json_value_dict_get_(const JSON_VALUE* v, const unsigned char* key, size_t
      key_len);
00467 WOLFSENTRY_API JSON_VALUE* json_value_dict_get(const JSON_VALUE* v, const unsigned char* key);
00468
00469 /* Add new item with the given key of type JSON_VALUE_NULL.
00470  *
00471  * Returns NULL if the key is already used.
00472  */
00473 WOLFSENTRY_API JSON_VALUE* json_value_dict_add_(
00474 #ifdef WOLFSENTRY
00475     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00476 #endif
00477     JSON_VALUE* v, const unsigned char* key, size_t key_len);
00478 WOLFSENTRY_API JSON_VALUE* json_value_dict_add(
00479 #ifdef WOLFSENTRY
00480     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00481 #endif
00482     JSON_VALUE* v, const unsigned char* key);
00483
00484 /* This is combined operation of json_value_dict_get() and json_value_dict_add().
00485  *
00486  * Get value of the given key. If no such value exists, new one is added.
00487  * Application can check for such situation with json_value_is_new().
00488  *
00489  * NULL is returned only in an out-of-memory situation.
00490  */
00491 WOLFSENTRY_API JSON_VALUE* json_value_dict_get_or_add_(
00492 #ifdef WOLFSENTRY
00493     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00494 #endif
00495     JSON_VALUE* v, const unsigned char* key, size_t key_len);
00496 WOLFSENTRY_API JSON_VALUE* json_value_dict_get_or_add(
00497 #ifdef WOLFSENTRY
00498     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00499 #endif
00500     JSON_VALUE* v, const unsigned char* key);
00501
00502 /* Remove and destroy (recursively) the given item from the dictionary.
00503  */
00504 WOLFSENTRY_API int json_value_dict_remove_(
00505 #ifdef WOLFSENTRY
00506     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00507 #endif
00508     JSON_VALUE* v, const unsigned char* key, size_t key_len);
00509 WOLFSENTRY_API int json_value_dict_remove(
00510 #ifdef WOLFSENTRY
00511     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00512 #endif
00513     JSON_VALUE* v, const unsigned char* key);
00514
00515 /* Walking over all items in the dictionary. The callback function is called
00516  * for every item in the dictionary, providing key and value and propagating
00517  * the user data into it. If the callback returns non-zero, the function
00518  * aborts immediately.
00519  *
00520  * Note dict_walk_ordered() is supported only if DICT_MAINTAINORDER
00521  * flag was used in init_dict().
00522  */
00523 WOLFSENTRY_API int json_value_dict_walk_ordered(const JSON_VALUE* v,
00524             int (*visit_func)(const JSON_VALUE*, JSON_VALUE*, void*), void* ctx);
00525 WOLFSENTRY_API int json_value_dict_walk_sorted(const JSON_VALUE* v,
00526             int (*visit_func)(const JSON_VALUE*, JSON_VALUE*, void*), void* ctx);
00527
00528 /* Remove and destroy all members (recursively).
00529  */
00530 WOLFSENTRY_API int json_value_dict_clean(
```

```
00531 #ifdef WOLFSENTRY
00532     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00533 #endif
00534     JSON_VALUE* v);
00535
00536 #ifdef WOLFSENTRY
00537 WOLFSENTRY_API int
00538 json_value_clone(WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_allocator *allocator),
00539                  const JSON_VALUE* node, JSON_VALUE *clone);
00540 #endif
00541
00542 #ifdef __cplusplus
00543 }
00544 #endif
00545
00546 #endif  /* CENTIJSON_VALUE_H */
```

10.4 wolfsentry/wolfsentry.h File Reference

The main include file for wolfSentry applications.

```
#include <wolfsentry/wolfsentry_settings.h>
#include <wolfsentry/wolfsentry_af.h>
#include <wolfsentry/wolfsentry_errcodes.h>
#include <wolfsentry/centijson_dom.h>
#include <wolfsentry/wolfsentry_util.h>
```

Data Structures

- struct wolfsentry_allocator

  Struct for passing shims that abstract the native implementation of the heap allocator.
- struct wolfsentry_timecbs

  Struct for passing shims that abstract the native implementation of time functions.
- struct wolfsentry_semcbs

  Struct for passing shims that abstract the native implementation of counting semaphores.
- struct wolfsentry_host_platform_interface

  struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores
- struct wolfsentry_route_endpoint

  struct for exporting socket addresses, with fixed-length fields
- struct wolfsentry_route_metadata_exports

  struct for exporting route metadata for access by applications
- struct wolfsentry_route_exports

  struct for exporting a route for access by applications
- struct wolfsentry_sockaddr

  struct for passing socket addresses into `wolfsentry_route_*()` API routines
- struct wolfsentry_eventconfig

  struct for representing event configuration
- struct wolfsentry_kv_pair

  public structure for passing user-defined values in/out of wolfSentry

Macros

- #define **WOLFSENTRY_VERSION_MAJOR**

  Macro for major version number of installed headers.
- #define **WOLFSENTRY_VERSION_MINOR**

  Macro for minor version number of installed headers.
- #define **WOLFSENTRY_VERSION_TINY**

  Macro for tiny version number of installed headers.
- #define **WOLFSENTRY_VERSION_ENCODE**(major, minor, tiny)

  Macro to convert a wolfSentry version to a single integer, for comparison to other similarly converted versions.
- #define **WOLFSENTRY_VERSION**

  The version recorded in wolfsentry.h, encoded as an integer.
- #define **WOLFSENTRY_VERSION_GT**(major, minor, tiny)

  Helper macro that is true if the given version is greater than that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_GE**(major, minor, tiny)

  Helper macro that is true if the given version is greater than or equal to that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_EQ**(major, minor, tiny)

  Helper macro that is true if the given version equals that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_LT**(major, minor, tiny)

  Helper macro that is true if the given version is less than that in wolfsentry.h.
- #define **WOLFSENTRY_VERSION_LE**(major, minor, tiny)

  Helper macro that is true if the given version is less than or equal to that in wolfsentry.h.
- #define **WOLFSENTRY_CONTEXT_ARGS_IN**

  Common context argument generator for use at the beginning of arg lists in function prototypes and definitions. Pair with `WOLFSENTRY_CONTEXT_ARGS_OUT` in the caller argument list.
- #define **WOLFSENTRY_CONTEXT_ARGS_IN_EX**(ctx)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` that allows a fully type-qualified `context` to be supplied explicitly (allowing contexts other than `struct wolfsentry_context`)
- #define **WOLFSENTRY_CONTEXT_ARGS_IN_EX4**(ctx, thr)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` that allows the identifiers for `context` and `thread` pointers to be supplied explicitly.
- #define **WOLFSENTRY_CONTEXT_ELEMENTS**

  Variant of `WOLFSENTRY_CONTEXT_ARGS_IN` for constructing `struct`s.
- #define **WOLFSENTRY_CONTEXT_SET_ELEMENTS**(s)

  Counterpart to `WOLFSENTRY_CONTEXT_ELEMENTS` to access the `wolfsentry` context.
- #define **WOLFSENTRY_CONTEXT_GET_ELEMENTS**(s)

  Counterpart to `WOLFSENTRY_CONTEXT_ELEMENTS` to access the `thread` context (exists only if `defined(WOLFSENTRY_THREADSAFE)`)
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT**

  Common context argument generator to use in calls to functions taking `WOLFSENTRY_CONTEXT_ARGS_IN`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX**(ctx)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` that allows passing an explicitly identified context argument generator to use in calls to functions taking `WOLFSENTRY_CONTEXT_ARGS_IN_EX`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX2**(x)

  Variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` corresponding to `WOLFSENTRY_CONTEXT_ELEMENTS`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX3**(x, y)

  Special-purpose variant of `WOLFSENTRY_CONTEXT_ARGS_OUT_EX` for accessing context element `y` in structure pointer `x`
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX4**(x, y)

  Special-purpose variant of `WOLFSENTRY_CONTEXT_ARGS_OUT` that simply expands to `x` or `x, y` depending on `WOLFSENTRY_THREADSAFE`
- #define **WOLFSENTRY_CONTEXT_ARGS_NOT_USED**

Helper macro for function implementations that need to accept `WOLFSENTRY_CONTEXT_ARGS_IN` for API conformance, but don't actually use the arguments.

- #define **WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED**

  Helper macro for function implementations that need to accept `WOLFSENTRY_CONTEXT_ARGS_IN` for API conformance, but don't actually use the `thread` argument.
- #define **WOLFSENTRY_THREAD_HEADER_DECLS**

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack.
- #define **WOLFSENTRY_THREAD_HEADER_INIT**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this performs the required thread context initialization, with options from its `wolfsentry_thread_flags_t flags` arg.
- #define **WOLFSENTRY_THREAD_HEADER_INIT_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this performs the required thread context initialization, with options from its `wolfsentry_thread_flags_t flags` arg, and returns on failure.
- #define **WOLFSENTRY_THREAD_HEADER**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack, and initializes it with options from its `wolfsentry_thread_flags_t flags` arg.
- #define **WOLFSENTRY_THREAD_HEADER_CHECK**()

  For `WOLFSENTRY_THREADSAFE` applications, checks if thread context initialization succeeded, and returns on failure.
- #define **WOLFSENTRY_THREAD_HEADER_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this allocates the required thread context on the stack, and initializes it with options from its `wolfsentry_thread_flags_t flags` arg, returning on failure.
- #define **WOLFSENTRY_THREAD_TAILER**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this cleans up a thread context allocated with `WOLFSENTRY_THREAD_HEADER*`, with options from its `wolfsentry_thread_flags_t flags` arg, storing the result.
- #define **WOLFSENTRY_THREAD_TAILER_CHECKED**(flags)

  For `WOLFSENTRY_THREADSAFE` applications, this cleans up a thread context allocated with `WOLFSENTRY_THREAD_HEADER*`, with options from its `wolfsentry_thread_flags_t flags` arg, returning on error.
- #define **WOLFSENTRY_THREAD_GET_ERROR**

  For `WOLFSENTRY_THREADSAFE` applications, this evaluates to the most recent result from `WOLFSENTRY_THREAD_HEADER_INIT` or `WOLFSENTRY_THREAD_TAILER()`
- #define **WOLFSENTRY_ACTION_RES_USER_SHIFT** 24U

  Bit shift for user-defined bit span in wolfsentry_action_res_t.
- #define **WOLFSENTRY_ACTION_RES_USER7** (1U << 31U)

  user-defined result bit #8 of 8. Defined with a macro to retain ISO C compliance on enum range.
- #define **WOLFSENTRY_ROUTE_DEFAULT_POLICY_MASK** (WOLFSENTRY_ACTION_RES_ACCEPT | WOLFSENTRY_ACTION_RES_REJECT | WOLFSENTRY_ACTION_RES_STOP | WOLFSENTRY_ACTION_RES_ERROR)

  Bit mask spanning the bits allowed by wolfsentry_route_table_default_policy_set()
- #define **WOLFSENTRY_ROUTE_WILDCARD_FLAGS**

  Bit mask for the wildcard bits in a wolfsentry_route_flags_t.
- #define **WOLFSENTRY_ROUTE_IMMUTABLE_FLAGS**

  Bit mask for the bits in a wolfsentry_route_flags_t that can't change after the implicated route has been inserted in the route table.
- #define WOLFSENTRY_ROUTE_INTERNAL_FLAGS
- #define **WOLFSENTRY_SOCKADDR**(n)

  Macro to instantiate a wolfsentry_sockaddr with an `addr` field sized to hold `n` bits of address data. Cast to `struct wolfsentry_sockaddr` to pass as API argument.
- #define **WOLFSENTRY_LENGTH_NULL_TERMINATED**

  A macro with a painfully long name that can be passed as a length to routines taking a length argument, to signify that the associated string is null-terminated and its length should be computed on that basis.
- #define **WOLFSENTRY_KV_FLAG_MASK**

  A bit mask to retain only the flag bits in a `wolfsentry_kv_type_t`.
- #define **WOLFSENTRY_KV_KEY_LEN**(kv)

Evaluates to the length of the key of a `wolfsentry_kv_pair`.

- #define **WOLFSENTRY_KV_KEY**(kv)

  Evaluates to the key of a `wolfsentry_kv_pair`.
- #define **WOLFSENTRY_KV_TYPE**(kv)

  Evaluates to the type of a `wolfsentry_kv_pair`, with flag bits masked out.
- #define **WOLFSENTRY_KV_V_UINT**(kv)

  Evaluates to the `uint64_t` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_UINT`.
- #define **WOLFSENTRY_KV_V_SINT**(kv)

  Evaluates to the `int64_t` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_INT`.
- #define **WOLFSENTRY_KV_V_FLOAT**(kv)

  Evaluates to the `double` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_FLOAT`.
- #define **WOLFSENTRY_KV_V_STRING_LEN**(kv)

  Evaluates to the `size_t` length of the value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_STRING`.
- #define **WOLFSENTRY_KV_V_STRING**(kv)

  Evaluates to the `char *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_STRING`.
- #define **WOLFSENTRY_KV_V_BYTES_LEN**(kv)

  Evaluates to the `size_t` length of the value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.
- #define **WOLFSENTRY_KV_V_BYTES**(kv)

  Evaluates to the `byte *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.
- #define **WOLFSENTRY_KV_V_JSON**(kv)

  Evaluates to the `JSON_VALUE *` value of a `wolfsentry_kv_pair` of type `WOLFSENTRY_KV_JSON`.
- #define **WOLFSENTRY_BASE64_DECODED_BUFSPC**(buf, len)

  Given valid base64 string `buf` of length `len`, evaluates to the exact decoded length.

Typedefs

- typedef void *(* **wolfsentry_malloc_cb_t**) (void *context, struct wolfsentry_thread_context *thread, size_t size)

  Pointer to malloc-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void(* **wolfsentry_free_cb_t**) (void *context, struct wolfsentry_thread_context *thread, void *ptr)

  Pointer to free-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void *(* **wolfsentry_realloc_cb_t**) (void *context, struct wolfsentry_thread_context *thread, void *ptr, size_t size)

  Pointer to realloc-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void *(* **wolfsentry_memalign_cb_t**) (void *context, struct wolfsentry_thread_context *thread, size_t alignment, size_t size)

  Pointer to memalign-like function. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef void(* **wolfsentry_free_aligned_cb_t**) (void *context, struct wolfsentry_thread_context *thread, void *ptr)

  Pointer to special-purpose free-like function, needed only if the `memalign` pointer in a `struct wolfsentry_allocator` is non-null. Can be same as routine supplied as `wolfsentry_free_cb_t`, or can be a separate routine, e.g. with special handling for pad bytes. Takes extra initial args `context` and, if `!defined(WOLFSENTRY_SINGLETHREADED)`, `thread` arg.
- typedef wolfsentry_errcode_t(* **wolfsentry_get_time_cb_t**) (void *context, wolfsentry_time_t *ts)

  Pointer to function that returns time denominated in `wolfsentry_time_t`. Takes an initial `context` arg, which can be ignored.
- typedef wolfsentry_time_t(* **wolfsentry_diff_time_cb_t**) (wolfsentry_time_t earlier, wolfsentry_time_t later)

  Pointer to function that subtracts `earlier` from `later`, returning the result.

- typedef wolfsentry_time_t(* **wolfsentry_add_time_cb_t**) (wolfsentry_time_t start_time, wolfsentry_time_t time_interval)

  Pointer to function that adds two `wolfsentry_time_t` times, returning the result.
- typedef wolfsentry_errcode_t(* **wolfsentry_to_epoch_time_cb_t**) (wolfsentry_time_t when, time_t *epoch_secs, long *epoch_nsecs)

  Pointer to function that converts a `wolfsentry_time_t` to seconds and nanoseconds since midnight UTC, 1970-Jan-1.
- typedef wolfsentry_errcode_t(* **wolfsentry_from_epoch_time_cb_t**) (time_t epoch_secs, long epoch_nsecs, wolfsentry_time_t *when)

  Pointer to function that converts seconds and nanoseconds since midnight UTC, 1970-Jan-1, to a `wolfsentry_time_t`.
- typedef wolfsentry_errcode_t(* **wolfsentry_interval_to_seconds_cb_t**) (wolfsentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs)

  Pointer to function that converts a `wolfsentry_time_t` expressing an interval to the corresponding seconds and nanoseconds.
- typedef wolfsentry_errcode_t(* **wolfsentry_interval_from_seconds_cb_t**) (time_t howlong_secs, long howlong_nsecs, wolfsentry_time_t *howlong)

  Pointer to function that converts seconds and nanoseconds expressing an interval to the corresponding `wolfsentry_time_t`.
- typedef int(* sem_init_cb_t) (sem_t *sem, int pshared, unsigned int value)
- typedef int(* sem_post_cb_t) (sem_t *sem)
- typedef int(* sem_wait_cb_t) (sem_t *sem)
- typedef int(* sem_timedwait_cb_t) (sem_t *sem, const struct timespec *abs_timeout)
- typedef int(* sem_trywait_cb_t) (sem_t *sem)
- typedef int(* sem_destroy_cb_t) (sem_t *sem)
- typedef wolfsentry_errcode_t(* wolfsentry_action_callback_t) (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_action *action, void *handler_arg, void *caller_arg, const struct wolfsentry_event *trigger_event, wolfsentry_action_type_t action_type, const struct wolfsentry_route *trigger_route, struct wolfsentry_route_table *route_table, struct wolfsentry_route *rule_route, wolfsentry_action_res_t *action_results)

  A callback that is triggered when an action is taken.
- typedef wolfsentry_errcode_t(* **wolfsentry_make_id_cb_t**) (void *context, wolfsentry_ent_id_t *id)
- typedef void(* **wolfsentry_cleanup_callback_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, void *cleanup_arg)

  Function type to pass to wolfsentry_cleanup_push()
- typedef wolfsentry_errcode_t(* **wolfsentry_addr_family_parser_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, const char *addr_text, int addr_text_len, byte *addr_internal, wolfsentry_addr_bits_t *addr_internal_bits)

  Function type for parsing handler, to pass to wolfsentry_addr_family_handler_install()
- typedef wolfsentry_errcode_t(* **wolfsentry_addr_family_formatter_t**) (WOLFSENTRY_CONTEXT_ARGS_IN, const byte *addr_internal, unsigned int addr_internal_bits, char *addr_text, int *addr_text_len)

  Function type for formatting handler, to pass to wolfsentry_addr_family_handler_install()
- typedef wolfsentry_errcode_t(* wolfsentry_kv_validator_t) (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair *kv)

Enumerations

- enum wolfsentry_init_flags_t {
  WOLFSENTRY_INIT_FLAG_NONE ,
  WOLFSENTRY_INIT_FLAG_LOCK_SHARED_ERROR_CHECKING }

  flags to pass to wolfsentry_init_ex(), to be `OR`d together.
- enum wolfsentry_thread_flags_t {
  WOLFSENTRY_THREAD_FLAG_NONE ,
  WOLFSENTRY_THREAD_FLAG_DEADLINE ,
  WOLFSENTRY_THREAD_FLAG_READONLY }

wolfsentry_thread_flags_t flags are to be ORed together.

- enum wolfsentry_lock_flags_t {
  WOLFSENTRY_LOCK_FLAG_NONE ,
  WOLFSENTRY_LOCK_FLAG_PSHARED ,
  WOLFSENTRY_LOCK_FLAG_SHARED_ERROR_CHECKING ,
  WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_MUTEX ,
  WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_SHARED ,
  WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_TRY_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO ,
  WOLFSENTRY_LOCK_FLAG_AUTO_DOWNGRADE ,
  WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAPHORE }

  flags to pass to wolfsentry_lock_*() functions, to be ORd together
- enum wolfsentry_object_type_t {
  WOLFSENTRY_OBJECT_TYPE_UNINITED ,
  WOLFSENTRY_OBJECT_TYPE_TABLE ,
  WOLFSENTRY_OBJECT_TYPE_ACTION ,
  WOLFSENTRY_OBJECT_TYPE_EVENT ,
  WOLFSENTRY_OBJECT_TYPE_ROUTE ,
  WOLFSENTRY_OBJECT_TYPE_KV ,
  WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER ,
  WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME }

  enum for communicating the type of an object.
- enum wolfsentry_action_flags_t {
  WOLFSENTRY_ACTION_FLAG_NONE ,
  WOLFSENTRY_ACTION_FLAG_DISABLED }

  enum for communicating attributes of an action object
- enum wolfsentry_action_type_t {
  WOLFSENTRY_ACTION_TYPE_NONE ,
  WOLFSENTRY_ACTION_TYPE_POST ,
  WOLFSENTRY_ACTION_TYPE_INSERT ,
  WOLFSENTRY_ACTION_TYPE_MATCH ,
  WOLFSENTRY_ACTION_TYPE_UPDATE ,
  WOLFSENTRY_ACTION_TYPE_DELETE ,
  WOLFSENTRY_ACTION_TYPE_DECISION }

  enum communicating (to action handlers and internal logic) what type of action is being evaluated
- enum wolfsentry_action_res_t {
  WOLFSENTRY_ACTION_RES_NONE ,
  WOLFSENTRY_ACTION_RES_ACCEPT ,
  WOLFSENTRY_ACTION_RES_REJECT ,
  WOLFSENTRY_ACTION_RES_CONNECT ,
  WOLFSENTRY_ACTION_RES_DISCONNECT ,
  WOLFSENTRY_ACTION_RES_DEROGATORY ,
  WOLFSENTRY_ACTION_RES_COMMENDABLE ,
  WOLFSENTRY_ACTION_RES_STOP ,
  WOLFSENTRY_ACTION_RES_DEALLOCATED ,
  WOLFSENTRY_ACTION_RES_INSERTED ,
  WOLFSENTRY_ACTION_RES_ERROR ,
  WOLFSENTRY_ACTION_RES_FALLTHROUGH ,
  WOLFSENTRY_ACTION_RES_UPDATE ,
  WOLFSENTRY_ACTION_RES_PORT_RESET ,
  WOLFSENTRY_ACTION_RES_SENDING ,
  WOLFSENTRY_ACTION_RES_RECEIVED ,
  WOLFSENTRY_ACTION_RES_BINDING ,
  WOLFSENTRY_ACTION_RES_LISTENING ,
  WOLFSENTRY_ACTION_RES_STOPPED_LISTENING ,
  WOLFSENTRY_ACTION_RES_CONNECTING_OUT ,

WOLFSENTRY_ACTION_RES_CLOSED ,
WOLFSENTRY_ACTION_RES_UNREACHABLE ,
WOLFSENTRY_ACTION_RES_SOCK_ERROR ,
WOLFSENTRY_ACTION_RES_CLOSE_WAIT ,
WOLFSENTRY_ACTION_RES_USER0 ,
WOLFSENTRY_ACTION_RES_USER1 ,
WOLFSENTRY_ACTION_RES_USER2 ,
WOLFSENTRY_ACTION_RES_USER3 ,
WOLFSENTRY_ACTION_RES_USER4 ,
WOLFSENTRY_ACTION_RES_USER5 ,
WOLFSENTRY_ACTION_RES_USER6 }

bit field used to communicate states and attributes through the evaluation pipeline.

- enum wolfsentry_route_flags_t {
WOLFSENTRY_ROUTE_FLAG_NONE = 0U ,
WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD ,
WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS ,
WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN ,
WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT ,
WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR_BITMASK ,
WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_BITMASK ,
WOLFSENTRY_ROUTE_FLAG_IN_TABLE ,
WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE ,
WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED ,
WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED ,
WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED ,
WOLFSENTRY_ROUTE_FLAG_GREENLISTED ,
WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_HITS ,
WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_CURRENT_CONNECTIONS ,
WOLFSENTRY_ROUTE_FLAG_PORT_RESET }

bit field specifying attributes of a route/rule

- enum wolfsentry_format_flags_t {
WOLFSENTRY_FORMAT_FLAG_NONE ,
WOLFSENTRY_FORMAT_FLAG_ALWAYS_NUMERIC }

bit field with options for rendering

- enum wolfsentry_event_flags_t {
WOLFSENTRY_EVENT_FLAG_NONE ,
WOLFSENTRY_EVENT_FLAG_IS_PARENT_EVENT ,
WOLFSENTRY_EVENT_FLAG_IS_SUBEVENT }

bit field with attribute flags for events

- enum wolfsentry_eventconfig_flags_t {
WOLFSENTRY_EVENTCONFIG_FLAG_NONE ,
WOLFSENTRY_EVENTCONFIG_FLAG_DEROGATORY_THRESHOLD_IGNORE_COMMENDABLE ,
WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY ,
WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_ACTIONS }

bit field with config flags for events

- enum wolfsentry_clone_flags_t {
WOLFSENTRY_CLONE_FLAG_NONE ,
WOLFSENTRY_CLONE_FLAG_AS_AT_CREATION ,
WOLFSENTRY_CLONE_FLAG_NO_ROUTES }

Flags to be ORd together to control the dynamics of wolfsentry_context_clone() and other cloning functions.

- enum wolfsentry_kv_type_t {
  WOLFSENTRY_KV_NONE = 0 ,
  WOLFSENTRY_KV_NULL ,
  WOLFSENTRY_KV_TRUE ,
  WOLFSENTRY_KV_FALSE ,
  WOLFSENTRY_KV_UINT ,
  WOLFSENTRY_KV_SINT ,
  WOLFSENTRY_KV_FLOAT ,
  WOLFSENTRY_KV_STRING ,
  WOLFSENTRY_KV_BYTES ,
  WOLFSENTRY_KV_JSON ,
  WOLFSENTRY_KV_FLAG_READONLY = 1<<30 }

  enum to represent the type of a user-defined value

Functions

- WOLFSENTRY_API struct wolfsentry_build_settings **wolfsentry_get_build_settings** (void)

  Return the `wolfsentry_build_settings` of the library as built.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_build_settings_compatible** (struct wolfsentry_build_settings caller_build_settings)

  Return success if the application and library were built with mutually compatible wolfSentry version and configuration.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_init_thread_context** (struct wolfsentry_thread_context *thread_context, wolfsentry_thread_flags_t init_thread_flags, void *user_context)

  Initialize `thread_context` according to `init_thread_flags`, storing `user_context` for later retrieval with wolfsentry_get_thread_user_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_alloc_thread_context** (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context, wolfsentry_thread_flags_t init_thread_flags, void *user_context)

  Allocate space for `thread_context` using the allocator in `hpi`, then call wolfsentry_init_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_id** (struct wolfsentry_thread_context *thread, wolfsentry_thread_id_t *id)

  Write the `wolfsentry_thread_id_t` of `thread` to `id`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_user_context** (struct wolfsentry_thread_context *thread, void **user_context)

  Store to `user_context` the pointer previously passed to wolfsentry_init_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_deadline** (struct wolfsentry_thread_context *thread, struct timespec *deadline)

  Store the deadline for `thread` to `deadline`, or if the thread has no deadline set, store WOLFSENTRY_DEADLINE_NEVER to `deadline->tv_sec` and `deadline->tv_nsec`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_thread_flags** (struct wolfsentry_thread_context *thread, wolfsentry_thread_flags_t *thread_flags)

  Store the flags of `thread` to `thread_flags`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_destroy_thread_context** (struct wolfsentry_thread_context *thread_context, wolfsentry_thread_flags_t thread_flags)

  Perform final integrity checking on the thread state, and deallocate its ID.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_free_thread_context** (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context, wolfsentry_thread_flags_t thread_flags)

  Call `wolfsentry_destroy_thread_context()` on `*thread_context`, and if that succeeds, deallocate the thread object previously allocated by wolfsentry_alloc_thread_context().
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_deadline_rel_usecs** (WOLFSENTRY_CONTEXT_ARGS_IN, int usecs)

  Set the thread deadline to `usecs` in the future. The thread will not wait for a lock beyond that deadline.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_deadline_abs** (WOLFSENTRY_CONTEXT_ARGS_IN, time_t epoch_secs, long epoch_nsecs)

  Set the thread deadline to the time identified by `epoch_secs` and `epoch_nsecs`. The thread will not wait for a lock beyond that deadline.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_clear_deadline** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Clear any thread deadline previously set. On time-unbounded calls such as wolfsentry_lock_shared() and wolfsentry_lock_mutex(), the thread will sleep until the lock is available.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_thread_readonly** (struct wolfsentry_thread_↩ context *thread_context)

  Set the thread state to allow only readonly locks to be gotten, allowing multiple shared locks to be concurrently held. If any mutexes or reservations are currently held, the call will fail.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_set_thread_readwrite** (struct wolfsentry_thread_↩ context *thread_context)

  Set the thread state to allow both readonly and mutex locks to be gotten. If multiple shared locks are currently held, the call will fail.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_init (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock *lock, wolfsentry_lock_flags_t flags)

  This initializes a semaphore lock structure created by the user.
- WOLFSENTRY_API size_t **wolfsentry_lock_size** (void)
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_alloc (struct wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock **lock, wolfsentry_lock_flags_t flags)

  Allocates and initializes a semaphore lock structure for use with wolfSentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Requests a shared lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Requests a shared lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Requests a shared lock with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Requests an exclusive lock with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex2shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Downgrade an exclusive lock to a shared lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Upgrade a shared lock to an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abstimed (struct wolfsentry_↩ rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_reserve (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Attempt to reserve a upgrade of a shared lock to an exclusive lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_abstimed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive with an absolute timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_timed (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  Redeem a reservation of a lock upgrade from shared to exclusive with a relative timeout.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abandon (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Abandon a reservation of a lock upgrade from shared to exclusive.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in shared state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_mutex (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in exclusive state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_either (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if the lock is held in either shared or exclusive state.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared2mutex_reservation (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Check if an upgrade reservation is held on the lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_get_flags (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t *flags)

  Extract the current flags from the lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_unlock (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Unlock a lock.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_destroy (struct wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Destroy a lock that was created with wolfsentry_lock_init()
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_free (struct wolfsentry_rwlock **lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags)

  Destroy and free a lock that was created with wolfsentry_lock_alloc(). The lock's pointer will also be set to NULL.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_now_plus_delta** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t td, wolfsentry_time_t *res)

  Generate a wolfsentry_time_t at a given offset from current time.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_to_timespec** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t t, struct timespec *ts)

  Convert a wolfsentry_time_t to a `struct timespec`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_time_now_plus_delta_timespec** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t td, struct timespec *ts)

  Generate a `struct timespec` at a given offset, supplied as wolfsentry_time_t, from current time.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_get_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t *time_p)

  Get current time as wolfsentry_time_t.

- WOLFSENTRY_API wolfsentry_time_t **wolfsentry_diff_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t later, wolfsentry_time_t earlier)

  Compute the interval between `later` and `earlier`, using wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_time_t **wolfsentry_add_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t start_time, wolfsentry_time_t time_interval)

  Compute the time `time_interval` after `start_time`, using wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_to_epoch_time** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t when, time_t *epoch_secs, long *epoch_nsecs)

  Convert a wolfsentry_time_t to seconds and nanoseconds since 1970-Jan-1 0:00 UTC.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_from_epoch_time** (struct wolfsentry_context *wolfsentry, time_t epoch_secs, long epoch_nsecs, wolfsentry_time_t *when)

  Convert seconds and nanoseconds since 1970-Jan-1 0:00 UTC to a wolfsentry_time_t.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_interval_to_seconds** (struct wolfsentry_context *wolfsentry, wolfsentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs)

  Convert an interval in wolfsentry_time_t to seconds and nanoseconds.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_interval_from_seconds** (struct wolfsentry_context *wolfsentry, time_t howlong_secs, long howlong_nsecs, wolfsentry_time_t *howlong)

  Convert an interval in seconds and nanoseconds to wolfsentry_time_t.
- WOLFSENTRY_API struct wolfsentry_timecbs * **wolfsentry_get_timecbs** (struct wolfsentry_context *wolfsentry)

  Return the active time handlers from the supplied context.
- WOLFSENTRY_API void * **wolfsentry_malloc** (WOLFSENTRY_CONTEXT_ARGS_IN, size_t size)

  Allocate `size` bytes using the `malloc` configured in the wolfSentry context.
- WOLFSENTRY_API_VOID **wolfsentry_free** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr)

  Free `ptr` using the `free` configured in the wolfSentry context.
- WOLFSENTRY_API void * **wolfsentry_realloc** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr, size_t size)

  Reallocate `ptr` to `size` bytes using the `realloc` configured in the wolfSentry context.
- WOLFSENTRY_API void * **wolfsentry_memalign** (WOLFSENTRY_CONTEXT_ARGS_IN, size_t alignment, size_t size)

  Allocate `size` bytes, aligned to `alignment`, using the `memalign` configured in the wolfSentry context.
- WOLFSENTRY_API_VOID **wolfsentry_free_aligned** (WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr)

  Free `ptr`, previously allocated with `wolfsentry_memalign()`, using the `free_aligned` configured in the wolfSentry context.
- WOLFSENTRY_API int **_wolfsentry_get_n_mallocs** (void)

  In library builds with `WOLFSENTRY_MALLOC_BUILTINS` and `WOLFSENTRY_MALLOC_DEBUG` defined, this returns the net number of allocations performed as of time of call. I.e., it returns zero iff all allocations have been freed.
- WOLFSENTRY_API struct wolfsentry_allocator * **wolfsentry_get_allocator** (struct wolfsentry_context *wolfsentry)

  Return a pointer to the `wolfsentry_allocator` associated with the supplied `wolfsentry_context`, mainly for passing to `json_init()`, `json_parse()`, `json_value_*()`, and `json_dom_*()`.
- WOLFSENTRY_API const char * **wolfsentry_action_res_assoc_by_flag** (wolfsentry_action_res_t res, unsigned int bit)

  Given a `bit` number (from 0 to 31), return the name of that bit if set in `res`, else return a null pointer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_action_res_assoc_by_name** (const char *bit_name, int bit_name_len, wolfsentry_action_res_t *res)

  Given a `bit_name`, set `*res` to the corresponding bit number if known, failing which, return `ITEM_NOT_FOUND`.
- WOLFSENTRY_API struct wolfsentry_host_platform_interface * **wolfsentry_get_hpi** (struct wolfsentry_context *wolfsentry)

  Return a pointer to the `wolfsentry_host_platform_interface` associated with the supplied `wolfsentry_context`, mainly for passing to `wolfsentry_alloc_thread_context()`, `wolfsentry_free_thread_c` `wolfsentry_lock_init()`, and `wolfsentry_lock_alloc()`.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_push** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_cleanup_callback_t handler, void *arg)

  Register `handler` to be called at shutdown with arg `arg`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_pop** (WOLFSENTRY_CONTEXT_ARGS_IN, int execute_p)

  Remove the most recently registered and unpopped handler from the cleanup stack, and if `execute_p` is nonzero, call it with the `arg` with which it was registered.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_cleanup_all** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Iteratively call wolfsentry_cleanup_pop(), executing each handler as it is popped, passing it the `arg` with which it was registered.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_install** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family_bynumber, const char *family_byname, int family_byname_len, wolfsentry_addr_family_parser parser, wolfsentry_addr_family_formatter_t formatter, int max_addr_bits)

  Install handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_get_parser** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_family_parser_t *parser)

  Retrieve the parsing handler for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_get_formatter** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_family_formatter_t *formatter)

  Retrieve the formatting handler for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_remove_bynumber** (WOLFSENTRY_CONTEX wolfsentry_addr_family_t family_bynumber, wolfsentry_action_res_t *action_results)

  Remove the handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_drop_reference** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_addr_family_bynumber *family_bynumber, wolfsentry_action_res_t *action_results)

  Release an address family record previously returned by wolfsentry_addr_family_ntop()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_handler_remove_byname** (WOLFSENTRY_CONTEXT_ const char *family_byname, int family_byname_len, wolfsentry_action_res_t *action_results)

  Remove the handlers for an address family.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_pton** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *family_name, int family_name_len, wolfsentry_addr_family_t *family_number)

  Look up an address family by name, returning its number.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_ntop** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, struct wolfsentry_addr_family_bynumber **addr_family, const char **family_name)

  Look up an address family by number, returning a pointer to its name. The caller must release `addr_family`, using wolfsentry_addr_family_drop_reference(), when done accessing `family_name`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_addr_family_max_addr_bits** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t family, wolfsentry_addr_bits_t *bits)

  Look up the max address size for an address family identified by number.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_init (struct wolfsentry_context *wolfsentry, struct wolfsentry_eventconfig *config)

  Initializes a wolfsentry_eventconfig struct with the defaults from the wolfsentry context. If no wolfsentry context is provided this will initialize to zero.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_check (const struct wolfsentry_eventconfig *config)

  Checks the config for self-consistency and validity.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_init_ex** (struct wolfsentry_build_settings caller_↩ build_settings, WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi), const struct wolfsentry_eventconfig *config, struct wolfsentry_context **wolfsentry, wolfsentry_init_flags_t flags)

  Variant of wolfsentry_init() that accepts a `flags` argument, for additional control over configuration.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init (struct wolfsentry_build_settings caller_build_↩ settings, WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi), const struct wolfsentry_eventconfig *config, struct wolfsentry_context **wolfsentry)

Allocates and initializes the wolfsentry context.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_get (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_eventconfig *config)

  Get the default config from a wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_update (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_eventconfig *config)

  Updates mutable fields of the default config (all but wolfsentry_eventconfig::route_private_data_size and wolfsentry_eventconfig::route_private_data_alignment)
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_flush (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flushes the route, event, and user value tables from the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_free (WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry))

  Frees the wolfsentry context and the tables within it. The wolfsentry context will be a pointer to NULL upon success.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_shutdown (WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfsentry_context **wolfsentry))

  Shut down wolfSentry, freeing all resources. Gets an exclusive lock on the context, then calls wolfsentry_context_free().
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_inhibit_actions (WOLFSENTRY_CONTEXT_ARGS_IN)

  Disable automatic dispatch of actions on the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_enable_actions (WOLFSENTRY_CONTEXT_ARGS_IN)

  Re-enable automatic dispatch of actions on the wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_clone (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_context **clone, wolfsentry_clone_flags_t flags)

  Clones a wolfsentry context.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_exchange (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_context *wolfsentry2)

  Swaps information between two wolfsentry contexts.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_mutex() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_abstimed** (WOLFSENTRY_CONTEXT_ARGS_I const struct timespec *abs_timeout)

  Calls wolfsentry_lock_mutex_abstimed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_abstimed_ex** (WOLFSENTRY_CONTEXT_ARG const struct timespec *abs_timeout, wolfsentry_lock_flags_t flags)

  variant of wolfsentry_context_lock_mutex_abstimed() with a `flags` arg.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_mutex_timed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_mutex_timed_ex** (WOLFSENTRY_CONTEXT_ARGS_I wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags)

  variant of wolfsentry_context_lock_mutex_timed() with a `flags` arg.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_shared() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_abstimed** (WOLFSENTRY_CONTEXT_ARGS_ const struct timespec *abs_timeout)

  Calls wolfsentry_lock_shared_abstimed() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_with_reservation_abstimed** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct timespec *abs_timeout)

  Calls wolfsentry_lock_shared_abstimed() on the context, with the `WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO` flag.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_shared_timed() on the context.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_lock_shared_with_reservation_timed** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_time_t max_wait)

  Calls wolfsentry_lock_shared_timed() on the context, with the `WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO` flag.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_unlock** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_unlock() on the context.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_context_unlock_and_abandon_reservation** (WOLFSENTRY_CONTEXT_ARGS_IN)

  Calls wolfsentry_lock_unlock() on the context, with the `WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO` flag.
- WOLFSENTRY_API wolfsentry_object_type_t wolfsentry_get_object_type (const void *object)

  Get the object type from a wolfsentry object pointer.
- WOLFSENTRY_API wolfsentry_ent_id_t wolfsentry_get_object_id (const void *object)

  Get the ID from a wolfsentry object pointer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_table_ent_get_by_id** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, struct wolfsentry_table_ent_header **ent)

  Retrieve an object pointer given its ID. Lock must be obtained before entry, and ent is only valid while lock is held, or if wolfsentry_object_checkout() is called for the object.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_object_checkout** (WOLFSENTRY_CONTEXT_ARGS_IN, void *object)

  Increment the refcount for an object, making it safe from deallocation until wolfsentry_object_release(). Caller must have a context lock on entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_object_release** (WOLFSENTRY_CONTEXT_ARGS_IN, void *object, wolfsentry_action_res_t *action_results)

  Decrement the refcount for an object, deallocating it if no references remain. Caller does not need to have a context lock on entry.
- WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_inserts (struct wolfsentry_table_header *table)

  Get the number of inserts into a table.
- WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_deletes (struct wolfsentry_table_header *table)

  Get the number of deletes from a table.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_check_flags_sensical** (wolfsentry_route_flags_t flags)

  Check the self-consistency of `flags`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_into_table** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_into_table** (WOLFSENTRY_CONTEXT_AR struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_route_exports *route_exports, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports, and takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Insert a route into the route table.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_route_exports *route_exports, wolfsentry_ent_id_t *id, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_into_table_and_check_out** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label↩ _len, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that takes an explicit `route_table`, and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_into_table_and_↩ check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_route_exports *route_exports, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports, takes an explicit `route_table`, and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_and_check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_insert_by_exports_and_check_out** (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_route_exports *route↩ _exports, struct wolfsentry_route **route, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_insert() that accepts the new route as wolfsentry_route_exports and returns the inserted route, which the caller must eventually drop using wolfsentry_route_drop_reference() or wolfsentry_object_release()
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_delete_from_table** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label↩ _len, wolfsentry_action_res_t *action_results, int *n_deleted)

  Variant of wolfsentry_route_delete() that takes an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *trigger_label, int trigger_label_len, wolfsentry_action_res_t *action_results, int *n_deleted)

  Delete route from the route table. The supplied parameters, including the flags, must match the route exactly, else `ITEM_NOT_FOUND` will result. To avoid fidgety parameter matching, use wolfsentry_route_delete_by_id(). The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_by_id (WOLFSENTRY_CONTEXT_ARGS_IN, void *caller_arg, wolfsentry_ent_id_t id, const char *trigger_label, int trigger_label_len, wolfsentry_action_res_t *action_results)

  Delete a route from its route table using its ID. The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_main_table (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table **table)

  Get a pointer to the internal route table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_start (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor)

  Open a cursor to interate through a routes table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_head (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor)

  Reset the cursor to the beginning of a table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_tail (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor)

  Move the cursor to the end of a table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_current (const struct wolfsentry↩ _route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

Get the current position for the table cursor.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_prev (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

  Get the previous position for the table cursor.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_next (const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, struct wolfsentry_route **route)

  Get the next position for the table cursor.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_end (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor)

  Frees the table cursor. Caller must have a lock on the context at entry.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_set (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t default_policy)

  Set a table's default policy.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_default_policy_set** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t default_policy)

  variant of wolfsentry_route_table_default_policy_set() that uses the main route table implicitly, and takes care of context locking.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_get (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *default_policy)

  Get a table's default policy. Caller must have a lock on the context at entry.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_default_policy_get** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t *default_policy)

  variant of wolfsentry_route_table_default_policy_get() that uses the main route table implicitly. Caller must have a lock on the context at entry.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, int exact_p, wolfsentry_route_flags_t *inexact_matches, struct wolfsentry_route **route)

  Increments a reference counter for a route.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_action_res_t *action_results)

  Decrease a reference counter for a route.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_clear_default_event** (WOLFSENTRY_CONTEXT_ARGS, struct wolfsentry_route_table *table)

  Clear an event previously set by wolfsentry_route_table_set_default_event().

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_set_default_event** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *table, const char *event_label, int event_label_len)

  Set an event to be used as a foster parent event for routes with no parent event of their own.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_get_default_event** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *table, char *event_label, int *event_label_len)

  Get the event, if any, set by wolfsentry_route_table_set_default_event()

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_fallthrough_route_get (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route_table *route_table, const struct wolfsentry_route **fallthrough_route)

  Retrieve the default route in a route table, chiefly to pass to wolfsentry_route_update_flags().

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs (const struct wolfsentry_route *route, wolfsentry_addr_family_t *af, wolfsentry_addr_bits_t *local_addr_len, const byte **local_addr, wolfsentry_addr_bits_t *remote_addr_len, const byte **remote_addr)

  Extract numeric address family and binary address pointers from a `wolfsentry_route`

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_export (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *route, struct wolfsentry_route_exports *route_exports)

  Exports a route.

- WOLFSENTRY_API const struct wolfsentry_event * wolfsentry_route_parent_event (const struct wolfsentry_route *route)

Get a parent event from a given route. Typically used in the wolfsentry_action_callback_t callback. Note: returned wolfsentry_event remains valid only as long as the wolfsentry lock is held (shared or exclusive).

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_table** (WOLFSENTRY_CONTEXT_ARGS struct wolfsentry_route_table *route_table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that accepts an explicit `route_table`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Submit an event into wolfsentry and pass it through the filters. The action_results are cleared on entry, and can be checked to see what actions wolfsentry took, and what actions the caller should take (most saliently, WOLFSENTRY_ACTION_RES_ACCEPT or WOLFSENTRY_ACTION_RES_REJECT). `action_results` can be filtered with constructs like `WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_REJECT)`
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_table_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *route_table, const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that accepts an explicit `route_table`, and doesn't clear `action_results` on entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_with_inited_result** (WOLFSENTRY_CONTEX const struct wolfsentry_sockaddr *remote, const struct wolfsentry_sockaddr *local, wolfsentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_ent_id_t *id, wolfsentry_route_flags_t *inexact_matches, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that doesn't clear `action_results` on entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_id** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, mainly for use by application code that tracks ID/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_id_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, and doesn't clear `action_results` on entry, mainly for use by application code that tracks ID/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_route** (WOLFSENTRY_CONTEXT_ARGS_ struct wolfsentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, mainly for use by application code that tracks route/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_event_dispatch_by_route_with_inited_result** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_event_dispatch() that preselects the matched route by ID, and doesn't clear `action_results` on entry, mainly for use by application code that tracks route/session relationships.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_routes_get** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *table, wolfsentry_hitcount_t *max_purgeable_routes)

  Retrieve the current limit for ephemeral routes in `table`. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_routes_set** (WOLFSENTRY_CONTEXT struct wolfsentry_route_table *table, wolfsentry_hitcount_t max_purgeable_routes)

  Set the limit for ephemeral routes in `table`. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_idle_time_get** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_time_t *max_purgeable_idle_time)

Retrieve the current absolute maximum idle time for a purgeable route (controls forced purges of routes with nonzero wolfsentry_route_metadata_exports.connection_count). Caller must have a lock on the context at entry.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_max_purgeable_idle_time_set** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_time_t max_↩ purgeable_idle_time)

  Set the maximum idle time for a purgeable route (controls forced purges of routes with nonzero wolfsentry_route_metadata_exports.connec Default is no limit. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_purge_time_set** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_time_t purge_after)

  Set the time after which `route` in `table` is to be subject to automatic purge. `0` sets the route as persistent. Caller must have a mutex on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Purges all stale (expired) routes from `table`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_stale_purge_one** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_stale_purge() that purges at most one stale route, to limit time spent working.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_stale_purge_one_opportunistically** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Variant of wolfsentry_route_stale_purge() that purges at most one stale route, and only if the context lock is uncontended.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flush_table (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route_table *table, wolfsentry_action_res_t *action_results)

  Flush routes from a given table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_clear_insert_action_status (WOLFSENTRY_CONTEXT_ARG wolfsentry_action_res_t *action_results)

  Clears the WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED flag on all routes in the table.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_insert_actions (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_action_res_t *action_results)

  Executes the insert actions for all routes in the table that don't have WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED set.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_private_data (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, void **private_data, size_t *private_data_size)

  Gets the private data for a given route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_flags (const struct wolfsentry_route *route, wolfsentry_route_flags_t *flags)

  Gets the flags for a route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_metadata (const struct wolfsentry_route *route, struct wolfsentry_route_metadata_exports *metadata)

  Gets the metadata for a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_metadata_exports** (struct wolfsentry_route_exports *route_exports)

  clear metadata counts (wolfsentry_route_metadata_exports::purge_after, wolfsentry_route_metadata_exports::connection_count, wolfsentry_route_metadata_exports::derogatory_count, and wolfsentry_route_metadata_exports::commendable_count) in wolfsentry_route_exports to prepare for use with wolfsentry_route_insert_by_exports()
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_update_flags (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_route *route, wolfsentry_route_flags_t flags_to_set, wolfsentry_route_flags_t flags_to_↩ clear, wolfsentry_route_flags_t *flags_before, wolfsentry_route_flags_t *flags_after, wolfsentry_action_res_t *action_results)

  Update the route flags.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_increment_derogatory_count** (WOLFSENTRY_CONTEXT_AR struct wolfsentry_route *route, int count_to_add, int *new_derogatory_count_ptr)

  Increase the derogatory event count of a route.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_increment_commendable_count** (WOLFSENTRY_CONTEXT_ struct wolfsentry_route *route, int count_to_add, int *new_commendable_count)

  Increase the commendable event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_derogatory_count** (WOLFSENTRY_CONTEXT_ARGS_I struct wolfsentry_route *route, int *old_derogatory_count_ptr)

  Reset the derogatory event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_reset_commendable_count** (WOLFSENTRY_CONTEXT_ARG struct wolfsentry_route *route, int *old_commendable_count_ptr)

  Reset the commendable event count of a route.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_set_wildcard (struct wolfsentry_route *route, wolfsentry_route_flags_t wildcards_to_set)

  Set wildcard flags for a route.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_format_address** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_addr_family_t sa_family, const byte *addr, unsigned int addr_bits, char *buf, int *buflen)

  Render a binary address in human-readable form to a buffer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_flag_assoc_by_flag** (wolfsentry_route_flags_t flag, const char **name)

  Retrieve the name of a route flag, given its numeric value. Note that `flag` must have exactly one bit set, else `ITEM_NOT_FOUND` will be returned.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_flag_assoc_by_name** (const char *name, int len, wolfsentry_route_flags_t *flag)

  Retrieve the numeric value of a route flag, given its name.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_format_json** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *r, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Render a route to an output buffer, in JSON format, advancing the output buffer pointer by the length of the rendered output.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_start** (WOLFSENTRY_CONTEXT_ARGS_IN const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Start a rendering loop to export the route table contents as a JSON document that is valid input for wolfsentry_config_json_feed() or wolfsentry_config_json_oneshot(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount. Caller must have a shared or exclusive lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_next** (WOLFSENTRY_CONTEXT_ARGS_IN const struct wolfsentry_route_table *table, struct wolfsentry_cursor *cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Render a route within a loop started with wolfsentry_route_table_dump_json_start(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_table_dump_json_end** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_table *table, struct wolfsentry_cursor **cursor, unsigned char **json_out, size_t *json_out_len, wolfsentry_format_flags_t flags)

  Finish a rendering loop started with wolfsentry_route_table_dump_json_start(), advancing the output buffer pointer by the length of the rendered output, and decrementing `json_out_len` by the same amount.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_route_render_flags** (wolfsentry_route_flags_t flags, FILE *f)

  Render route flags in human-readable form to a stream.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route *r, FILE *f)

  Renders route information to a file pointer.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_exports_render (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_route_exports *r, FILE *f)

  Renders route exports information to a file pointer.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_insert (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_flags_t flags, wolfsentry_action_callback_t handler, void *handler_arg, wolfsentry_ent_id_t *id)

  Insert a new action into wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_res_t *action_results)

  Delete an action from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_flush_all (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flush all actions from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_action **action)

  Get a reference to an action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action *action, wolfsentry_action_res_t *action_results)

  Drop a reference to an action.
- WOLFSENTRY_API const char * wolfsentry_action_get_label (const struct wolfsentry_action *action)

  Get the label for an action. This is the internal pointer to the label so should not be freed by the application.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_flags (struct wolfsentry_action *action, wolfsentry_action_flags_t *flags)

  Get the flags for an action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_update_flags (struct wolfsentry_action *action, wolfsentry_action_flags_t flags_to_set, wolfsentry_action_flags_t flags_to_clear, wolfsentry_action_flags_t *flags_before, wolfsentry_action_flags_t *flags_after)

  Update the flags for an action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_insert (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_priority_t priority, const struct wolfsentry_eventconfig *config, wolfsentry_event_flags_t flags, wolfsentry_ent_id_t *id)

  Insert an event into wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, wolfsentry_action_res_t *action_results)

  Delete an event from wolfsentry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_flush_all (WOLFSENTRY_CONTEXT_ARGS_IN)

  Flush all events from wolfsentry.
- WOLFSENTRY_API const char * wolfsentry_event_get_label (const struct wolfsentry_event *event)

  Get the label for an event. This is the internal pointer to the label so should not be freed by the application.
- WOLFSENTRY_API wolfsentry_event_flags_t wolfsentry_event_get_flags (const struct wolfsentry_event *event)

  Get the flags for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_config (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_eventconfig *config)

  Get the configuration for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_update_config (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, const struct wolfsentry_eventconfig *config)

  Update the configuration for an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_reference (WOLFSENTRY_CONTEXT_ARGS_IN, const char *label, int label_len, struct wolfsentry_event **event)

  Get a reference to an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_drop_reference (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_event *event, wolfsentry_action_res_t *action_results)

  Drop a reference to an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_prepend (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

Prepend an action into an event.

- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_append (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

  Append an action into an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_insert_after (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len, const char *point_action_label, int point_action_label_len)

  Insert an action into an event after another action.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_delete (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, const char *action_label, int action_label_len)

  Delete an action from an event.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_set_aux_event (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, const char *aux_event_label, int aux_event_label_len)

  Set an auxiliary event for an event.
- WOLFSENTRY_API const struct wolfsentry_event * **wolfsentry_event_get_aux_event** (const struct wolfsentry_event *event)

  Retrieve an auxiliary event previously set with wolfsentry_event_set_aux_event().
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_start (WOLFSENTRY_CONTEXT_ARGS_IN, const char *event_label, int event_label_len, wolfsentry_action_type_t which_action_list, struct wolfsentry_action_list_ent **cursor)

  Open a cursor for the actions in an event. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_next (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action_list_ent **cursor, const char **action_label, int *action_label_len)

  Get the next action in an event cursor. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_done (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_action_list_ent **cursor)

  End iteration started with wolfsentry_event_action_list_start(). Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_set_validator** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_kv_validator_t validator, wolfsentry_action_res_t *action_results)

  Install a supplied `wolfsentry_kv_validator_t` to validate all user values before inserting them into the value table.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_set_mutability** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int mutable)

  Set the user-defined value with the designated `key` as readwrite (`mutable`=1) or readonly (`mutable`=0). A read-only value cannot be changed or deleted.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_mutability** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int *mutable)

  Query the mutability of the user-defined value with the designated `key`. Readonly value cannot be changed or deleted.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_type** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t *type)

  Returns the type of the value with the designated `key`, using `WOLFSENTRY_KV_TYPE()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_delete** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len)

  Deletes the value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_null** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_NULL` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bool** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_TRUE` or `WOLFSENTRY_KV_FALSE` value with the designated `key`.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_bool** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, wolfsentry_kv_type_t *value)

  Gets a `WOLFSENTRY_KV_TRUE` or `WOLFSENTRY_KV_FALSE` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_uint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, uint64_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_uint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, uint64_t *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_sint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int64_t value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_SINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_sint** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, int64_t *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_double** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, double value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_FLOAT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_get_float** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, double *value)

  Gets a `WOLFSENTRY_KV_UINT` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_string** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const char *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_STRING` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_string (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const char **value, int *value_len, struct wolfsentry_kv_pair_internal **user↩_value_record)

  Gets a `WOLFSENTRY_KV_STRING` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bytes** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const byte *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_BYTES` value with the designated `key` and a binary-clean `value`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_bytes_base64** (WOLFSENTRY_CONTEXT_ARGS const char *key, int key_len, const char *value, int value_len, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_BYTES` value with the designated `key` and a base64-encoded `value`.
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bytes (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, const byte **value, int *value_len, struct wolfsentry_kv_pair_internal **user↩_value_record)

  Gets a `WOLFSENTRY_KV_BYTES` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_store_json** (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, JSON_VALUE *value, int overwrite_p)

  Inserts or overwrites a `WOLFSENTRY_KV_JSON` value with the designated `key` and a `value` from `json_dom↩_parse()` (or built up programmatically with the `centijson_value.h` API).
- WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_json (WOLFSENTRY_CONTEXT_ARGS_IN, const char *key, int key_len, JSON_VALUE **value, struct wolfsentry_kv_pair_internal **user_value_record)

  Gets a `WOLFSENTRY_KV_JSON` value with the designated `key`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_value_release_record** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair_internal **user_value_record)

  Release a `user_value_record` from `wolfsentry_user_value_get_string()`, `wolfsentry_user_value_get_by` or `wolfsentry_user_value_get_json()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_pair_export** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_kv_pair_internal *kv, const struct wolfsentry_kv_pair **kv_exports)

  Extract the `struct wolfsentry_kv_pair` from a `struct wolfsentry_kv_pair_internal`. Caller must have a shared or exclusive lock on the context.

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_type_to_string** (wolfsentry_kv_type_t type, const char **out)

  Return a human-readable rendering of a `wolfsentry_kv_type_t`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_kv_render_value** (WOLFSENTRY_CONTEXT_ARGS_IN, const struct wolfsentry_kv_pair *kv, char *out, int *out_len)

  Render `kv` in human-readable form to caller-preallocated buffer `out`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_start** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor **cursor)

  Start an iteration loop on the user values table of this context. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_seek_to_head** (WOLFSENTRY_CONTEXT_ARG struct wolfsentry_cursor *cursor)

  Move the cursor to point to the start of the user values table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_seek_to_tail** (WOLFSENTRY_CONTEXT_ARGS struct wolfsentry_cursor *cursor)

  Move the cursor to point to the end of the user values table. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_current** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Return the item to which the cursor currently points, without moving the cursor. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_prev** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Move the cursor to the previous item, and return it. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_next** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor *cursor, struct wolfsentry_kv_pair_internal **kv)

  Move the cursor to the next item, and return it. Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_values_iterate_end** (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfsentry_cursor **cursor)

  End an iteration loop started with wolfsentry_user_values_iterate_start(). Caller must have a lock on the context at entry.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_base64_decode** (const char *src, size_t src_len, byte *dest, size_t *dest_spc, int ignore_junk_p)

  Convert base64-encoded input `src` to binary output `dest`, optionally ignoring (with nonzero `ignore_junk_p`) non-base64 characters in `src`.

10.4.1 Detailed Description

The main include file for wolfSentry applications.

Include this file in your application for core wolfSentry capabilities.

10.5 wolfsentry.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
```

```
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_H
00030 #define WOLFSENTRY_H
00031
00053 #define WOLFSENTRY_VERSION_MAJOR 1
00055 #define WOLFSENTRY_VERSION_MINOR 6
00057 #define WOLFSENTRY_VERSION_TINY 2
00059 #define WOLFSENTRY_VERSION_ENCODE(major, minor, tiny) (((major) « 16U) | ((minor) « 8U) | (tiny))
00061 #define WOLFSENTRY_VERSION WOLFSENTRY_VERSION_ENCODE(WOLFSENTRY_VERSION_MAJOR,
      WOLFSENTRY_VERSION_MINOR, WOLFSENTRY_VERSION_TINY)
00063 #define WOLFSENTRY_VERSION_GT(major, minor, tiny) (WOLFSENTRY_VERSION >
      WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00065 #define WOLFSENTRY_VERSION_GE(major, minor, tiny) (WOLFSENTRY_VERSION >=
      WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00067 #define WOLFSENTRY_VERSION_EQ(major, minor, tiny) (WOLFSENTRY_VERSION ==
      WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00069 #define WOLFSENTRY_VERSION_LT(major, minor, tiny) (WOLFSENTRY_VERSION <
      WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00071 #define WOLFSENTRY_VERSION_LE(major, minor, tiny) (WOLFSENTRY_VERSION <=
      WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00075 typedef enum {
00076     WOLFSENTRY_INIT_FLAG_NONE = 0,
00077     WOLFSENTRY_INIT_FLAG_LOCK_SHARED_ERROR_CHECKING = 1«0
00078 } wolfsentry_init_flags_t;
00079
00082 #ifndef WOLFSENTRY
00084 #define WOLFSENTRY /* activate wolfSentry codepaths in CentiJSON headers */
00086 #endif
00087
00088 #include <wolfsentry/wolfsentry_settings.h>
00089 #include <wolfsentry/wolfsentry_af.h>
00090 #include <wolfsentry/wolfsentry_errcodes.h>
00091
00092 struct wolfsentry_allocator;
00093 struct wolfsentry_context;
00094 struct wolfsentry_thread_context;
00095
00100 #ifdef WOLFSENTRY_THREADSAFE
00101
00102 typedef void *(*wolfsentry_malloc_cb_t)(void *context, struct wolfsentry_thread_context *thread,
      size_t size);
00104 typedef void (*wolfsentry_free_cb_t)(void *context, struct wolfsentry_thread_context *thread, void
      *ptr);
00108 typedef void *(*wolfsentry_realloc_cb_t)(void *context, struct wolfsentry_thread_context *thread, void
      *ptr, size_t size);
00112 typedef void *(*wolfsentry_memalign_cb_t)(void *context, struct wolfsentry_thread_context *thread,
      size_t alignment, size_t size);
00116 typedef void (*wolfsentry_free_aligned_cb_t)(void *context, struct wolfsentry_thread_context *thread,
      void *ptr);
00122 #else /* !WOLFSENTRY_THREADSAFE */
00123
00124 typedef void *(*wolfsentry_malloc_cb_t)(void *context, size_t size);
00125 typedef void (*wolfsentry_free_cb_t)(void *context, void *ptr);
00126 typedef void *(*wolfsentry_realloc_cb_t)(void *context, void *ptr, size_t size);
00127 typedef void *(*wolfsentry_memalign_cb_t)(void *context, size_t alignment, size_t size);
00128 typedef void (*wolfsentry_free_aligned_cb_t)(void *context, void *ptr);
00129
00130 #endif /* WOLFSENTRY_THREADSAFE */
00131
00133 struct wolfsentry_allocator {
00134     void *context;
00136     wolfsentry_malloc_cb_t malloc;
00138     wolfsentry_free_cb_t free;
00140     wolfsentry_realloc_cb_t realloc;
00142     wolfsentry_memalign_cb_t memalign;
00146     wolfsentry_free_aligned_cb_t free_aligned;
00148 };
00149
00156 typedef wolfsentry_errcode_t (*wolfsentry_get_time_cb_t)(void *context, wolfsentry_time_t *ts);
00159 typedef wolfsentry_time_t (*wolfsentry_diff_time_cb_t)(wolfsentry_time_t earlier, wolfsentry_time_t
      later);
00161 typedef wolfsentry_time_t (*wolfsentry_add_time_cb_t)(wolfsentry_time_t start_time, wolfsentry_time_t
      time_interval);
00163 typedef wolfsentry_errcode_t (*wolfsentry_to_epoch_time_cb_t)(wolfsentry_time_t when, time_t
      *epoch_secs, long *epoch_nsecs);
00165 typedef wolfsentry_errcode_t (*wolfsentry_from_epoch_time_cb_t)(time_t epoch_secs, long epoch_nsecs,
      wolfsentry_time_t *when);
00167 typedef wolfsentry_errcode_t (*wolfsentry_interval_to_seconds_cb_t)(wolfsentry_time_t howlong, time_t
```

```
    *howlong_secs, long *howlong_nsecs);
00169 typedef wolfsentry_errcode_t (*wolfsentry_interval_from_seconds_cb_t)(time_t howlong_secs, long
    howlong_nsecs, wolfsentry_time_t *howlong);
00173 struct wolfsentry_timecbs {
00174     void *context;
00176     wolfsentry_get_time_cb_t get_time;
00178     wolfsentry_diff_time_cb_t diff_time;
00180     wolfsentry_add_time_cb_t add_time;
00182     wolfsentry_to_epoch_time_cb_t to_epoch_time;
00184     wolfsentry_from_epoch_time_cb_t from_epoch_time;
00186     wolfsentry_interval_to_seconds_cb_t interval_to_seconds;
00188     wolfsentry_interval_from_seconds_cb_t interval_from_seconds;
00190 };
00191
00194 #ifdef WOLFSENTRY_THREADSAFE
00195
00200 typedef int (*sem_init_cb_t)(sem_t *sem, int pshared, unsigned int value);
00202 typedef int (*sem_post_cb_t)(sem_t *sem);
00204 typedef int (*sem_wait_cb_t)(sem_t *sem);
00206 typedef int (*sem_timedwait_cb_t)(sem_t *sem, const struct timespec *abs_timeout);
00208 typedef int (*sem_trywait_cb_t)(sem_t *sem);
00210 typedef int (*sem_destroy_cb_t)(sem_t *sem);
00214 struct wolfsentry_semcbs {
00215     sem_init_cb_t sem_init;
00217     sem_post_cb_t sem_post;
00219     sem_wait_cb_t sem_wait;
00221     sem_timedwait_cb_t sem_timedwait;
00223     sem_trywait_cb_t sem_trywait;
00225     sem_destroy_cb_t sem_destroy;
00227 };
00228
00231 #endif /* WOLFSENTRY_THREADSAFE */
00232
00238 struct wolfsentry_host_platform_interface {
00239     struct wolfsentry_build_settings caller_build_settings;
00240  /* must be first */
00241     struct wolfsentry_allocator allocator;
00243     struct wolfsentry_timecbs timecbs;
00245 #ifdef WOLFSENTRY_THREADSAFE
00246     struct wolfsentry_semcbs semcbs;
00248 #endif
00249 };
00250
00251 WOLFSENTRY_API struct wolfsentry_build_settings wolfsentry_get_build_settings(void);
00253 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_build_settings_compatible(struct
    wolfsentry_build_settings caller_build_settings);
00258 #ifdef WOLFSENTRY_THREADSAFE
00259
00265 typedef enum {
00266     WOLFSENTRY_THREAD_FLAG_NONE = 0,
00268     WOLFSENTRY_THREAD_FLAG_DEADLINE = 1«0,
00270     WOLFSENTRY_THREAD_FLAG_READONLY = 1«1
00272 } wolfsentry_thread_flags_t;
00273
00274 #define WOLFSENTRY_CONTEXT_ARGS_IN struct wolfsentry_context *wolfsentry, struct
    wolfsentry_thread_context *thread
00276 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX(ctx) ctx, struct wolfsentry_thread_context *thread
00281 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX4(ctx, thr) struct wolfsentry_context *ctx, struct
    wolfsentry_thread_context *thr
00283 #define WOLFSENTRY_CONTEXT_ELEMENTS struct wolfsentry_context *wolfsentry; struct
    wolfsentry_thread_context *thread
00285 #define WOLFSENTRY_CONTEXT_SET_ELEMENTS(s) (s).wolfsentry = wolfsentry; (s).thread = thread
00287 #define WOLFSENTRY_CONTEXT_GET_ELEMENTS(s) (s).wolfsentry, (s).thread
00289 #define WOLFSENTRY_CONTEXT_ARGS_OUT wolfsentry, thread
00291 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX(ctx) ctx, thread
00293 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX2(x) (x)->wolfsentry, (x)->thread
00295 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX3(x, y) (x)->y, (x)->thread
00297 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(x, y) x, y
00299 #define WOLFSENTRY_CONTEXT_ARGS_NOT_USED (void)wolfsentry; (void)thread
00301 #define WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED (void)thread
00304 /* note WOLFSENTRY_THREAD_HEADER_DECLS includes final semicolon. */
00305 #define WOLFSENTRY_THREAD_HEADER_DECLS                                          \
00306     struct wolfsentry_thread_context_public thread_buffer =             \
00307         WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER;                   \
00308     struct wolfsentry_thread_context *thread =                          \
00309         (struct wolfsentry_thread_context *)&thread_buffer;             \
00310     wolfsentry_errcode_t _thread_context_ret;
00313 #define WOLFSENTRY_THREAD_HEADER_INIT(flags)                                    \
00314     (_thread_context_ret =                                              \
00315         wolfsentry_init_thread_context(thread, flags, NULL /* user_context */))
00318 #define WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(flags)                            \
00319     do {                                                                \
00320         _thread_context_ret =                                           \
00321             wolfsentry_init_thread_context(thread, flags, NULL /* user_context */); \
00322         if (_thread_context_ret < 0)                                    \
00323             return _thread_context_ret;                                 \
00324     } while (0)
```

```
00327 #define WOLFSENTRY_THREAD_HEADER(flags)                                        \
00328     struct wolfsentry_thread_context_public thread_buffer =                    \
00329         WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER;                          \
00330     struct wolfsentry_thread_context *thread =                                 \
00331         (struct wolfsentry_thread_context *)&thread_buffer;                    \
00332     wolfsentry_errcode_t _thread_context_ret =                                 \
00333         wolfsentry_init_thread_context(thread, flags, NULL /* user_context */)
00336 #define WOLFSENTRY_THREAD_HEADER_CHECK()                                       \
00337     do {                                                                       \
00338         if (_thread_context_ret < 0)                                           \
00339             return _thread_context_ret;                                        \
00340     } while (0)
00343 #define WOLFSENTRY_THREAD_HEADER_CHECKED(flags)                                \
00344     WOLFSENTRY_THREAD_HEADER(flags);                                           \
00345     WOLFSENTRY_THREAD_HEADER_CHECK()
00348 #define WOLFSENTRY_THREAD_TAILER(flags) (_thread_context_ret =
    wolfsentry_destroy_thread_context(thread, flags))
00350 #define WOLFSENTRY_THREAD_TAILER_CHECKED(flags) do { WOLFSENTRY_THREAD_TAILER(flags); if
    (_thread_context_ret < 0) return _thread_context_ret; } while (0)
00352 #define WOLFSENTRY_THREAD_GET_ERROR _thread_context_ret
00356 typedef enum {
00357     WOLFSENTRY_LOCK_FLAG_NONE = 0,
00359     WOLFSENTRY_LOCK_FLAG_PSHARED = 1<<0,
00361     WOLFSENTRY_LOCK_FLAG_SHARED_ERROR_CHECKING = 1<<1,
00363     WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_MUTEX = 1<<2,
00365     WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_SHARED = 1<<3,
00367     WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO = 1<<4,
00369     WOLFSENTRY_LOCK_FLAG_TRY_RESERVATION_TOO = 1<<5,
00371     WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO = 1<<6,
00373     WOLFSENTRY_LOCK_FLAG_AUTO_DOWNGRADE = 1<<7,
00375     WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAPHORE = 1<<8
00377 } wolfsentry_lock_flags_t;
00378
00379 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init_thread_context(struct wolfsentry_thread_context
    *thread_context, wolfsentry_thread_flags_t init_thread_flags, void *user_context);
00381 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_alloc_thread_context(struct
    wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context,
    wolfsentry_thread_flags_t init_thread_flags, void *user_context);
00383 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_id(struct wolfsentry_thread_context *thread,
    wolfsentry_thread_id_t *id);
00385 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_user_context(struct
    wolfsentry_thread_context *thread, void **user_context);
00387 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_deadline(struct wolfsentry_thread_context
    *thread, struct timespec *deadline);
00389 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_flags(struct wolfsentry_thread_context
    *thread, wolfsentry_thread_flags_t *thread_flags);
00391 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_destroy_thread_context(struct wolfsentry_thread_context
    *thread_context, wolfsentry_thread_flags_t thread_flags);
00393 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_free_thread_context(struct
    wolfsentry_host_platform_interface *hpi, struct wolfsentry_thread_context **thread_context,
    wolfsentry_thread_flags_t thread_flags);
00395 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_deadline_rel_usecs(WOLFSENTRY_CONTEXT_ARGS_IN, int
    usecs);
00397 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_deadline_abs(WOLFSENTRY_CONTEXT_ARGS_IN, time_t
    epoch_secs, long epoch_nsecs);
00399 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_clear_deadline(WOLFSENTRY_CONTEXT_ARGS_IN);
00401 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_thread_readonly(struct wolfsentry_thread_context
    *thread_context);
00403 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_thread_readwrite(struct wolfsentry_thread_context
    *thread_context);
00406 struct wolfsentry_rwlock;
00407
00422 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_init(struct wolfsentry_host_platform_interface
    *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock *lock,
    wolfsentry_lock_flags_t flags);
00423 WOLFSENTRY_API size_t wolfsentry_lock_size(void);
00438 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_alloc(struct wolfsentry_host_platform_interface
    *hpi, struct wolfsentry_thread_context *thread, struct wolfsentry_rwlock **lock,
    wolfsentry_lock_flags_t flags);
00450 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared(struct wolfsentry_rwlock *lock, struct
    wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00463 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_abstimed(struct wolfsentry_rwlock *lock,
    struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t
    flags);
00476 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared_timed(struct wolfsentry_rwlock *lock,
    struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags);
00488 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex(struct wolfsentry_rwlock *lock, struct
    wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00501 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_abstimed(struct wolfsentry_rwlock *lock,
    struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout, wolfsentry_lock_flags_t
    flags);
00514 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex_timed(struct wolfsentry_rwlock *lock, struct
    wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags);
00526 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex2shared(struct wolfsentry_rwlock *lock,
    struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00538 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex(struct wolfsentry_rwlock *lock,
    struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
```

```
00551 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abstimed(struct wolfsentry_rwlock
      *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout,
      wolfsentry_lock_flags_t flags);
00564 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_timed(struct wolfsentry_rwlock *lock,
      struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t flags);
00580 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_reserve(struct wolfsentry_rwlock
      *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00592 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem(struct wolfsentry_rwlock
      *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00605 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_abstimed(struct
      wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, const struct timespec *abs_timeout,
      wolfsentry_lock_flags_t flags);
00618 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem_timed(struct wolfsentry_rwlock
      *lock, struct wolfsentry_thread_context *thread, wolfsentry_time_t max_wait, wolfsentry_lock_flags_t
      flags);
00630 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abandon(struct wolfsentry_rwlock
      *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00644 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00658 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_mutex(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00673 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_either(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00686 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared2mutex_reservation(struct
      wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00698 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_get_flags(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t *flags);
00710 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_unlock(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00723 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_destroy(struct wolfsentry_rwlock *lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00737 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_free(struct wolfsentry_rwlock **lock, struct
      wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00738
00739 #else /* !WOLFSENTRY_THREADSAFE */
00740
00741 #define WOLFSENTRY_CONTEXT_ARGS_IN struct wolfsentry_context *wolfsentry
00742 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX(ctx) ctx
00743 #define WOLFSENTRY_CONTEXT_ELEMENTS struct wolfsentry_context *wolfsentry
00744 #define WOLFSENTRY_CONTEXT_SET_ELEMENTS(s) (s).wolfsentry = wolfsentry
00745 #define WOLFSENTRY_CONTEXT_GET_ELEMENTS(s) (s).wolfsentry
00746 #define WOLFSENTRY_CONTEXT_ARGS_OUT wolfsentry
00747 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX(ctx) ctx
00748 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX2(x) (x)->wolfsentry
00749 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX3(x, y) (x)->y
00750 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(x, y) x
00751 #define WOLFSENTRY_CONTEXT_ARGS_NOT_USED (void)wolfsentry
00752 #define WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED DO_NOTHING
00753
00754 #define WOLFSENTRY_THREAD_HEADER_DECLS
00755 #define WOLFSENTRY_THREAD_HEADER(flags) DO_NOTHING
00756 #define WOLFSENTRY_THREAD_HEADER_INIT(flags) 0
00757 #define WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(flags) DO_NOTHING
00758 #define WOLFSENTRY_THREAD_HEADER_CHECKED(flags) DO_NOTHING
00759 #define WOLFSENTRY_THREAD_HEADER_CHECK() DO_NOTHING
00760 #define WOLFSENTRY_THREAD_GET_ERROR 0
00761 #define WOLFSENTRY_THREAD_TAILER(flags) 0
00762 #define WOLFSENTRY_THREAD_TAILER_CHECKED(flags) DO_NOTHING
00763
00764 #define wolfsentry_lock_init(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00765 #define wolfsentry_lock_alloc(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00766 #define wolfsentry_lock_shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00767 #define wolfsentry_lock_shared_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00768 #define wolfsentry_lock_mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00769 #define wolfsentry_lock_mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00770 #define wolfsentry_lock_mutex_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00771 #define wolfsentry_lock_mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00772 #define wolfsentry_lock_mutex2shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00773 #define wolfsentry_lock_shared2mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00774 #define wolfsentry_lock_shared2mutex_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00775 #define wolfsentry_lock_shared2mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00776 #define wolfsentry_lock_shared2mutex_reserve(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00777 #define wolfsentry_lock_shared2mutex_redeem(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00778 #define wolfsentry_lock_shared2mutex_redeem_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00779 #define wolfsentry_lock_shared2mutex_redeem_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00780 #define wolfsentry_lock_shared2mutex_abandon(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00781 #define wolfsentry_lock_have_shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00782 #define wolfsentry_lock_have_mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00783 #define wolfsentry_lock_have_either(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00784 #define wolfsentry_lock_have_shared2mutex_reservation(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00785 #define wolfsentry_lock_unlock(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00786 #define wolfsentry_lock_destroy(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00787 #define wolfsentry_lock_free(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00788
00789 #endif /* WOLFSENTRY_THREADSAFE */
00790
00798 typedef enum {
```

```
00799     WOLFSENTRY_OBJECT_TYPE_UNINITED = 0,
00801     WOLFSENTRY_OBJECT_TYPE_TABLE,
00803     WOLFSENTRY_OBJECT_TYPE_ACTION,
00805     WOLFSENTRY_OBJECT_TYPE_EVENT,
00807     WOLFSENTRY_OBJECT_TYPE_ROUTE,
00809     WOLFSENTRY_OBJECT_TYPE_KV,
00811     WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER,
00813     WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME
00815 } wolfsentry_object_type_t;
00816
00824 typedef enum {
00825     WOLFSENTRY_ACTION_FLAG_NONE       = 0U,
00827     WOLFSENTRY_ACTION_FLAG_DISABLED   = 1U « 0U
00829 } wolfsentry_action_flags_t;
00830
00832 typedef enum {
00833     WOLFSENTRY_ACTION_TYPE_NONE = 0,
00835     WOLFSENTRY_ACTION_TYPE_POST = 1,
00837     WOLFSENTRY_ACTION_TYPE_INSERT = 2,
00839     WOLFSENTRY_ACTION_TYPE_MATCH = 3,
00841     WOLFSENTRY_ACTION_TYPE_UPDATE = 4,
00843     WOLFSENTRY_ACTION_TYPE_DELETE = 5,
00845     WOLFSENTRY_ACTION_TYPE_DECISION = 6
00847 } wolfsentry_action_type_t;
00848
00850 typedef enum {
00851     WOLFSENTRY_ACTION_RES_NONE        = 0U,
00853     WOLFSENTRY_ACTION_RES_ACCEPT      = 1U « 0U,
00855     WOLFSENTRY_ACTION_RES_REJECT      = 1U « 1U,
00857     WOLFSENTRY_ACTION_RES_CONNECT     = 1U « 2U,
00859     WOLFSENTRY_ACTION_RES_DISCONNECT  = 1U « 3U,
00861     WOLFSENTRY_ACTION_RES_DEROGATORY  = 1U « 4U,
00863     WOLFSENTRY_ACTION_RES_COMMENDABLE = 1U « 5U,
00866     WOLFSENTRY_ACTION_RES_EXCLUDE_REJECT_ROUTES = WOLFSENTRY_ACTION_RES_DEROGATORY |
      WOLFSENTRY_ACTION_RES_COMMENDABLE, /* internal use -- overload used by wolfsentry_route_lookup_0() */
00868     WOLFSENTRY_ACTION_RES_STOP        = 1U « 6U,
00870     WOLFSENTRY_ACTION_RES_DEALLOCATED = 1U « 7U,
00872     WOLFSENTRY_ACTION_RES_INSERTED    = 1U « 8U,
00874     WOLFSENTRY_ACTION_RES_ERROR       = 1U « 9U,
00876     WOLFSENTRY_ACTION_RES_FALLTHROUGH = 1U « 10U,
00878     WOLFSENTRY_ACTION_RES_UPDATE      = 1U « 11U,
00880     WOLFSENTRY_ACTION_RES_PORT_RESET  = 1U « 12U,
00882     WOLFSENTRY_ACTION_RES_SENDING     = 1U « 13U,
00884     WOLFSENTRY_ACTION_RES_RECEIVED    = 1U « 14U,
00886     WOLFSENTRY_ACTION_RES_BINDING     = 1U « 15U,
00888     WOLFSENTRY_ACTION_RES_LISTENING   = 1U « 16U,
00890     WOLFSENTRY_ACTION_RES_STOPPED_LISTENING = 1U « 17U,
00892     WOLFSENTRY_ACTION_RES_CONNECTING_OUT = 1U « 18U,
00894     WOLFSENTRY_ACTION_RES_CLOSED      = 1U « 19U,
00896     WOLFSENTRY_ACTION_RES_UNREACHABLE = 1U « 20U,
00898     WOLFSENTRY_ACTION_RES_SOCK_ERROR  = 1U « 21U,
00900     WOLFSENTRY_ACTION_RES_CLOSE_WAIT  = 1U « 22U,
00903     WOLFSENTRY_ACTION_RES_RESERVED23  = 1U « 23U,
00905     WOLFSENTRY_ACTION_RES_USER0       = 1U « 24U,
00907     WOLFSENTRY_ACTION_RES_USER1       = 1U « 25U,
00909     WOLFSENTRY_ACTION_RES_USER2       = 1U « 26U,
00911     WOLFSENTRY_ACTION_RES_USER3       = 1U « 27U,
00913     WOLFSENTRY_ACTION_RES_USER4       = 1U « 28U,
00915     WOLFSENTRY_ACTION_RES_USER5       = 1U « 29U,
00917     WOLFSENTRY_ACTION_RES_USER6       = 1U « 30U
00919     /* see macro definition of WOLFSENTRY_ACTION_RES_USER7 below. */
00920
00922 } wolfsentry_action_res_t;
00923
00925 #define WOLFSENTRY_ACTION_RES_USER_BASE WOLFSENTRY_ACTION_RES_USER0
00928 #define WOLFSENTRY_ACTION_RES_USER_SHIFT 24U
00930 #define WOLFSENTRY_ACTION_RES_USER7 (1U « 31U)
00935 struct wolfsentry_table_header;
00936 struct wolfsentry_table_ent_header;
00937 struct wolfsentry_route;
00938 struct wolfsentry_route_table;
00939 struct wolfsentry_event;
00940 struct wolfsentry_event_table;
00941 struct wolfsentry_action;
00942 struct wolfsentry_action_table;
00943 struct wolfsentry_action_list;
00944 struct wolfsentry_action_list_ent;
00945 struct wolfsentry_cursor;
00946
00968 typedef wolfsentry_errcode_t (*wolfsentry_action_callback_t)(
00969     WOLFSENTRY_CONTEXT_ARGS_IN,
00970     const struct wolfsentry_action *action,
00971     void *handler_arg,
00972     void *caller_arg,
00973     const struct wolfsentry_event *trigger_event,
00974     wolfsentry_action_type_t action_type,
00975     const struct wolfsentry_route *trigger_route,
```

```
00976     struct wolfsentry_route_table *route_table,
00977     struct wolfsentry_route *rule_route,
00978     wolfsentry_action_res_t *action_results);
00979
00986 #define WOLFSENTRY_ROUTE_DEFAULT_POLICY_MASK (WOLFSENTRY_ACTION_RES_ACCEPT |
      WOLFSENTRY_ACTION_RES_REJECT | WOLFSENTRY_ACTION_RES_STOP | WOLFSENTRY_ACTION_RES_ERROR)
00990 typedef enum {
00991     WOLFSENTRY_ROUTE_FLAG_NONE                          = 0U,
00993     /* note the wildcard bits need to be at the start, in order of field
00994      * comparison by wolfsentry_route_key_cmp_1(), due to math in
00995      * wolfsentry_route_lookup_0().
00996      */
00997     WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD            = 1U«0U,
00999     WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD       = 1U«1U,
01001     WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD             = 1U«2U,
01003     WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD        = 1U«3U,
01005     WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD        = 1U«4U,
01007     WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD       = 1U«5U,
01009     WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD     = 1U«6U,
01011     WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD      = 1U«7U,
01013     WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD         = 1U«8U,
01015     WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS          = 1U«9U,
01017     WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN                  = 1U«10U,
01019     WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT                 = 1U«11U,
01021     WOLFSENTRY_ROUTE_FLAG_REMOTE_ADDR_BITMASK           = 1U«12U,
01023     WOLFSENTRY_ROUTE_FLAG_LOCAL_ADDR_BITMASK            = 1U«13U,
01026     /* immutable above here. */
01027
01028     /* internal use from here... */
01029     WOLFSENTRY_ROUTE_FLAG_IN_TABLE                      = 1U«14U,
01031     WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE                = 1U«15U,
01033     WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED         = 1U«16U,
01035     WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED         = 1U«17U,
01038     /* ...to here. */
01039
01040     /* mutable below here. */
01041
01042     WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED                  = 1U«20U,
01044     WOLFSENTRY_ROUTE_FLAG_GREENLISTED                   = 1U«21U,
01046     WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_HITS               = 1U«22U,
01048     WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_CURRENT_CONNECTIONS = 1U«23U,
01050     WOLFSENTRY_ROUTE_FLAG_PORT_RESET                    = 1U«24U
01052 } wolfsentry_route_flags_t;
01053
01054 /* note, _PARENT_EVENT_WILDCARD is excluded because it isn't an intrinsic attribute of network/bus
      traffic. */
01055 #define WOLFSENTRY_ROUTE_WILDCARD_FLAGS
      ((wolfsentry_route_flags_t)WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD - 1U)
01058 #define WOLFSENTRY_ROUTE_IMMUTABLE_FLAGS ((wolfsentry_route_flags_t)WOLFSENTRY_ROUTE_FLAG_IN_TABLE -
      1U)
01061 #define WOLFSENTRY_ROUTE_INTERNAL_FLAGS ((wolfsentry_route_flags_t) \
01062                                          (WOLFSENTRY_ROUTE_FLAG_IN_TABLE | \
01063                                           WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE | \
01064                                           WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED | \
01065                                           WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED))
01066
01068 #define WOLFSENTRY_ROUTE_FLAG_TRIGGER_WILDCARD WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD /* xxx
      backward compatibility */
01072 struct wolfsentry_route_endpoint {
01073     wolfsentry_port_t sa_port;
01075     wolfsentry_addr_bits_t addr_len;
01077     byte extra_port_count;
01079     byte interface;
01081 };
01082
01084 struct wolfsentry_route_metadata_exports {
01085     wolfsentry_time_t insert_time;
01087     wolfsentry_time_t last_hit_time;
01089     wolfsentry_time_t last_penaltybox_time;
01091     wolfsentry_time_t purge_after;
01093     uint16_t connection_count;
01095     uint16_t derogatory_count;
01097     uint16_t commendable_count;
01099     wolfsentry_hitcount_t hit_count;
01101 };
01102
01104 struct wolfsentry_route_exports {
01105     const char *parent_event_label;
01107     int parent_event_label_len;
01109     wolfsentry_route_flags_t flags;
01111     wolfsentry_addr_family_t sa_family;
01113     wolfsentry_proto_t sa_proto;
01115     struct wolfsentry_route_endpoint remote;
01117     struct wolfsentry_route_endpoint local;
01119     const byte *remote_address;
01121     const byte *local_address;
01123     const wolfsentry_port_t *remote_extra_ports;
```

```
01125     const wolfsentry_port_t *local_extra_ports;
01127     struct wolfsentry_route_metadata_exports meta;
01129     void *private_data;
01131     size_t private_data_size;
01133 };
01134
01136 struct wolfsentry_sockaddr {
01137     wolfsentry_addr_family_t sa_family;
01139     wolfsentry_proto_t sa_proto;
01141     wolfsentry_port_t sa_port;
01143     wolfsentry_addr_bits_t addr_len;
01145     byte interface;
01147     attr_align_to(4) byte addr[WOLFSENTRY_FLEXIBLE_ARRAY_SIZE];
01149 };
01150
01151 #define WOLFSENTRY_SOCKADDR(n) struct {          \
01152     wolfsentry_addr_family_t sa_family;          \
01153     wolfsentry_proto_t sa_proto;                 \
01154     wolfsentry_port_t sa_port;                   \
01155     wolfsentry_addr_bits_t addr_len;             \
01156     byte interface;                              \
01157     attr_align_to(4) byte addr[WOLFSENTRY_BITS_TO_BYTES(n)];    \
01158 }
01162 typedef enum {
01163     WOLFSENTRY_FORMAT_FLAG_NONE = 0,
01165     WOLFSENTRY_FORMAT_FLAG_ALWAYS_NUMERIC = 1U « 0U
01167 } wolfsentry_format_flags_t;
01168
01176 typedef enum {
01177     WOLFSENTRY_EVENT_FLAG_NONE = 0,
01179     WOLFSENTRY_EVENT_FLAG_IS_PARENT_EVENT = 1U « 0U,
01181     WOLFSENTRY_EVENT_FLAG_IS_SUBEVENT = 1U « 1U
01183 } wolfsentry_event_flags_t;
01184
01186 typedef enum {
01187     WOLFSENTRY_EVENTCONFIG_FLAG_NONE = 0U,
01189     WOLFSENTRY_EVENTCONFIG_FLAG_DEROGATORY_THRESHOLD_IGNORE_COMMENDABLE = 1U « 0U,
01191     WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY = 1U « 1U,
01193     WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_ACTIONS = 1U « 2U
01195 } wolfsentry_eventconfig_flags_t;
01196
01198 struct wolfsentry_eventconfig {
01199     size_t route_private_data_size;
01201     size_t route_private_data_alignment;
01203     uint32_t max_connection_count;
01205     wolfsentry_hitcount_t derogatory_threshold_for_penaltybox;
01207     wolfsentry_time_t penaltybox_duration;
01209     wolfsentry_time_t route_idle_time_for_purge;
01211     wolfsentry_eventconfig_flags_t flags;
01213     wolfsentry_route_flags_t route_flags_to_add_on_insert;
01215     wolfsentry_route_flags_t route_flags_to_clear_on_insert;
01217     wolfsentry_action_res_t action_res_filter_bits_set;
01219     wolfsentry_action_res_t action_res_filter_bits_unset;
01221     wolfsentry_action_res_t action_res_bits_to_add;
01223     wolfsentry_action_res_t action_res_bits_to_clear;
01225 };
01226
01233 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_now_plus_delta(struct wolfsentry_context
      *wolfsentry, wolfsentry_time_t td, wolfsentry_time_t *res);
01236 #ifdef WOLFSENTRY_THREADSAFE
01237 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_to_timespec(struct wolfsentry_context *wolfsentry,
      wolfsentry_time_t t, struct timespec *ts);
01239 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_now_plus_delta_timespec(struct wolfsentry_context
      *wolfsentry, wolfsentry_time_t td, struct timespec *ts);
01241 #endif
01242
01243 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_time(struct wolfsentry_context *wolfsentry,
      wolfsentry_time_t *time_p);
01245 WOLFSENTRY_API wolfsentry_time_t wolfsentry_diff_time(struct wolfsentry_context *wolfsentry,
      wolfsentry_time_t later, wolfsentry_time_t earlier);
01247 WOLFSENTRY_API wolfsentry_time_t wolfsentry_add_time(struct wolfsentry_context *wolfsentry,
      wolfsentry_time_t start_time, wolfsentry_time_t time_interval);
01249 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_to_epoch_time(struct wolfsentry_context *wolfsentry,
      wolfsentry_time_t when, time_t *epoch_secs, long *epoch_nsecs);
01251 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_from_epoch_time(struct wolfsentry_context *wolfsentry,
      time_t epoch_secs, long epoch_nsecs, wolfsentry_time_t *when);
01253 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_interval_to_seconds(struct wolfsentry_context
      *wolfsentry, wolfsentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs);
01255 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_interval_from_seconds(struct wolfsentry_context
      *wolfsentry, time_t howlong_secs, long howlong_nsecs, wolfsentry_time_t *howlong);
01258 WOLFSENTRY_API struct wolfsentry_timecbs *wolfsentry_get_timecbs(struct wolfsentry_context
      *wolfsentry);
01266 typedef wolfsentry_errcode_t (*wolfsentry_make_id_cb_t)(void *context, wolfsentry_ent_id_t *id);
01272 WOLFSENTRY_API void *wolfsentry_malloc(WOLFSENTRY_CONTEXT_ARGS_IN, size_t size);
01274 WOLFSENTRY_API_VOID wolfsentry_free(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr);
01276 WOLFSENTRY_API void *wolfsentry_realloc(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr, size_t size);
01278 WOLFSENTRY_API void *wolfsentry_memalign(WOLFSENTRY_CONTEXT_ARGS_IN, size_t alignment, size_t size);
```

```
01280 WOLFSENTRY_API_VOID wolfsentry_free_aligned(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr);
01282 #if (defined(WOLFSENTRY_MALLOC_BUILTINS) && defined(WOLFSENTRY_MALLOC_DEBUG)) ||
      defined(WOLFSENTRY_FOR_DOXYGEN)
01283 WOLFSENTRY_API int _wolfsentry_get_n_mallocs(void);
01285 #endif
01286
01287 WOLFSENTRY_API struct wolfsentry_allocator *wolfsentry_get_allocator(struct wolfsentry_context
      *wolfsentry);
01292 #if defined(WOLFSENTRY_PROTOCOL_NAMES) || !defined(WOLFSENTRY_NO_JSON)
01296 WOLFSENTRY_API const char *wolfsentry_action_res_assoc_by_flag(wolfsentry_action_res_t res, unsigned
      int bit);
01298 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_res_assoc_by_name(const char *bit_name, int
      bit_name_len, wolfsentry_action_res_t *res);
01301 #endif
01302
01307 WOLFSENTRY_API struct wolfsentry_host_platform_interface *wolfsentry_get_hpi(struct wolfsentry_context
      *wolfsentry);
01310 typedef void (*wolfsentry_cleanup_callback_t)(
01311     WOLFSENTRY_CONTEXT_ARGS_IN,
01312     void *cleanup_arg);
01315 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_cleanup_push(
01316     WOLFSENTRY_CONTEXT_ARGS_IN,
01317     wolfsentry_cleanup_callback_t handler,
01318     void *arg);
01321 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_cleanup_pop(
01322     WOLFSENTRY_CONTEXT_ARGS_IN,
01323     int execute_p);
01326 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_cleanup_all(
01327     WOLFSENTRY_CONTEXT_ARGS_IN);
01336 /* must return _BUFFER_TOO_SMALL and set *addr_internal_bits to an
01337  * accurate value when supplied with a NULL output buf ptr.
01338  * whenever _BUFFER_TOO_SMALL is returned, *addr_*_bits must be set to an
01339  * accurate value.
01340  */
01341 typedef wolfsentry_errcode_t (*wolfsentry_addr_family_parser_t)(
01342     WOLFSENTRY_CONTEXT_ARGS_IN,
01343     const char *addr_text,
01344     int addr_text_len,
01345     byte *addr_internal,
01346     wolfsentry_addr_bits_t *addr_internal_bits);
01349 typedef wolfsentry_errcode_t (*wolfsentry_addr_family_formatter_t)(
01350     WOLFSENTRY_CONTEXT_ARGS_IN,
01351     const byte *addr_internal,
01352     unsigned int addr_internal_bits,
01353     char *addr_text,
01354     int *addr_text_len);
01357 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_handler_install(
01358     WOLFSENTRY_CONTEXT_ARGS_IN,
01359     wolfsentry_addr_family_t family_bynumber,
01360     const char *family_byname, /* if defined(WOLFSENTRY_PROTOCOL_NAMES), must not be NULL, else
      ignored. */
01361     int family_byname_len,
01362     wolfsentry_addr_family_parser_t parser,
01363     wolfsentry_addr_family_formatter_t formatter,
01364     int max_addr_bits);
01367 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_get_parser(
01368     WOLFSENTRY_CONTEXT_ARGS_IN,
01369     wolfsentry_addr_family_t family,
01370     wolfsentry_addr_family_parser_t *parser);
01373 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_get_formatter(
01374     WOLFSENTRY_CONTEXT_ARGS_IN,
01375     wolfsentry_addr_family_t family,
01376     wolfsentry_addr_family_formatter_t *formatter);
01379 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_handler_remove_bynumber(
01380     WOLFSENTRY_CONTEXT_ARGS_IN,
01381     wolfsentry_addr_family_t family_bynumber,
01382     wolfsentry_action_res_t *action_results);
01385 struct wolfsentry_addr_family_bynumber;
01386
01387 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_drop_reference(
01388     WOLFSENTRY_CONTEXT_ARGS_IN,
01389     struct wolfsentry_addr_family_bynumber *family_bynumber,
01390     wolfsentry_action_res_t *action_results);
01393 #ifdef WOLFSENTRY_PROTOCOL_NAMES
01394
01395 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_handler_remove_byname(
01396     WOLFSENTRY_CONTEXT_ARGS_IN,
01397     const char *family_byname,
01398     int family_byname_len,
01399     wolfsentry_action_res_t *action_results);
01402 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_pton(
01403     WOLFSENTRY_CONTEXT_ARGS_IN,
01404     const char *family_name,
01405     int family_name_len,
01406     wolfsentry_addr_family_t *family_number);
01409 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_ntop(
01410     WOLFSENTRY_CONTEXT_ARGS_IN,
```

```
01411     wolfsentry_addr_family_t family,
01412     struct wolfsentry_addr_family_bynumber **addr_family,
01413     const char **family_name);
01416 #endif /* WOLFSENTRY_PROTOCOL_NAMES */
01417
01418 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_max_addr_bits(
01419     WOLFSENTRY_CONTEXT_ARGS_IN,
01420     wolfsentry_addr_family_t family,
01421     wolfsentry_addr_bits_t *bits);
01439 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_init(
01440     struct wolfsentry_context *wolfsentry,
01441     struct wolfsentry_eventconfig *config);
01449 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_check(
01450     const struct wolfsentry_eventconfig *config);
01451
01457 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init_ex(
01458     struct wolfsentry_build_settings caller_build_settings,
01459     WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi),
01460     const struct wolfsentry_eventconfig *config,
01461     struct wolfsentry_context **wolfsentry,
01462     wolfsentry_init_flags_t flags);
01477 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init(
01478     struct wolfsentry_build_settings caller_build_settings,
01479     WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi),
01480     const struct wolfsentry_eventconfig *config,
01481     struct wolfsentry_context **wolfsentry);
01489 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_get(
01490     WOLFSENTRY_CONTEXT_ARGS_IN,
01491     struct wolfsentry_eventconfig *config);
01501 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_update(
01502     WOLFSENTRY_CONTEXT_ARGS_IN,
01503     const struct wolfsentry_eventconfig *config);
01511 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_flush(WOLFSENTRY_CONTEXT_ARGS_IN);
01521 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_free(WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct
      wolfsentry_context **wolfsentry));
01530 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct
      wolfsentry_context **wolfsentry));
01538 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_inhibit_actions(WOLFSENTRY_CONTEXT_ARGS_IN);
01546 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_enable_actions(WOLFSENTRY_CONTEXT_ARGS_IN);
01547
01549 typedef enum {
01550     WOLFSENTRY_CLONE_FLAG_NONE = 0U,
01552     WOLFSENTRY_CLONE_FLAG_AS_AT_CREATION = 1U << 0U,
01554     WOLFSENTRY_CLONE_FLAG_NO_ROUTES = 2U << 0U
01556 } wolfsentry_clone_flags_t;
01567 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_clone(WOLFSENTRY_CONTEXT_ARGS_IN, struct
      wolfsentry_context **clone, wolfsentry_clone_flags_t flags);
01577 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_exchange(WOLFSENTRY_CONTEXT_ARGS_IN, struct
      wolfsentry_context *wolfsentry2);
01578
01585 #ifdef WOLFSENTRY_THREADSAFE
01586
01587 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex(
01588     WOLFSENTRY_CONTEXT_ARGS_IN);
01590 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_abstimed(
01591     WOLFSENTRY_CONTEXT_ARGS_IN,
01592     const struct timespec *abs_timeout);
01594 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_abstimed_ex(
01595     WOLFSENTRY_CONTEXT_ARGS_IN,
01596     const struct timespec *abs_timeout,
01597     wolfsentry_lock_flags_t flags);
01599 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_timed(
01600     WOLFSENTRY_CONTEXT_ARGS_IN,
01601     wolfsentry_time_t max_wait);
01603 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_timed_ex(
01604     WOLFSENTRY_CONTEXT_ARGS_IN,
01605     wolfsentry_time_t max_wait,
01606     wolfsentry_lock_flags_t flags);
01608 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared(
01609     WOLFSENTRY_CONTEXT_ARGS_IN);
01611 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_abstimed(
01612     WOLFSENTRY_CONTEXT_ARGS_IN,
01613     const struct timespec *abs_timeout);
01615 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_with_reservation_abstimed(
01616     WOLFSENTRY_CONTEXT_ARGS_IN,
01617     const struct timespec *abs_timeout);
01619 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_timed(
01620     WOLFSENTRY_CONTEXT_ARGS_IN,
01621     wolfsentry_time_t max_wait);
01623 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_with_reservation_timed(
01624     WOLFSENTRY_CONTEXT_ARGS_IN,
01625     wolfsentry_time_t max_wait);
01627 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_unlock(
01628     WOLFSENTRY_CONTEXT_ARGS_IN);
01630 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_unlock_and_abandon_reservation(
01631     WOLFSENTRY_CONTEXT_ARGS_IN);
01634 #else /* !WOLFSENTRY_THREADSAFE */
```

```
01635
01636 #define wolfsentry_context_lock_mutex(x) WOLFSENTRY_ERROR_ENCODE(OK)
01637 #define wolfsentry_context_lock_mutex_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01638 #define wolfsentry_context_lock_mutex_timed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01639 #define wolfsentry_context_lock_shared(x) WOLFSENTRY_ERROR_ENCODE(OK)
01640 #define wolfsentry_context_lock_shared_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01641 #define wolfsentry_context_lock_shared_with_reservation_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01642 #define wolfsentry_context_lock_shared_timed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01643 #define wolfsentry_context_unlock(x) WOLFSENTRY_ERROR_ENCODE(OK)
01644
01645 #endif /* WOLFSENTRY_THREADSAFE */
01646
01649 #define WOLFSENTRY_LENGTH_NULL_TERMINATED (-1)
01663 WOLFSENTRY_API wolfsentry_object_type_t wolfsentry_get_object_type(const void *object);
01664
01672 WOLFSENTRY_API wolfsentry_ent_id_t wolfsentry_get_object_id(const void *object);
01673
01674 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_table_ent_get_by_id(
01675     WOLFSENTRY_CONTEXT_ARGS_IN,
01676     wolfsentry_ent_id_t id,
01677     struct wolfsentry_table_ent_header **ent);
01680 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_object_checkout(WOLFSENTRY_CONTEXT_ARGS_IN, void
      *object);
01683 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_object_release(WOLFSENTRY_CONTEXT_ARGS_IN, void
      *object, wolfsentry_action_res_t *action_results);
01693 WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_inserts(struct wolfsentry_table_header
      *table);
01694
01702 WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_deletes(struct wolfsentry_table_header
      *table);
01703
01710 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_check_flags_sensical(
01711     wolfsentry_route_flags_t flags);
01714 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_into_table(
01715     WOLFSENTRY_CONTEXT_ARGS_IN,
01716     struct wolfsentry_route_table *route_table,
01717     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01718     const struct wolfsentry_sockaddr *remote,
01719     const struct wolfsentry_sockaddr *local,
01720     wolfsentry_route_flags_t flags,
01721     const char *event_label,
01722     int event_label_len,
01723     wolfsentry_ent_id_t *id,
01724     wolfsentry_action_res_t *action_results);
01727 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_into_table(
01728     WOLFSENTRY_CONTEXT_ARGS_IN,
01729     struct wolfsentry_route_table *route_table,
01730     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01731     const struct wolfsentry_route_exports *route_exports,
01732     wolfsentry_ent_id_t *id,
01733     wolfsentry_action_res_t *action_results);
01752 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert(
01753     WOLFSENTRY_CONTEXT_ARGS_IN,
01754     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01755     const struct wolfsentry_sockaddr *remote,
01756     const struct wolfsentry_sockaddr *local,
01757     wolfsentry_route_flags_t flags,
01758     const char *event_label,
01759     int event_label_len,
01760     wolfsentry_ent_id_t *id,
01761     wolfsentry_action_res_t *action_results);
01762
01763 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports(
01764     WOLFSENTRY_CONTEXT_ARGS_IN,
01765     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01766     const struct wolfsentry_route_exports *route_exports,
01767     wolfsentry_ent_id_t *id,
01768     wolfsentry_action_res_t *action_results);
01771 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_into_table_and_check_out(
01772     WOLFSENTRY_CONTEXT_ARGS_IN,
01773     struct wolfsentry_route_table *route_table,
01774     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01775     const struct wolfsentry_sockaddr *remote,
01776     const struct wolfsentry_sockaddr *local,
01777     wolfsentry_route_flags_t flags,
01778     const char *event_label,
01779     int event_label_len,
01780     struct wolfsentry_route **route,
01781     wolfsentry_action_res_t *action_results);
01784 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_into_table_and_check_out(
01785     WOLFSENTRY_CONTEXT_ARGS_IN,
01786     struct wolfsentry_route_table *route_table,
01787     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01788     const struct wolfsentry_route_exports *route_exports,
01789     struct wolfsentry_route **route,
01790     wolfsentry_action_res_t *action_results);
01793 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_and_check_out(
```

```
01794     WOLFSENTRY_CONTEXT_ARGS_IN,
01795     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01796     const struct wolfsentry_sockaddr *remote,
01797     const struct wolfsentry_sockaddr *local,
01798     wolfsentry_route_flags_t flags,
01799     const char *event_label,
01800     int event_label_len,
01801     struct wolfsentry_route **route,
01802     wolfsentry_action_res_t *action_results);
01805 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_and_check_out(
01806     WOLFSENTRY_CONTEXT_ARGS_IN,
01807     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01808     const struct wolfsentry_route_exports *route_exports,
01809     struct wolfsentry_route **route,
01810     wolfsentry_action_res_t *action_results);
01813 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_from_table(
01814     WOLFSENTRY_CONTEXT_ARGS_IN,
01815     struct wolfsentry_route_table *route_table,
01816     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01817     const struct wolfsentry_sockaddr *remote,
01818     const struct wolfsentry_sockaddr *local,
01819     wolfsentry_route_flags_t flags,
01820     const char *event_label,
01821     int event_label_len,
01822     wolfsentry_action_res_t *action_results,
01823     int *n_deleted);
01842 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete(
01843     WOLFSENTRY_CONTEXT_ARGS_IN,
01844     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01845     const struct wolfsentry_sockaddr *remote,
01846     const struct wolfsentry_sockaddr *local,
01847     wolfsentry_route_flags_t flags,
01848     const char *trigger_label,
01849     int trigger_label_len,
01850     wolfsentry_action_res_t *action_results,
01851     int *n_deleted);
01852
01866 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_by_id(
01867     WOLFSENTRY_CONTEXT_ARGS_IN,
01868     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01869     wolfsentry_ent_id_t id,
01870     const char *trigger_label,
01871     int trigger_label_len,
01872     wolfsentry_action_res_t *action_results);
01873
01885 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_main_table(
01886     WOLFSENTRY_CONTEXT_ARGS_IN,
01887     struct wolfsentry_route_table **table);
01888
01901 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_start(
01902     WOLFSENTRY_CONTEXT_ARGS_IN,
01903     const struct wolfsentry_route_table *table,
01904     struct wolfsentry_cursor **cursor);
01905
01914 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_head(
01915     const struct wolfsentry_route_table *table,
01916     struct wolfsentry_cursor *cursor);
01917
01926 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_tail(
01927     const struct wolfsentry_route_table *table,
01928     struct wolfsentry_cursor *cursor);
01929
01939 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_current(
01940     const struct wolfsentry_route_table *table,
01941     struct wolfsentry_cursor *cursor,
01942     struct wolfsentry_route **route);
01943
01953 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_prev(
01954     const struct wolfsentry_route_table *table,
01955     struct wolfsentry_cursor *cursor,
01956     struct wolfsentry_route **route);
01957
01967 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_next(
01968     const struct wolfsentry_route_table *table,
01969     struct wolfsentry_cursor *cursor,
01970     struct wolfsentry_route **route);
01971
01984 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_end(
01985     WOLFSENTRY_CONTEXT_ARGS_IN,
01986     const struct wolfsentry_route_table *table,
01987     struct wolfsentry_cursor **cursor);
01988
01999 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_set(
02000     WOLFSENTRY_CONTEXT_ARGS_IN,
02001     struct wolfsentry_route_table *table,
02002     wolfsentry_action_res_t default_policy);
02003
```

```
02004 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_default_policy_set(
02005     WOLFSENTRY_CONTEXT_ARGS_IN,
02006     wolfsentry_action_res_t default_policy);
02022 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_get(
02023     WOLFSENTRY_CONTEXT_ARGS_IN,
02024     struct wolfsentry_route_table *table,
02025     wolfsentry_action_res_t *default_policy);
02026
02027 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_default_policy_get(
02028     WOLFSENTRY_CONTEXT_ARGS_IN,
02029     wolfsentry_action_res_t *default_policy);
02049 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_reference(
02050     WOLFSENTRY_CONTEXT_ARGS_IN,
02051     const struct wolfsentry_route_table *table,
02052     const struct wolfsentry_sockaddr *remote,
02053     const struct wolfsentry_sockaddr *local,
02054     wolfsentry_route_flags_t flags,
02055     const char *event_label,
02056     int event_label_len,
02057     int exact_p,
02058     wolfsentry_route_flags_t *inexact_matches,
02059     struct wolfsentry_route **route);
02060
02071 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_drop_reference(
02072     WOLFSENTRY_CONTEXT_ARGS_IN,
02073     struct wolfsentry_route *route,
02074     wolfsentry_action_res_t *action_results);
02075
02076 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_clear_default_event(
02077     WOLFSENTRY_CONTEXT_ARGS_IN,
02078     struct wolfsentry_route_table *table);
02081 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_set_default_event(
02082     WOLFSENTRY_CONTEXT_ARGS_IN,
02083     struct wolfsentry_route_table *table,
02084     const char *event_label,
02085     int event_label_len);
02088 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_get_default_event(
02089     WOLFSENTRY_CONTEXT_ARGS_IN,
02090     struct wolfsentry_route_table *table,
02091     char *event_label,
02092     int *event_label_len);
02103 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_fallthrough_route_get(
02104     WOLFSENTRY_CONTEXT_ARGS_IN,
02105     struct wolfsentry_route_table *route_table,
02106     const struct wolfsentry_route **fallthrough_route);
02107
02116 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs(
02117     const struct wolfsentry_route *route,
02118     wolfsentry_addr_family_t *af,
02119     wolfsentry_addr_bits_t *local_addr_len,
02120     const byte **local_addr,
02121     wolfsentry_addr_bits_t *remote_addr_len,
02122     const byte **remote_addr);
02123
02139 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_export(
02140     WOLFSENTRY_CONTEXT_ARGS_IN,
02141     const struct wolfsentry_route *route,
02142     struct wolfsentry_route_exports *route_exports);
02143
02144 /* returned wolfsentry_event remains valid only as long as the wolfsentry lock
02145  * is held (shared or exclusive), unless the route was obtained via
02146  * wolfsentry_route_get_reference(), in which case it's valid until
02147  * wolfsentry_route_drop_reference()..
02148  */
02158 WOLFSENTRY_API const struct wolfsentry_event *wolfsentry_route_parent_event(const struct
      wolfsentry_route *route);
02159
02160 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_table(
02161     WOLFSENTRY_CONTEXT_ARGS_IN,
02162     struct wolfsentry_route_table *route_table,
02163     const struct wolfsentry_sockaddr *remote,
02164     const struct wolfsentry_sockaddr *local,
02165     wolfsentry_route_flags_t flags,
02166     const char *event_label,
02167     int event_label_len,
02168     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02169     wolfsentry_ent_id_t *id,
02170     wolfsentry_route_flags_t *inexact_matches,
02171     wolfsentry_action_res_t *action_results);
02191 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch(
02192     WOLFSENTRY_CONTEXT_ARGS_IN,
02193     const struct wolfsentry_sockaddr *remote,
02194     const struct wolfsentry_sockaddr *local,
02195     wolfsentry_route_flags_t flags,
02196     const char *event_label,
02197     int event_label_len,
02198     void *caller_arg, /* passed to action callback(s). */
```

```
02199     wolfsentry_ent_id_t *id,
02200     wolfsentry_route_flags_t *inexact_matches,
02201     wolfsentry_action_res_t *action_results);
02202
02203 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_table_with_inited_result(
02204     WOLFSENTRY_CONTEXT_ARGS_IN,
02205     struct wolfsentry_route_table *route_table,
02206     const struct wolfsentry_sockaddr *remote,
02207     const struct wolfsentry_sockaddr *local,
02208     wolfsentry_route_flags_t flags,
02209     const char *event_label,
02210     int event_label_len,
02211     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02212     wolfsentry_ent_id_t *id,
02213     wolfsentry_route_flags_t *inexact_matches,
02214     wolfsentry_action_res_t *action_results);
02217 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_inited_result(
02218     WOLFSENTRY_CONTEXT_ARGS_IN,
02219     const struct wolfsentry_sockaddr *remote,
02220     const struct wolfsentry_sockaddr *local,
02221     wolfsentry_route_flags_t flags,
02222     const char *event_label,
02223     int event_label_len,
02224     void *caller_arg, /* passed to action callback(s). */
02225     wolfsentry_ent_id_t *id,
02226     wolfsentry_route_flags_t *inexact_matches,
02227     wolfsentry_action_res_t *action_results);
02230 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_id(
02231     WOLFSENTRY_CONTEXT_ARGS_IN,
02232     wolfsentry_ent_id_t id,
02233     const char *event_label,
02234     int event_label_len,
02235     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02236     wolfsentry_action_res_t *action_results
02237     );
02240 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_id_with_inited_result(
02241     WOLFSENTRY_CONTEXT_ARGS_IN,
02242     wolfsentry_ent_id_t id,
02243     const char *event_label,
02244     int event_label_len,
02245     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02246     wolfsentry_action_res_t *action_results
02247     );
02250 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_route(
02251     WOLFSENTRY_CONTEXT_ARGS_IN,
02252     struct wolfsentry_route *route,
02253     const char *event_label,
02254     int event_label_len,
02255     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02256     wolfsentry_action_res_t *action_results
02257     );
02260 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_route_with_inited_result(
02261     WOLFSENTRY_CONTEXT_ARGS_IN,
02262     struct wolfsentry_route *route,
02263     const char *event_label,
02264     int event_label_len,
02265     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02266     wolfsentry_action_res_t *action_results
02267     );
02270 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_routes_get(
02271     WOLFSENTRY_CONTEXT_ARGS_IN,
02272     struct wolfsentry_route_table *table,
02273     wolfsentry_hitcount_t *max_purgeable_routes);
02276 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_routes_set(
02277     WOLFSENTRY_CONTEXT_ARGS_IN,
02278     struct wolfsentry_route_table *table,
02279     wolfsentry_hitcount_t max_purgeable_routes);
02282 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_idle_time_get(
02283     WOLFSENTRY_CONTEXT_ARGS_IN,
02284     struct wolfsentry_route_table *table,
02285     wolfsentry_time_t *max_purgeable_idle_time);
02288 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_idle_time_set(
02289     WOLFSENTRY_CONTEXT_ARGS_IN,
02290     struct wolfsentry_route_table *table,
02291     wolfsentry_time_t max_purgeable_idle_time);
02294 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_purge_time_set(
02295     WOLFSENTRY_CONTEXT_ARGS_IN,
02296     struct wolfsentry_route *route,
02297     wolfsentry_time_t purge_after);
02310 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge(
02311     WOLFSENTRY_CONTEXT_ARGS_IN,
02312     struct wolfsentry_route_table *table,
02313     wolfsentry_action_res_t *action_results);
02314
02315 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge_one(
02316     WOLFSENTRY_CONTEXT_ARGS_IN,
02317     struct wolfsentry_route_table *table,
```

```
02318     wolfsentry_action_res_t *action_results);
02321 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge_one_opportunistically(
02322     WOLFSENTRY_CONTEXT_ARGS_IN,
02323     struct wolfsentry_route_table *table,
02324     wolfsentry_action_res_t *action_results);
02337 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flush_table(
02338     WOLFSENTRY_CONTEXT_ARGS_IN,
02339     struct wolfsentry_route_table *table,
02340     wolfsentry_action_res_t *action_results);
02341
02350 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_clear_insert_action_status(
02351     WOLFSENTRY_CONTEXT_ARGS_IN,
02352     wolfsentry_action_res_t *action_results);
02353
02362 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_insert_actions(
02363     WOLFSENTRY_CONTEXT_ARGS_IN,
02364     wolfsentry_action_res_t *action_results);
02365
02377 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_private_data(
02378     WOLFSENTRY_CONTEXT_ARGS_IN,
02379     struct wolfsentry_route *route,
02380     void **private_data,
02381     size_t *private_data_size);
02382
02391 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_flags(
02392     const struct wolfsentry_route *route,
02393     wolfsentry_route_flags_t *flags);
02394
02403 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_metadata(
02404     const struct wolfsentry_route *route,
02405     struct wolfsentry_route_metadata_exports *metadata);
02406
02407 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_metadata_exports(
02408     struct wolfsentry_route_exports *route_exports);
02425 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_update_flags(
02426     WOLFSENTRY_CONTEXT_ARGS_IN,
02427     struct wolfsentry_route *route,
02428     wolfsentry_route_flags_t flags_to_set,
02429     wolfsentry_route_flags_t flags_to_clear,
02430     wolfsentry_route_flags_t *flags_before,
02431     wolfsentry_route_flags_t *flags_after,
02432     wolfsentry_action_res_t *action_results);
02433
02434 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_increment_derogatory_count(
02435     WOLFSENTRY_CONTEXT_ARGS_IN,
02436     struct wolfsentry_route *route,
02437     int count_to_add,
02438     int *new_derogatory_count_ptr);
02441 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_increment_commendable_count(
02442     WOLFSENTRY_CONTEXT_ARGS_IN,
02443     struct wolfsentry_route *route,
02444     int count_to_add,
02445     int *new_commendable_count);
02448 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_derogatory_count(
02449     WOLFSENTRY_CONTEXT_ARGS_IN,
02450     struct wolfsentry_route *route,
02451     int *old_derogatory_count_ptr);
02454 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_commendable_count(
02455     WOLFSENTRY_CONTEXT_ARGS_IN,
02456     struct wolfsentry_route *route,
02457     int *old_commendable_count_ptr);
02468 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_set_wildcard(
02469     struct wolfsentry_route *route,
02470     wolfsentry_route_flags_t wildcards_to_set);
02471
02472 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_format_address(
02473     WOLFSENTRY_CONTEXT_ARGS_IN,
02474     wolfsentry_addr_family_t sa_family,
02475     const byte *addr,
02476     unsigned int addr_bits,
02477     char *buf,
02478     int *buflen);
02481 #if defined(WOLFSENTRY_PROTOCOL_NAMES) || defined(WOLFSENTRY_JSON_DUMP_UTILS) ||
      !defined(WOLFSENTRY_NO_JSON)
02482
02483 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flag_assoc_by_flag(
02484     wolfsentry_route_flags_t flag,
02485     const char **name);
02488 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flag_assoc_by_name(
02489     const char *name,
02490     int len,
02491     wolfsentry_route_flags_t *flag);
02494 #endif /* WOLFSENTRY_PROTOCOL_NAMES || WOLFSENTRY_JSON_DUMP_UTILS || !WOLFSENTRY_NO_JSON */
02495
02496 #if !defined(WOLFSENTRY_NO_JSON) || defined(WOLFSENTRY_JSON_DUMP_UTILS)
02497
02498 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_format_json(
```

```
02499     WOLFSENTRY_CONTEXT_ARGS_IN,
02500     const struct wolfsentry_route *r,
02501     unsigned char **json_out,
02502     size_t *json_out_len,
02503     wolfsentry_format_flags_t flags);
02506 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_start(
02507     WOLFSENTRY_CONTEXT_ARGS_IN,
02508     const struct wolfsentry_route_table *table,
02509     struct wolfsentry_cursor **cursor,
02510     unsigned char **json_out,
02511     size_t *json_out_len,
02512     wolfsentry_format_flags_t flags);
02515 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_next(
02516     WOLFSENTRY_CONTEXT_ARGS_IN,
02517     const struct wolfsentry_route_table *table,
02518     struct wolfsentry_cursor *cursor,
02519     unsigned char **json_out,
02520     size_t *json_out_len,
02521     wolfsentry_format_flags_t flags);
02524 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_end(
02525     WOLFSENTRY_CONTEXT_ARGS_IN,
02526     const struct wolfsentry_route_table *table,
02527     struct wolfsentry_cursor **cursor,
02528     unsigned char **json_out,
02529     size_t *json_out_len,
02530     wolfsentry_format_flags_t flags);
02533 #endif /* !WOLFSENTRY_NO_JSON || WOLFSENTRY_JSON_DUMP_UTILS */
02534
02535 #ifndef WOLFSENTRY_NO_STDIO_STREAMS
02536 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render_flags(wolfsentry_route_flags_t flags, FILE
      *f);
02549 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render(WOLFSENTRY_CONTEXT_ARGS_IN, const struct
      wolfsentry_route *r, FILE *f);
02560 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_exports_render(WOLFSENTRY_CONTEXT_ARGS_IN, const
      struct wolfsentry_route_exports *r, FILE *f);
02561 #endif
02562
02583 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_insert(
02584     WOLFSENTRY_CONTEXT_ARGS_IN,
02585     const char *label,
02586     int label_len,
02587     wolfsentry_action_flags_t flags,
02588     wolfsentry_action_callback_t handler,
02589     void *handler_arg,
02590     wolfsentry_ent_id_t *id);
02591
02603 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_delete(
02604     WOLFSENTRY_CONTEXT_ARGS_IN,
02605     const char *label,
02606     int label_len,
02607     wolfsentry_action_res_t *action_results);
02608
02616 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_flush_all(WOLFSENTRY_CONTEXT_ARGS_IN);
02617
02629 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_reference(
02630     WOLFSENTRY_CONTEXT_ARGS_IN,
02631     const char *label,
02632     int label_len,
02633     struct wolfsentry_action **action);
02634
02645 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_drop_reference(
02646     WOLFSENTRY_CONTEXT_ARGS_IN,
02647     struct wolfsentry_action *action,
02648     wolfsentry_action_res_t *action_results);
02649
02657 WOLFSENTRY_API const char *wolfsentry_action_get_label(const struct wolfsentry_action *action);
02658
02667 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_flags(
02668     struct wolfsentry_action *action,
02669     wolfsentry_action_flags_t *flags);
02670
02682 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_update_flags(
02683     struct wolfsentry_action *action,
02684     wolfsentry_action_flags_t flags_to_set,
02685     wolfsentry_action_flags_t flags_to_clear,
02686     wolfsentry_action_flags_t *flags_before,
02687     wolfsentry_action_flags_t *flags_after);
02688
02709 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_insert(
02710     WOLFSENTRY_CONTEXT_ARGS_IN,
02711     const char *label,
02712     int label_len,
02713     wolfsentry_priority_t priority,
02714     const struct wolfsentry_eventconfig *config,
02715     wolfsentry_event_flags_t flags,
02716     wolfsentry_ent_id_t *id);
02717
```

```
02727 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_delete(
02728     WOLFSENTRY_CONTEXT_ARGS_IN,
02729     const char *label,
02730     int label_len,
02731     wolfsentry_action_res_t *action_results);
02732
02740 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_flush_all(WOLFSENTRY_CONTEXT_ARGS_IN);
02741
02749 WOLFSENTRY_API const char *wolfsentry_event_get_label(const struct wolfsentry_event *event);
02750
02758 WOLFSENTRY_API wolfsentry_event_flags_t wolfsentry_event_get_flags(const struct wolfsentry_event
      *event);
02759
02771 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_config(
02772     WOLFSENTRY_CONTEXT_ARGS_IN,
02773     const char *label,
02774     int label_len,
02775     struct wolfsentry_eventconfig *config);
02776
02788 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_update_config(
02789     WOLFSENTRY_CONTEXT_ARGS_IN,
02790     const char *label,
02791     int label_len,
02792     const struct wolfsentry_eventconfig *config);
02793
02805 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_get_reference(
02806     WOLFSENTRY_CONTEXT_ARGS_IN,
02807     const char *label,
02808     int label_len,
02809     struct wolfsentry_event **event);
02810
02821 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_drop_reference(
02822     WOLFSENTRY_CONTEXT_ARGS_IN,
02823     struct wolfsentry_event *event,
02824     wolfsentry_action_res_t *action_results);
02825
02839 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_prepend(
02840     WOLFSENTRY_CONTEXT_ARGS_IN,
02841     const char *event_label,
02842     int event_label_len,
02843     wolfsentry_action_type_t which_action_list,
02844     const char *action_label,
02845     int action_label_len);
02846
02860 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_append(
02861     WOLFSENTRY_CONTEXT_ARGS_IN,
02862     const char *event_label,
02863     int event_label_len,
02864     wolfsentry_action_type_t which_action_list,
02865     const char *action_label,
02866     int action_label_len);
02867
02883 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_insert_after(
02884     WOLFSENTRY_CONTEXT_ARGS_IN,
02885     const char *event_label,
02886     int event_label_len,
02887     wolfsentry_action_type_t which_action_list,
02888     const char *action_label,
02889     int action_label_len,
02890     const char *point_action_label,
02891     int point_action_label_len);
02892
02906 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_delete(
02907     WOLFSENTRY_CONTEXT_ARGS_IN,
02908     const char *event_label,
02909     int event_label_len,
02910     wolfsentry_action_type_t which_action_list,
02911     const char *action_label,
02912     int action_label_len);
02913
02926 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_set_aux_event(
02927     WOLFSENTRY_CONTEXT_ARGS_IN,
02928     const char *event_label,
02929     int event_label_len,
02930     const char *aux_event_label,
02931     int aux_event_label_len);
02932
02933 WOLFSENTRY_API const struct wolfsentry_event *wolfsentry_event_get_aux_event(
02934     const struct wolfsentry_event *event);
02951 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_start(
02952     WOLFSENTRY_CONTEXT_ARGS_IN,
02953     const char *event_label,
02954     int event_label_len,
02955     wolfsentry_action_type_t which_action_list,
02956     struct wolfsentry_action_list_ent **cursor);
02957
02971 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_next(
```

```
02972     WOLFSENTRY_CONTEXT_ARGS_IN,
02973     struct wolfsentry_action_list_ent **cursor,
02974     const char **action_label,
02975     int *action_label_len);
02976
02988 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_done(
02989     WOLFSENTRY_CONTEXT_ARGS_IN,
02990     struct wolfsentry_action_list_ent **cursor);
02991
02994 #ifdef WOLFSENTRY_HAVE_JSON_DOM
02995 #include <wolfsentry/centijson_dom.h>
02996 #endif
02997
03003 typedef enum {
03004     WOLFSENTRY_KV_NONE = 0,
03005     WOLFSENTRY_KV_NULL,
03006     WOLFSENTRY_KV_TRUE,
03007     WOLFSENTRY_KV_FALSE,
03008     WOLFSENTRY_KV_UINT,
03009     WOLFSENTRY_KV_SINT,
03010     WOLFSENTRY_KV_FLOAT,
03011     WOLFSENTRY_KV_STRING,
03012     WOLFSENTRY_KV_BYTES,
03013     WOLFSENTRY_KV_JSON,
03014     WOLFSENTRY_KV_FLAG_READONLY = 1«30
03015 } wolfsentry_kv_type_t;
03016
03017 #define WOLFSENTRY_KV_FLAG_MASK WOLFSENTRY_KV_FLAG_READONLY
03021 struct wolfsentry_kv_pair {
03022     int key_len;
03024     wolfsentry_kv_type_t v_type;
03026     union {
03027         uint64_t v_uint;
03029         int64_t v_sint;
03031         double v_float;
03033         size_t string_len;
03035         size_t bytes_len;
03037 #ifdef WOLFSENTRY_HAVE_JSON_DOM
03038         JSON_VALUE v_json;
03039  /* 16 bytes */
03040 #endif
03041     } a;
03042     byte b[WOLFSENTRY_FLEXIBLE_ARRAY_SIZE];
03047 };
03048
03049 #define WOLFSENTRY_KV_KEY_LEN(kv) ((kv)->key_len)
03051 #define WOLFSENTRY_KV_KEY(kv) ((char *)((kv)->b))
03053 #define WOLFSENTRY_KV_TYPE(kv) ((uint32_t)(kv)->v_type & ~(uint32_t)WOLFSENTRY_KV_FLAG_MASK)
03055 #define WOLFSENTRY_KV_V_UINT(kv) ((kv)->a.v_uint)
03057 #define WOLFSENTRY_KV_V_SINT(kv) ((kv)->a.v_sint)
03059 #define WOLFSENTRY_KV_V_FLOAT(kv) ((kv)->a.v_float)
03061 #define WOLFSENTRY_KV_V_STRING_LEN(kv) ((kv)->a.string_len)
03063 #define WOLFSENTRY_KV_V_STRING(kv) ((char *)((kv)->b + (kv)->key_len + 1))
03065 #define WOLFSENTRY_KV_V_BYTES_LEN(kv) ((kv)->a.bytes_len)
03067 #define WOLFSENTRY_KV_V_BYTES(kv) ((kv)->b + (kv)->key_len + 1)
03069 #ifdef WOLFSENTRY_HAVE_JSON_DOM
03070 #define WOLFSENTRY_KV_V_JSON(kv) (&(kv)->a.v_json)
03072 #endif
03073
03074 typedef wolfsentry_errcode_t (*wolfsentry_kv_validator_t)(
03075     WOLFSENTRY_CONTEXT_ARGS_IN,
03076     struct wolfsentry_kv_pair *kv);
03079 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_set_validator(
03080     WOLFSENTRY_CONTEXT_ARGS_IN,
03081     wolfsentry_kv_validator_t validator,
03082     wolfsentry_action_res_t *action_results);
03085 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_set_mutability(
03086     WOLFSENTRY_CONTEXT_ARGS_IN,
03087     const char *key,
03088     int key_len,
03089     int mutable);
03092 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_mutability(
03093     WOLFSENTRY_CONTEXT_ARGS_IN,
03094     const char *key,
03095     int key_len,
03096     int *mutable);
03099 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_type(
03100     WOLFSENTRY_CONTEXT_ARGS_IN,
03101     const char *key,
03102     int key_len,
03103     wolfsentry_kv_type_t *type);
03106 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_delete(
03107     WOLFSENTRY_CONTEXT_ARGS_IN,
03108     const char *key,
03109     int key_len);
03112 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_null(
03113     WOLFSENTRY_CONTEXT_ARGS_IN,
```

```
03114     const char *key,
03115     int key_len,
03116     int overwrite_p);
03119 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bool(
03120     WOLFSENTRY_CONTEXT_ARGS_IN,
03121     const char *key,
03122     int key_len,
03123     wolfsentry_kv_type_t value,
03124     int overwrite_p);
03127 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bool(
03128     WOLFSENTRY_CONTEXT_ARGS_IN,
03129     const char *key,
03130     int key_len,
03131     wolfsentry_kv_type_t *value);
03134 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_uint(
03135     WOLFSENTRY_CONTEXT_ARGS_IN,
03136     const char *key,
03137     int key_len,
03138     uint64_t value,
03139     int overwrite_p);
03142 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_uint(
03143     WOLFSENTRY_CONTEXT_ARGS_IN,
03144     const char *key,
03145     int key_len,
03146     uint64_t *value);
03149 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_sint(
03150     WOLFSENTRY_CONTEXT_ARGS_IN,
03151     const char *key,
03152     int key_len,
03153     int64_t value,
03154     int overwrite_p);
03157 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_sint(
03158     WOLFSENTRY_CONTEXT_ARGS_IN,
03159     const char *key,
03160     int key_len,
03161     int64_t *value);
03164 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_double(
03165     WOLFSENTRY_CONTEXT_ARGS_IN,
03166     const char *key,
03167     int key_len,
03168     double value,
03169     int overwrite_p);
03172 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_float(
03173     WOLFSENTRY_CONTEXT_ARGS_IN,
03174     const char *key,
03175     int key_len,
03176     double *value);
03179 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_string(
03180     WOLFSENTRY_CONTEXT_ARGS_IN,
03181     const char *key,
03182     int key_len,
03183     const char *value,
03184     int value_len,
03185     int overwrite_p);
03188 struct wolfsentry_kv_pair_internal;
03189
03196 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_string(
03197     WOLFSENTRY_CONTEXT_ARGS_IN,
03198     const char *key,
03199     int key_len,
03200     const char **value,
03201     int *value_len,
03202     struct wolfsentry_kv_pair_internal **user_value_record);
03203
03204 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bytes(
03205     WOLFSENTRY_CONTEXT_ARGS_IN,
03206     const char *key,
03207     int key_len,
03208     const byte *value,
03209     int value_len,
03210     int overwrite_p);
03213 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bytes_base64(
03214     WOLFSENTRY_CONTEXT_ARGS_IN,
03215     const char *key,
03216     int key_len,
03217     const char *value,
03218     int value_len,
03219     int overwrite_p);
03228 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bytes(
03229     WOLFSENTRY_CONTEXT_ARGS_IN,
03230     const char *key,
03231     int key_len,
03232     const byte **value,
03233     int *value_len,
03234     struct wolfsentry_kv_pair_internal **user_value_record);
03235
03236 #ifdef WOLFSENTRY_HAVE_JSON_DOM
```

```
03237 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_json(
03238     WOLFSENTRY_CONTEXT_ARGS_IN,
03239     const char *key,
03240     int key_len,
03241     JSON_VALUE *value,
03242     int overwrite_p);
03251 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_json(
03252     WOLFSENTRY_CONTEXT_ARGS_IN,
03253     const char *key,
03254     int key_len,
03255     JSON_VALUE **value,
03256     struct wolfsentry_kv_pair_internal **user_value_record);
03257 #endif /* WOLFSENTRY_HAVE_JSON_DOM */
03258
03259 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_release_record(
03260     WOLFSENTRY_CONTEXT_ARGS_IN,
03261     struct wolfsentry_kv_pair_internal **user_value_record);
03264 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_pair_export(
03265     WOLFSENTRY_CONTEXT_ARGS_IN,
03266     struct wolfsentry_kv_pair_internal *kv,
03267     const struct wolfsentry_kv_pair **kv_exports);
03270 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_type_to_string(
03271     wolfsentry_kv_type_t type,
03272     const char **out);
03275 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_render_value(
03276     WOLFSENTRY_CONTEXT_ARGS_IN,
03277     const struct wolfsentry_kv_pair *kv,
03278     char *out,
03279     int *out_len);
03282 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_start(
03283     WOLFSENTRY_CONTEXT_ARGS_IN,
03284     struct wolfsentry_cursor **cursor);
03287 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_seek_to_head(
03288     WOLFSENTRY_CONTEXT_ARGS_IN,
03289     struct wolfsentry_cursor *cursor);
03292 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_seek_to_tail(
03293     WOLFSENTRY_CONTEXT_ARGS_IN,
03294     struct wolfsentry_cursor *cursor);
03297 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_current(
03298     WOLFSENTRY_CONTEXT_ARGS_IN,
03299     struct wolfsentry_cursor *cursor,
03300     struct wolfsentry_kv_pair_internal **kv);
03303 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_prev(
03304     WOLFSENTRY_CONTEXT_ARGS_IN,
03305     struct wolfsentry_cursor *cursor,
03306     struct wolfsentry_kv_pair_internal **kv);
03309 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_next(
03310     WOLFSENTRY_CONTEXT_ARGS_IN,
03311     struct wolfsentry_cursor *cursor,
03312     struct wolfsentry_kv_pair_internal **kv);
03315 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_end(
03316     WOLFSENTRY_CONTEXT_ARGS_IN,
03317     struct wolfsentry_cursor **cursor);
03320 #define WOLFSENTRY_BASE64_DECODED_BUFSPC(buf, len) \
03321     (((((len)+3)/4)*3) - ((len) > 1 ? \
03322                           ((buf)[(len)-1] == '=') : \
03323                           0) \
03324      - ((len) > 2 ? ((buf)[(len)-2] == '=') : 0)) \
03325
03327 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_base64_decode(
03328     const char *src,
03329     size_t src_len,
03330     byte *dest,
03331     size_t *dest_spc,
03332     int ignore_junk_p);
03337 #ifdef WOLFSENTRY_LWIP
03338     #include "wolfsentry/wolfsentry_lwip.h"
03339 #endif
03340
03341 /* conditionally include wolfsentry_util.h last -- none of the above rely on it.
03342  */
03343 #ifndef WOLFSENTRY_NO_UTIL_H
03344 #include <wolfsentry/wolfsentry_util.h>
03345 #endif
03346
03347 #endif /* WOLFSENTRY_H */
```

10.6 wolfsentry/wolfsentry_af.h File Reference

Definitions for address families.

Macros

- #define **WOLFSENTRY_AF_UNSPEC** 0
- #define **WOLFSENTRY_AF_UNIX** 1

  Unix domain sockets.
- #define **WOLFSENTRY_AF_LOCAL** 1

  POSIX name for WOLFSENTRY_AF_UNIX.
- #define **WOLFSENTRY_AF_INET** 2

  Internet IP Protocol.
- #define **WOLFSENTRY_AF_AX25** 3

  Amateur Radio AX.25.
- #define **WOLFSENTRY_AF_IPX** 4

  Novell IPX.
- #define **WOLFSENTRY_AF_APPLETALK** 5

  AppleTalk DDP.
- #define **WOLFSENTRY_AF_NETROM** 6

  Amateur Radio NET/ROM.
- #define **WOLFSENTRY_AF_BRIDGE** 7

  Multiprotocol bridge.
- #define **WOLFSENTRY_AF_ATMPVC** 8

  ATM PVCs.
- #define **WOLFSENTRY_AF_X25** 9

  Reserved for X.25 project.
- #define **WOLFSENTRY_AF_INET6** 10

  IP version 6.
- #define **WOLFSENTRY_AF_ROSE** 11

  Amateur Radio X.25 PLP.
- #define **WOLFSENTRY_AF_DECnet** 12

  Reserved for DECnet project.
- #define **WOLFSENTRY_AF_NETBEUI** 13

  Reserved for 802.2LLC project.
- #define **WOLFSENTRY_AF_SECURITY** 14

  Security callback pseudo AF.
- #define **WOLFSENTRY_AF_KEY** 15

  PF_KEY key management API.
- #define **WOLFSENTRY_AF_NETLINK** 16
- #define **WOLFSENTRY_AF_ROUTE** WOLFSENTRY_AF_NETLINK

  Alias to emulate 4.4BSD.
- #define **WOLFSENTRY_AF_PACKET** 17

  Packet family.
- #define **WOLFSENTRY_AF_ASH** 18

  Ash.
- #define **WOLFSENTRY_AF_ECONET** 19

  Acorn Econet.
- #define **WOLFSENTRY_AF_ATMSVC** 20

  ATM SVCs.
- #define **WOLFSENTRY_AF_RDS** 21

  RDS sockets.
- #define **WOLFSENTRY_AF_SNA** 22

  Linux SNA Project (nutters!)
- #define **WOLFSENTRY_AF_IRDA** 23

IRDA sockets.

- #define **WOLFSENTRY_AF_PPPOX** 24

  PPPoX sockets.
- #define **WOLFSENTRY_AF_WANPIPE** 25

  Wanpipe API Sockets.
- #define **WOLFSENTRY_AF_LLC** 26

  Linux LLC.
- #define **WOLFSENTRY_AF_IB** 27

  Native InfiniBand address.
- #define **WOLFSENTRY_AF_MPLS** 28

  MPLS.
- #define **WOLFSENTRY_AF_CAN** 29

  Controller Area Network.
- #define **WOLFSENTRY_AF_TIPC** 30

  TIPC sockets.
- #define **WOLFSENTRY_AF_BLUETOOTH** 31

  Bluetooth sockets.
- #define **WOLFSENTRY_AF_IUCV** 32

  IUCV sockets.
- #define **WOLFSENTRY_AF_RXRPC** 33

  RxRPC sockets.
- #define **WOLFSENTRY_AF_ISDN** 34

  mISDN sockets
- #define **WOLFSENTRY_AF_PHONET** 35

  Phonet sockets.
- #define **WOLFSENTRY_AF_IEEE802154** 36

  IEEE802154 sockets.
- #define **WOLFSENTRY_AF_CAIF** 37

  CAIF sockets.
- #define **WOLFSENTRY_AF_ALG** 38

  Algorithm sockets.
- #define **WOLFSENTRY_AF_NFC** 39

  NFC sockets.
- #define **WOLFSENTRY_AF_VSOCK** 40

  vSockets
- #define **WOLFSENTRY_AF_KCM** 41

  Kernel Connection Multiplexor.
- #define **WOLFSENTRY_AF_QIPCRTR** 42

  Qualcomm IPC Router.
- #define **WOLFSENTRY_AF_SMC** 43

  smc sockets: reserve number for PF_SMC protocol family that reuses WOLFSENTRY_AF_INET address family
- #define **WOLFSENTRY_AF_XDP** 44

  XDP sockets.
- #define **WOLFSENTRY_AF_BSD_OFFSET** 100

  from FreeBSD at commit a56e5ad6, except WOLFSENTRY_AF_LINK64, added here.
- #define **WOLFSENTRY_AF_IMPLINK** (WOLFSENTRY_AF_BSD_OFFSET + 3)

  arpanet imp addresses
- #define **WOLFSENTRY_AF_PUP** (WOLFSENTRY_AF_BSD_OFFSET + 4)

  pup protocols: e.g. BSP
- #define **WOLFSENTRY_AF_CHAOS** (WOLFSENTRY_AF_BSD_OFFSET + 5)

  mit CHAOS protocols

- #define **WOLFSENTRY_AF_NETBIOS** (WOLFSENTRY_AF_BSD_OFFSET + 6)

  SMB protocols.
- #define **WOLFSENTRY_AF_ISO** (WOLFSENTRY_AF_BSD_OFFSET + 7)

  ISO protocols.
- #define **WOLFSENTRY_AF_OSI** WOLFSENTRY_AF_ISO
- #define **WOLFSENTRY_AF_ECMA** (WOLFSENTRY_AF_BSD_OFFSET + 8)

  European computer manufacturers.
- #define **WOLFSENTRY_AF_DATAKIT** (WOLFSENTRY_AF_BSD_OFFSET + 9)

  datakit protocols
- #define **WOLFSENTRY_AF_DLI** (WOLFSENTRY_AF_BSD_OFFSET + 13)

  DEC Direct data link interface.
- #define **WOLFSENTRY_AF_LAT** (WOLFSENTRY_AF_BSD_OFFSET + 14)

  LAT.
- #define **WOLFSENTRY_AF_HYLINK** (WOLFSENTRY_AF_BSD_OFFSET + 15)

  NSC Hyperchannel.
- #define **WOLFSENTRY_AF_LINK48** (WOLFSENTRY_AF_BSD_OFFSET + 18)

  Link layer interface, explicit EUI-48.
- #define **WOLFSENTRY_AF_LINK** WOLFSENTRY_AF_LINK48

  Link layer interface, implicit EUI-48.
- #define **WOLFSENTRY_AF_LINK64** (WOLFSENTRY_AF_BSD_OFFSET + 19)

  Link layer interface, explicit EUI-64.
- #define **WOLFSENTRY_AF_COIP** (WOLFSENTRY_AF_BSD_OFFSET + 20)

  connection-oriented IP, aka ST II
- #define **WOLFSENTRY_AF_CNT** (WOLFSENTRY_AF_BSD_OFFSET + 21)

  Computer Network Technology.
- #define **WOLFSENTRY_AF_SIP** (WOLFSENTRY_AF_BSD_OFFSET + 24)

  Simple Internet Protocol.
- #define **WOLFSENTRY_AF_SLOW** (WOLFSENTRY_AF_BSD_OFFSET + 33)

  802.3ad slow protocol
- #define **WOLFSENTRY_AF_SCLUSTER** (WOLFSENTRY_AF_BSD_OFFSET + 34)

  Sitara cluster protocol.
- #define **WOLFSENTRY_AF_ARP** (WOLFSENTRY_AF_BSD_OFFSET + 35)
- #define **WOLFSENTRY_AF_IEEE80211** (WOLFSENTRY_AF_BSD_OFFSET + 37)

  IEEE 802.11 protocol.
- #define **WOLFSENTRY_AF_INET_SDP** (WOLFSENTRY_AF_BSD_OFFSET + 40)

  OFED Socket Direct Protocol ipv4.
- #define **WOLFSENTRY_AF_INET6_SDP** (WOLFSENTRY_AF_BSD_OFFSET + 42)

  OFED Socket Direct Protocol ipv6.
- #define **WOLFSENTRY_AF_HYPERV** (WOLFSENTRY_AF_BSD_OFFSET + 43)

  HyperV sockets.
- #define **WOLFSENTRY_AF_USER_OFFSET** 256

10.6.1 Detailed Description

Definitions for address families.

Included by `wolfsentry.h`.

10.7 wolfsentry_af.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry_af.h
00003  *
00004  * Copyright (C) 2022-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_AF_H
00030 #define WOLFSENTRY_AF_H
00031
00036 /* per Linux kernel 5.12, include/linux/socket.h */
00037
00038 #define WOLFSENTRY_AF_UNSPEC       0
00039 #define WOLFSENTRY_AF_UNIX         1
00040 #define WOLFSENTRY_AF_LOCAL        1
00041 #define WOLFSENTRY_AF_INET         2
00042 #define WOLFSENTRY_AF_AX25         3
00043 #define WOLFSENTRY_AF_IPX          4
00044 #define WOLFSENTRY_AF_APPLETALK    5
00045 #define WOLFSENTRY_AF_NETROM       6
00046 #define WOLFSENTRY_AF_BRIDGE       7
00047 #define WOLFSENTRY_AF_ATMPVC       8
00048 #define WOLFSENTRY_AF_X25          9
00049 #define WOLFSENTRY_AF_INET6        10
00050 #define WOLFSENTRY_AF_ROSE         11
00051 #define WOLFSENTRY_AF_DECnet       12
00052 #define WOLFSENTRY_AF_NETBEUI      13
00053 #define WOLFSENTRY_AF_SECURITY     14
00054 #define WOLFSENTRY_AF_KEY          15
00055 #define WOLFSENTRY_AF_NETLINK      16
00056 #define WOLFSENTRY_AF_ROUTE        WOLFSENTRY_AF_NETLINK
00057 #define WOLFSENTRY_AF_PACKET       17
00058 #define WOLFSENTRY_AF_ASH          18
00059 #define WOLFSENTRY_AF_ECONET       19
00060 #define WOLFSENTRY_AF_ATMSVC       20
00061 #define WOLFSENTRY_AF_RDS          21
00062 #define WOLFSENTRY_AF_SNA          22
00063 #define WOLFSENTRY_AF_IRDA         23
00064 #define WOLFSENTRY_AF_PPPOX        24
00065 #define WOLFSENTRY_AF_WANPIPE      25
00066 #define WOLFSENTRY_AF_LLC          26
00067 #define WOLFSENTRY_AF_IB           27
00068 #define WOLFSENTRY_AF_MPLS         28
00069 #define WOLFSENTRY_AF_CAN          29
00070 #define WOLFSENTRY_AF_TIPC         30
00071 #define WOLFSENTRY_AF_BLUETOOTH    31
00072 #define WOLFSENTRY_AF_IUCV         32
00073 #define WOLFSENTRY_AF_RXRPC        33
00074 #define WOLFSENTRY_AF_ISDN         34
00075 #define WOLFSENTRY_AF_PHONET       35
00076 #define WOLFSENTRY_AF_IEEE802154   36
00077 #define WOLFSENTRY_AF_CAIF         37
00078 #define WOLFSENTRY_AF_ALG          38
00079 #define WOLFSENTRY_AF_NFC          39
00080 #define WOLFSENTRY_AF_VSOCK        40
00081 #define WOLFSENTRY_AF_KCM          41
00082 #define WOLFSENTRY_AF_QIPCRTR      42
00083 #define WOLFSENTRY_AF_SMC          43
00084 #define WOLFSENTRY_AF_XDP          44
00086 #define WOLFSENTRY_AF_BSD_OFFSET 100
00087
00089 #define WOLFSENTRY_AF_IMPLINK      (WOLFSENTRY_AF_BSD_OFFSET + 3)
00090 #define WOLFSENTRY_AF_PUP          (WOLFSENTRY_AF_BSD_OFFSET + 4)
00091 #define WOLFSENTRY_AF_CHAOS        (WOLFSENTRY_AF_BSD_OFFSET + 5)
00092 #define WOLFSENTRY_AF_NETBIOS      (WOLFSENTRY_AF_BSD_OFFSET + 6)
00093 #define WOLFSENTRY_AF_ISO          (WOLFSENTRY_AF_BSD_OFFSET + 7)
00094 #define WOLFSENTRY_AF_OSI          WOLFSENTRY_AF_ISO
```

```
00095 #define WOLFSENTRY_AF_ECMA         (WOLFSENTRY_AF_BSD_OFFSET + 8)
00096 #define WOLFSENTRY_AF_DATAKIT      (WOLFSENTRY_AF_BSD_OFFSET + 9)
00097 #define WOLFSENTRY_AF_DLI          (WOLFSENTRY_AF_BSD_OFFSET + 13)
00098 #define WOLFSENTRY_AF_LAT          (WOLFSENTRY_AF_BSD_OFFSET + 14)
00099 #define WOLFSENTRY_AF_HYLINK       (WOLFSENTRY_AF_BSD_OFFSET + 15)
00100 #define WOLFSENTRY_AF_LINK48       (WOLFSENTRY_AF_BSD_OFFSET + 18)
00101 #define WOLFSENTRY_AF_LINK         WOLFSENTRY_AF_LINK48
00102 #define WOLFSENTRY_AF_LINK64       (WOLFSENTRY_AF_BSD_OFFSET + 19)
00103 #define WOLFSENTRY_AF_COIP         (WOLFSENTRY_AF_BSD_OFFSET + 20)
00104 #define WOLFSENTRY_AF_CNT          (WOLFSENTRY_AF_BSD_OFFSET + 21)
00105 #define WOLFSENTRY_AF_SIP          (WOLFSENTRY_AF_BSD_OFFSET + 24)
00106 #define WOLFSENTRY_AF_SLOW         (WOLFSENTRY_AF_BSD_OFFSET + 33)
00107 #define WOLFSENTRY_AF_SCLUSTER     (WOLFSENTRY_AF_BSD_OFFSET + 34)
00108 #define WOLFSENTRY_AF_ARP          (WOLFSENTRY_AF_BSD_OFFSET + 35)
00109 #define WOLFSENTRY_AF_IEEE80211    (WOLFSENTRY_AF_BSD_OFFSET + 37)
00110 #define WOLFSENTRY_AF_INET_SDP     (WOLFSENTRY_AF_BSD_OFFSET + 40)
00111 #define WOLFSENTRY_AF_INET6_SDP    (WOLFSENTRY_AF_BSD_OFFSET + 42)
00112 #define WOLFSENTRY_AF_HYPERV       (WOLFSENTRY_AF_BSD_OFFSET + 43)
00114 #define WOLFSENTRY_AF_USER_OFFSET 256
00115
00118 #endif /* WOLFSENTRY_AF_H */
```

10.8 wolfsentry/wolfsentry_errcodes.h File Reference

Definitions for diagnostics.

`#include <errno.h>`

Macros

- #define **WOLFSENTRY_SOURCE_ID**

  In each source file in the wolfSentry library, `WOLFSENTRY_SOURCE_ID` is defined to a number that is decoded using `enum wolfsentry_source_id`. Application source files that use the below error encoding and rendering macros must also define `WOLFSENTRY_SOURCE_ID` to a number, starting with `WOLFSENTRY_SOURCE_ID_USER_BASE`, and can use `wolfsentry_user_source_string_set()` or `WOLFSENTRY_REGISTER_SOURCE()` to arrange for error and warning messages that render the source code file by name.
- #define **WOLFSENTRY_ERRCODE_FMT**

  String-literal macro for formatting `wolfsentry_errcode_t` using `printf()`-type functions.
- #define **WOLFSENTRY_SOURCE_ID_MAX** 127
- #define **WOLFSENTRY_ERROR_ID_MAX** 255
- #define **WOLFSENTRY_LINE_NUMBER_MAX** 65535
- #define **WOLFSENTRY_ERROR_DECODE_ERROR_CODE**(x)

  Extract the bare error (negative) or success (zero/positive) code from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_DECODE_SOURCE_ID**(x)

  Extract the bare source file ID from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_DECODE_LINE_NUMBER**(x)

  Extract the bare source line number from an encoded `wolfsentry_errcode_t`
- #define **WOLFSENTRY_ERROR_RECODE**(x)

  Take an encoded `wolfsentry_errcode_t` and recode it with the current source ID and line number.
- #define **WOLFSENTRY_ERROR_CODE_IS**(x, name)

  Take an encoded `wolfsentry_errcode_t x` and test if its error code matches short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_CODE_IS**(x, name)

  Take an encoded `wolfsentry_errcode_t x` and test if its error code matches short-form success `name` (e.g. `OK`).
- #define **WOLFSENTRY_IS_FAILURE**(x)

  Evaluates to true if `x` is a `wolfsentry_errcode_t` that encodes a failure.

- #define **WOLFSENTRY_IS_SUCCESS**(x)

  Evaluates to true if `x` is a `wolfsentry_errcode_t` that encodes a success.
- #define **WOLFSENTRY_ERROR_FMT**

  Convenience string-constant macro for formatting a `wolfsentry_errcode_t` for rendering by a `printf`-type function.
- #define **WOLFSENTRY_ERROR_FMT_ARGS**(x)

  Convenience macro supplying args to match the format directives in `WOLFSENTRY_ERROR_FMT`.
- #define **WOLFSENTRY_ERROR_ENCODE**(name)

  Compute a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_ENCODE**(x)

  Compute a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `OK`).
- #define WOLFSENTRY_DEBUG_CALL_TRACE

  Define to build the library or application to output codepoint and error code info at each return point.
- #define **WOLFSENTRY_ERROR_RETURN**(x)

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).
- #define **WOLFSENTRY_SUCCESS_RETURN**(x)

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `OK`).
- #define **WOLFSENTRY_ERROR_RETURN_RECODED**(x)

  Take an encoded `wolfsentry_errcode_t`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_ERROR_RERETURN**(x)

  Return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_RETURN_VALUE**(x)

  Return an arbitrary value.
- #define **WOLFSENTRY_RETURN_VOID**

  Return from a void function.
- #define **WOLFSENTRY_SUCCESS_RETURN_RECODED**(x)

  Take an encoded `wolfsentry_errcode_t`, recode it with the current source ID and line number, and return it.
- #define **WOLFSENTRY_SUCCESS_RERETURN**(x)

  Return an encoded `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_UNLOCK_FOR_RETURN_EX**(ctx)

  Unlock a previously locked `wolfsentry_context`, and if the unlock fails, return the error.
- #define **WOLFSENTRY_UNLOCK_FOR_RETURN**()

  Unlock the current context, and if the unlock fails, return the error.
- #define **WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN_EX**(ctx)

  Unlock a previously locked `wolfsentry_context`, and abandon a held promotion reservation if any (see `wolfsentry_lock_unlock()`), and if the operation fails, return the error.
- #define **WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN**()

  Unlock the current context, and abandon a held promotion reservation if any (see `wolfsentry_lock_unlock()`), and if the operation fails, return the error.
- #define **WOLFSENTRY_MUTEX_EX**(ctx)

  Get a mutex on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_MUTEX_OR_RETURN**()

  Get a mutex on the current context, and on failure, return the `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_SHARED_EX**(ctx)

  Get a shared lock on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_SHARED_OR_RETURN**()

  Get a shared lock on the current context, and on failure, return the `wolfsentry_errcode_t`.
- #define **WOLFSENTRY_PROMOTABLE_EX**(ctx)

Get a mutex on a `wolfsentry_context`, evaluating to the resulting `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_PROMOTABLE_OR_RETURN**()

  Get a shared lock with mutex promotion reservation on the current context, and on failure, return the `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_UNLOCK_AND_RETURN**(ret)

  Unlock the current context, and return the supplied `wolfsentry_errcode_t` .

- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN**(name)

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).

- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED**(x)

  Unlock the current context, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.

- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_EX**(ctx, name)

  Unlock a previously locked `wolfsentry_context ctx`, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form error `name` (e.g. `INVALID_ARG`).

- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED_EX**(ctx, x)

  Unlock a previously locked `wolfsentry_context ctx`, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.

- #define **WOLFSENTRY_ERROR_UNLOCK_AND_RERETURN**(x)

  Unlock the current context, and return an encoded `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_ERROR_RERETURN_AND_UNLOCK**(y)

  Calculate the `wolfsentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN**(name)

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the designated short-form success `name` (e.g. `INVALID_ARG`).

- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN_RECODED**(x)

  Unlock the current context, then take an encoded `wolfsentry_errcode_t x`, recode it with the current source ID and line number, and return it.

- #define **WOLFSENTRY_SUCCESS_UNLOCK_AND_RERETURN**(x)

  Unlock the current context, and return an encoded `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_SUCCESS_RERETURN_AND_UNLOCK**(y)

  Calculate the `wolfsentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfsentry_errcode_t`.

- #define **WOLFSENTRY_UNLOCK_AND_RETURN_VALUE**(x)

  Unlock the current context, and return a value `x`.

- #define **WOLFSENTRY_UNLOCK_AND_RETURN_VOID**

  Unlock the current context, and return void.

- #define **WOLFSENTRY_RETURN_OK**

  Return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.

- #define **WOLFSENTRY_UNLOCK_AND_RETURN_OK**

  Unlock the current context, and return a `wolfsentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.

- #define **WOLFSENTRY_RERETURN_IF_ERROR**(y)

  If `wolfsentry_errcode_t y` is a failure code, return it.

- #define **WOLFSENTRY_UNLOCK_AND_RERETURN_IF_ERROR**(y)

  If `wolfsentry_errcode_t y` is a failure code, unlock the current context and return the code.

- #define **WOLFSENTRY_WARN**(fmt, ...)

  Render a warning message using `WOLFSENTRY_PRINTF_ERR()`, or if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, `DO_NOTHING`.

- #define **WOLFSENTRY_WARN_ON_FAILURE**(...)

  Evaluate the supplied expression, and if the resulting `wolfsentry_errcode_t` encodes an error, render the expression and the decoded error using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, don't render a warning.

- #define **WOLFSENTRY_WARN_ON_FAILURE_LIBC**(...)

  Evaluate the supplied expression, and if it evaluates to a negative value, render the expression and the decoded `errno` using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO_STREAMS` or `WOLFSENTRY_NO_DIAG_MSGS` is set, don't render a warning.
- #define **WOLFSENTRY_REGISTER_SOURCE**()

  Helper macro to call `wolfsentry_user_source_string_set()` with appropriate arguments.
- #define **WOLFSENTRY_REGISTER_ERROR**(name, msg)

  Helper macro to call `wolfsentry_user_error_string_set()` with appropriate arguments, given a short-form `name` and freeform string `msg`.

Typedefs

- typedef int32_t **wolfsentry_errcode_t**

  The structured result code type for wolfSentry. It encodes a failure or success code, a source code file ID, and a line number.

Enumerations

- enum **wolfsentry_source_id** {
  WOLFSENTRY_SOURCE_ID_UNSET = 0 ,
  WOLFSENTRY_SOURCE_ID_ACTIONS_C = 1 ,
  WOLFSENTRY_SOURCE_ID_EVENTS_C = 2 ,
  WOLFSENTRY_SOURCE_ID_WOLFSENTRY_INTERNAL_C = 3 ,
  WOLFSENTRY_SOURCE_ID_ROUTES_C = 4 ,
  WOLFSENTRY_SOURCE_ID_WOLFSENTRY_UTIL_C = 5 ,
  WOLFSENTRY_SOURCE_ID_KV_C = 6 ,
  WOLFSENTRY_SOURCE_ID_ADDR_FAMILIES_C = 7 ,
  WOLFSENTRY_SOURCE_ID_JSON_LOAD_CONFIG_C = 8 ,
  WOLFSENTRY_SOURCE_ID_JSON_JSON_UTIL_C = 9 ,
  WOLFSENTRY_SOURCE_ID_LWIP_PACKET_FILTER_GLUE_C = 10 ,
  WOLFSENTRY_SOURCE_ID_ACTION_BUILTINS_C = 11 ,
  WOLFSENTRY_SOURCE_ID_USER_BASE = 112 }
- enum **wolfsentry_error_id** {
  WOLFSENTRY_ERROR_ID_OK = 0 ,
  WOLFSENTRY_ERROR_ID_NOT_OK = -1 ,
  WOLFSENTRY_ERROR_ID_INTERNAL_CHECK_FATAL = -2 ,
  WOLFSENTRY_ERROR_ID_SYS_OP_FATAL = -3 ,
  WOLFSENTRY_ERROR_ID_SYS_OP_FAILED = -4 ,
  WOLFSENTRY_ERROR_ID_SYS_RESOURCE_FAILED = -5 ,
  WOLFSENTRY_ERROR_ID_INCOMPATIBLE_STATE = -6 ,
  WOLFSENTRY_ERROR_ID_TIMED_OUT = -7 ,
  WOLFSENTRY_ERROR_ID_INVALID_ARG = -8 ,
  WOLFSENTRY_ERROR_ID_BUSY = -9 ,
  WOLFSENTRY_ERROR_ID_INTERRUPTED = -10 ,
  WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_BIG = -11 ,
  WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_SMALL = -12 ,
  WOLFSENTRY_ERROR_ID_STRING_ARG_TOO_LONG = -13 ,
  WOLFSENTRY_ERROR_ID_BUFFER_TOO_SMALL = -14 ,
  WOLFSENTRY_ERROR_ID_IMPLEMENTATION_MISSING = -15 ,
  WOLFSENTRY_ERROR_ID_ITEM_NOT_FOUND = -16 ,
  WOLFSENTRY_ERROR_ID_ITEM_ALREADY_PRESENT = -17 ,
  WOLFSENTRY_ERROR_ID_ALREADY_STOPPED = -18 ,
  WOLFSENTRY_ERROR_ID_WRONG_OBJECT = -19 ,
  WOLFSENTRY_ERROR_ID_DATA_MISSING = -20 ,
  WOLFSENTRY_ERROR_ID_NOT_PERMITTED = -21 ,

WOLFSENTRY_ERROR_ID_ALREADY = -22 ,
WOLFSENTRY_ERROR_ID_CONFIG_INVALID_KEY = -23 ,
WOLFSENTRY_ERROR_ID_CONFIG_INVALID_VALUE = -24 ,
WOLFSENTRY_ERROR_ID_CONFIG_OUT_OF_SEQUENCE = -25 ,
WOLFSENTRY_ERROR_ID_CONFIG_UNEXPECTED = -26 ,
WOLFSENTRY_ERROR_ID_CONFIG_MISPLACED_KEY = -27 ,
WOLFSENTRY_ERROR_ID_CONFIG_PARSER = -28 ,
WOLFSENTRY_ERROR_ID_CONFIG_MISSING_HANDLER = -29 ,
WOLFSENTRY_ERROR_ID_CONFIG_JSON_VALUE_SIZE = -30 ,
WOLFSENTRY_ERROR_ID_OP_NOT_SUPP_FOR_PROTO = -31 ,
WOLFSENTRY_ERROR_ID_WRONG_TYPE = -32 ,
WOLFSENTRY_ERROR_ID_BAD_VALUE = -33 ,
WOLFSENTRY_ERROR_ID_DEADLOCK_AVERTED = -34 ,
WOLFSENTRY_ERROR_ID_OVERFLOW_AVERTED = -35 ,
WOLFSENTRY_ERROR_ID_LACKING_MUTEX = -36 ,
WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK = -37 ,
WOLFSENTRY_ERROR_ID_LIB_MISMATCH = -38 ,
WOLFSENTRY_ERROR_ID_LIBCONFIG_MISMATCH = -39 ,
WOLFSENTRY_ERROR_ID_IO_FAILED = -40 ,
WOLFSENTRY_ERROR_ID_WRONG_ATTRIBUTES = -41 ,
WOLFSENTRY_ERROR_ID_USER_BASE = -128 ,
WOLFSENTRY_SUCCESS_ID_OK = 0 ,
WOLFSENTRY_SUCCESS_ID_LOCK_OK_AND_GOT_RESV = 1 ,
WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX = 2 ,
WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK = 3 ,
WOLFSENTRY_SUCCESS_ID_USED_FALLBACK = 4 ,
WOLFSENTRY_SUCCESS_ID_YES = 5 ,
WOLFSENTRY_SUCCESS_ID_NO = 6 ,
WOLFSENTRY_SUCCESS_ID_ALREADY_OK = 7 ,
WOLFSENTRY_SUCCESS_ID_DEFERRED = 8 ,
WOLFSENTRY_SUCCESS_ID_USER_BASE = 128 }

Functions

- WOLFSENTRY_API const char * **wolfsentry_errcode_source_string** (wolfsentry_errcode_t e)

  Return the name of the source code file associated with `wolfsentry_errcode_t e`, or "unknown user defined source", or "unknown source".
- WOLFSENTRY_API const char * **wolfsentry_errcode_error_string** (wolfsentry_errcode_t e)

  Return a description of the failure or success code associated with `wolfsentry_errcode_t e`, or various "unknown" strings if not known.
- WOLFSENTRY_API const char * **wolfsentry_errcode_error_name** (wolfsentry_errcode_t e)

  Return the short name of the failure or success code associated with `wolfsentry_errcode_t e`, or `wolfsentry_errcode_error_string(e)` if not known.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_source_string_set** (enum wolfsentry_↩ source_id wolfsentry_source_id, const char *source_string)

  Register a source code file so that `wolfsentry_errcode_source_string()`, and therefore `WOLFSENTRY_ERROR_FMT_ARG` and `WOLFSENTRY_WARN_ON_FAILURE()`, can render it. Note that `source_string` must be a string constant or otherwise remain valid for the duration of runtime.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_user_error_string_set** (enum wolfsentry_error_↩ id wolfsentry_error_id, const char *message_string)

  Register an error (negative) or success (positive) code, and corresponding message, so that `wolfsentry_errcode_error_strin` and therefore `WOLFSENTRY_ERROR_FMT_ARGS()` and `WOLFSENTRY_WARN_ON_FAILURE()`, can render it in human-readable form. Note that `error_string` must be a string constant or otherwise remain valid for the duration of runtime.

10.8.1 Detailed Description

Definitions for diagnostics.

Included by wolfsentry.h.

10.9 wolfsentry_errcodes.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry_errcodes.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_ERRCODES_H
00030 #define WOLFSENTRY_ERRCODES_H
00031
00036 #ifdef WOLFSENTRY_FOR_DOXYGEN
00037 #define WOLFSENTRY_SOURCE_ID
00039 #endif
00040
00041 typedef int32_t wolfsentry_errcode_t;
00042 #ifdef FREERTOS
00043 #define WOLFSENTRY_ERRCODE_FMT "%d"
00044 #elif defined(PRId32)
00045 #define WOLFSENTRY_ERRCODE_FMT "%" PRId32
00046 #else
00047 #define WOLFSENTRY_ERRCODE_FMT "%d"
00049 #endif
00050
00051 /* these must be all-1s */
00052 #define WOLFSENTRY_SOURCE_ID_MAX 127
00053 #define WOLFSENTRY_ERROR_ID_MAX 255
00054 #define WOLFSENTRY_LINE_NUMBER_MAX 65535
00055
00058 #define WOLFSENTRY_ERROR_ENCODE_0(x) (((x) < 0) ?                              \
00059         -(((-(x)) & WOLFSENTRY_ERROR_ID_MAX)                                   \
00060            | ((__LINE__ & WOLFSENTRY_LINE_NUMBER_MAX) « 8)                    \
00061            | ((WOLFSENTRY_SOURCE_ID & WOLFSENTRY_SOURCE_ID_MAX) « 24))       \
00062     :                                                                          \
00063         (((x) & WOLFSENTRY_ERROR_ID_MAX)                                       \
00064            | ((__LINE__ & WOLFSENTRY_LINE_NUMBER_MAX) « 8)                    \
00065            | ((WOLFSENTRY_SOURCE_ID & WOLFSENTRY_SOURCE_ID_MAX) « 24)))
00066
00067 #if defined(__GNUC__) && !defined(__STRICT_ANSI__)
00068 #define WOLFSENTRY_ERROR_ENCODE_1(x) ({                                        \
00069     wolfsentry_errcode_t _xret = (x);                                          \
00070     wolfsentry_static_assert2(((x) >= -WOLFSENTRY_ERROR_ID_MAX)                \
00071                   && ((x) <= WOLFSENTRY_ERROR_ID_MAX),                         \
00072                   "error code must be -"                                       \
00073                   _q(WOLFSENTRY_ERROR_ID_MAX)                                  \
00074                   " <= e <= "                                                  \
00075                   _q(WOLFSENTRY_ERROR_ID_MAX) )                                \
00076     wolfsentry_static_assert2(__LINE__ <= WOLFSENTRY_LINE_NUMBER_MAX,          \
00077                   "line number must be 1-" _q(WOLFSENTRY_LINE_NUMBER_MAX) )    \
00078     wolfsentry_static_assert2((WOLFSENTRY_SOURCE_ID >= 0)                      \
00079                   && (WOLFSENTRY_SOURCE_ID <= 0x7f),                           \
00080                   "source file ID must be 0-" _q(WOLFSENTRY_SOURCE_ID_MAX) )   \
00081     WOLFSENTRY_ERROR_ENCODE_0(_xret);                                          \
00082 })
00083 #else
```

```
00084 #define WOLFSENTRY_ERROR_ENCODE_1(x) WOLFSENTRY_ERROR_ENCODE_0(x)
00085 #endif
00086
00087 #define WOLFSENTRY_ERROR_DECODE_ERROR_CODE_1(x) ((int)(((x) < 0) ? -(-(x) & WOLFSENTRY_ERROR_ID_MAX) :
      ((x) & WOLFSENTRY_ERROR_ID_MAX)))
00088 #define WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(x) ((int)(((x) < 0) ? ((-(x)) » 24) : ((x) » 24)))
00089 #define WOLFSENTRY_ERROR_DECODE_LINE_NUMBER_1(x) ((int)(((x) < 0) ? (((-(x)) » 8) &
      WOLFSENTRY_LINE_NUMBER_MAX) : (((x) » 8) & WOLFSENTRY_LINE_NUMBER_MAX)))
00090
00093 #ifdef WOLFSENTRY_NO_INLINE
00094
00095 #if defined(__GNUC__) && !defined(__STRICT_ANSI__)
00096 #define WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x) ({ wolfsentry_errcode_t _xret = (x);
      WOLFSENTRY_ERROR_DECODE_ERROR_CODE_1(_xret); })
00098 #define WOLFSENTRY_ERROR_DECODE_SOURCE_ID(x) ({ wolfsentry_errcode_t _xret = (x);
      WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(_xret); })
00100 #define WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(x) ({ wolfsentry_errcode_t _xret = (x);
      WOLFSENTRY_ERROR_DECODE_LINE_NUMBER_1(_xret); })
00102 #else
00103 #define WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x) WOLFSENTRY_ERROR_DECODE_ERROR_CODE_1(x)
00104 #define WOLFSENTRY_ERROR_DECODE_SOURCE_ID(x) WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(x)
00105 #define WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(x) WOLFSENTRY_ERROR_DECODE_LINE_NUMBER_1(x)
00106 #endif
00107
00108 #else
00109
00110 static inline int WOLFSENTRY_ERROR_DECODE_ERROR_CODE(wolfsentry_errcode_t x) {
00111     return WOLFSENTRY_ERROR_DECODE_ERROR_CODE_1(x);
00112 }
00113 static inline int WOLFSENTRY_ERROR_DECODE_SOURCE_ID(wolfsentry_errcode_t x) {
00114     return WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(x);
00115 }
00116 static inline int WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(wolfsentry_errcode_t x) {
00117     return WOLFSENTRY_ERROR_DECODE_LINE_NUMBER_1(x);
00118 }
00119
00120 #endif
00121
00122 #define WOLFSENTRY_ERROR_RECODE(x) WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x))
00124 #define WOLFSENTRY_ERROR_CODE_IS(x, name) (WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x) ==
      WOLFSENTRY_ERROR_ID_ ## name)
00126 #define WOLFSENTRY_SUCCESS_CODE_IS(x, name) (WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x) ==
      WOLFSENTRY_SUCCESS_ID_ ## name)
00129 #define WOLFSENTRY_IS_FAILURE(x) ((x)<0)
00131 #define WOLFSENTRY_IS_SUCCESS(x) ((x)>=0)
00134 #ifdef WOLFSENTRY_ERROR_STRINGS
00135 #define WOLFSENTRY_ERROR_FMT "code " WOLFSENTRY_ERRCODE_FMT " (%s), src " WOLFSENTRY_ERRCODE_FMT "
      (%s), line " WOLFSENTRY_ERRCODE_FMT
00137 #define WOLFSENTRY_ERROR_FMT_ARGS(x) WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x),
      wolfsentry_errcode_error_string(x), WOLFSENTRY_ERROR_DECODE_SOURCE_ID(x),
      wolfsentry_errcode_source_string(x), WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(x)
00139 #else
00140 #define WOLFSENTRY_ERROR_FMT "code " WOLFSENTRY_ERRCODE_FMT ", src " WOLFSENTRY_ERRCODE_FMT ", line "
      WOLFSENTRY_ERRCODE_FMT
00141 #define WOLFSENTRY_ERROR_FMT_ARGS(x) WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x),
      WOLFSENTRY_ERROR_DECODE_SOURCE_ID(x), WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(x)
00142 #endif /* WOLFSENTRY_ERROR_STRINGS */
00143
00144 #define WOLFSENTRY_ERROR_ENCODE(name) WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_ID_ ## name)
00146 #define WOLFSENTRY_SUCCESS_ENCODE(x) WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_SUCCESS_ID_ ## x)
00149 #ifdef WOLFSENTRY_FOR_DOXYGEN
00150 #define WOLFSENTRY_DEBUG_CALL_TRACE
00161 #undef WOLFSENTRY_DEBUG_CALL_TRACE
00162 #endif
00163
00164 #if defined(WOLFSENTRY_DEBUG_CALL_TRACE) && !defined(WOLFSENTRY_NO_STDIO_STREAMS)
00165     #define WOLFSENTRY_ERROR_RETURN(x) WOLFSENTRY_ERROR_RETURN_1(WOLFSENTRY_ERROR_ID_ ## x)
00166     #define WOLFSENTRY_SUCCESS_RETURN(x) WOLFSENTRY_ERROR_RETURN_1(WOLFSENTRY_SUCCESS_ID_ ## x)
00167     #if defined(WOLFSENTRY_ERROR_STRINGS) && defined(__GNUC__) && !defined(__STRICT_ANSI__)
00168         #ifdef WOLFSENTRY_CALL_DEPTH_RETURNS_STRING
00169         WOLFSENTRY_API const char *_wolfsentry_call_depth(void);
00170         #define _INDENT_FMT "%s"
00171         #define _INDENT_ARGS _wolfsentry_call_depth()
00172         #else
00173         WOLFSENTRY_API unsigned int _wolfsentry_call_depth(void);
00174         #define _INDENT_FMT "%*s"
00175         #define _INDENT_ARGS _wolfsentry_call_depth(), ""
00176         #endif
00177         #define WOLFSENTRY_ERROR_RETURN_1(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
      ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR(_INDENT_FMT "%s L%d %s(): return %d (%s)\n",
      _INDENT_ARGS, _fn, __LINE__, __FUNCTION__, x, wolfsentry_errcode_error_name(x)); return
      WOLFSENTRY_ERROR_ENCODE_1(x); } while (0)
00178         #define WOLFSENTRY_ERROR_RETURN_RECODED(x) do { wolfsentry_errcode_t _xret = (x); const char
      *_fn = strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; }
      WOLFSENTRY_PRINTF_ERR(_INDENT_FMT "%s L%d %s(): return-recoded %d (%s)\n", _INDENT_ARGS, _fn,
      __LINE__, __FUNCTION__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret),
      wolfsentry_errcode_error_name(_xret)); return
```

```
    WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); } while (0)
00179         #define WOLFSENTRY_ERROR_RERETURN(x) do { wolfsentry_errcode_t _xret = (x); const char *_fn =
    strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR(_INDENT_FMT
    "%s L%d %s(): rereturn %d (%s)\n", _INDENT_ARGS, _fn, __LINE__, __FUNCTION__,
    WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret), wolfsentry_errcode_error_name(_xret)); return (_xret); }
    while (0)
00180         #define WOLFSENTRY_RETURN_VALUE(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR(_INDENT_FMT "%s L%d %s(): return value\n",
    _INDENT_ARGS, _fn, __LINE__, __FUNCTION__); return (x); } while (0)
00181         #define WOLFSENTRY_RETURN_VOID do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR(_INDENT_FMT "%s L%d %s(): return void\n",
    _INDENT_ARGS, _fn, __LINE__, __FUNCTION__); return; } while (0)
00182     #elif defined(WOLFSENTRY_ERROR_STRINGS)
00183         #define WOLFSENTRY_ERROR_RETURN_1(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return %d (%s)\n", _fn, __LINE__, x,
    wolfsentry_errcode_error_name(x)); return WOLFSENTRY_ERROR_ENCODE_1(x); } while (0)
00184         #define WOLFSENTRY_ERROR_RETURN_RECODED(x) do { wolfsentry_errcode_t _xret = (x); const char
    *_fn = strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s
    L%d: return-recoded %d (%s)\n", _fn, __LINE__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret),
    wolfsentry_errcode_error_name(_xret)); return
    WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); } while (0)
00185         #define WOLFSENTRY_ERROR_RERETURN(x) do { wolfsentry_errcode_t _xret = (x); const char *_fn =
    strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d:
    rereturn %d (%s)\n", _fn, __LINE__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret),
    wolfsentry_errcode_error_name(_xret)); return (_xret); } while (0)
00186         #define WOLFSENTRY_RETURN_VALUE(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return value\n", _fn, __LINE__);
    return (x); } while (0)
00187         #define WOLFSENTRY_RETURN_VOID do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return void\n", _fn, __LINE__);
    return; } while (0)
00188     #else
00189         #define WOLFSENTRY_ERROR_RETURN_1(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return %d\n", _fn, __LINE__, x);
    return WOLFSENTRY_ERROR_ENCODE_1(x); } while (0)
00190         #define WOLFSENTRY_ERROR_RETURN_RECODED(x) do { wolfsentry_errcode_t _xret = (x); const char
    *_fn = strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s
    L%d: return-recoded %d\n", _fn, __LINE__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); return
    WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); } while (0)
00191         #define WOLFSENTRY_ERROR_RERETURN(x) do { wolfsentry_errcode_t _xret = (x); const char *_fn =
    strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d:
    rereturn %d\n", _fn, __LINE__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); return (_xret); } while (0)
00192         #define WOLFSENTRY_RETURN_VALUE(x) do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return value\n", _fn, __LINE__);
    return (x); } while (0)
00193         #define WOLFSENTRY_RETURN_VOID do { const char *_fn = strrchr(__FILE__, '/'); if (_fn) {
    ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d: return void\n", _fn, __LINE__);
    return; } while (0)
00194     #endif
00195 #else
00196     #define WOLFSENTRY_ERROR_RETURN(x) return WOLFSENTRY_ERROR_ENCODE(x)
00198     #define WOLFSENTRY_SUCCESS_RETURN(x) return WOLFSENTRY_SUCCESS_ENCODE(x)
00200     #define WOLFSENTRY_ERROR_RETURN_RECODED(x) return
    WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(x))
00202     #define WOLFSENTRY_ERROR_RERETURN(x) return (x)
00204     #define WOLFSENTRY_RETURN_VALUE(x) return (x)
00206     #define WOLFSENTRY_RETURN_VOID return
00208 #endif
00209
00210 #define WOLFSENTRY_SUCCESS_RETURN_RECODED(x) WOLFSENTRY_ERROR_RETURN_RECODED(x)
00212 #define WOLFSENTRY_SUCCESS_RERETURN(x) WOLFSENTRY_ERROR_RERETURN(x)
00215 #ifdef WOLFSENTRY_THREADSAFE
00216
00217     #define WOLFSENTRY_UNLOCK_FOR_RETURN_EX(ctx) do {                   \
00218         wolfsentry_errcode_t _lock_ret;                                 \
00219         if ((_lock_ret = wolfsentry_context_unlock(ctx, thread)) < 0) { \
00220             WOLFSENTRY_ERROR_RERETURN(_lock_ret);                       \
00221         }                                                               \
00222     } while (0)
00225     #define WOLFSENTRY_UNLOCK_FOR_RETURN() WOLFSENTRY_UNLOCK_FOR_RETURN_EX(wolfsentry)
00228     #define WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN_EX(ctx) do { \
00229         wolfsentry_errcode_t _lock_ret;                             \
00230         if ((_lock_ret = wolfsentry_context_unlock_and_abandon_reservation(ctx, thread)) < 0) { \
00231             WOLFSENTRY_ERROR_RERETURN(_lock_ret);                   \
00232         }                                                           \
00233     } while (0)
00236     #define WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN()
    WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN_EX(wolfsentry)
00239     #define WOLFSENTRY_MUTEX_EX(ctx) wolfsentry_context_lock_mutex_abstimed(ctx, thread, NULL)
00242     #define WOLFSENTRY_MUTEX_OR_RETURN() do {                   \
00243         wolfsentry_errcode_t _lock_ret;                         \
00244         if ((_lock_ret = WOLFSENTRY_MUTEX_EX(wolfsentry)) < 0)  \
00245             WOLFSENTRY_ERROR_RERETURN(_lock_ret);               \
00246     } while (0)
00249     #define WOLFSENTRY_SHARED_EX(ctx) wolfsentry_context_lock_shared_abstimed(ctx, thread, NULL)
00252     #define WOLFSENTRY_SHARED_OR_RETURN() do {                  \
00253         wolfsentry_errcode_t _lock_ret;                         \
```

```
00254         if (thread == NULL)                                         \
00255             _lock_ret = WOLFSENTRY_MUTEX_EX(wolfsentry);            \
00256         else                                                        \
00257             _lock_ret = WOLFSENTRY_SHARED_EX(wolfsentry);           \
00258         WOLFSENTRY_RERETURN_IF_ERROR(_lock_ret);                    \
00259     } while (0)
00262     #define WOLFSENTRY_PROMOTABLE_EX(ctx)
      wolfsentry_context_lock_shared_with_reservation_abstimed(ctx, thread, NULL)
00265     #define WOLFSENTRY_PROMOTABLE_OR_RETURN() do {                  \
00266         wolfsentry_errcode_t _lock_ret;                             \
00267         if (thread == NULL)                                         \
00268             _lock_ret = WOLFSENTRY_MUTEX_EX(wolfsentry);            \
00269         else                                                        \
00270             _lock_ret = WOLFSENTRY_PROMOTABLE_EX(wolfsentry);       \
00271         WOLFSENTRY_RERETURN_IF_ERROR(_lock_ret);                    \
00272     } while (0)
00275     #define WOLFSENTRY_UNLOCK_AND_RETURN(ret) do {                  \
00276         WOLFSENTRY_UNLOCK_FOR_RETURN();                             \
00277         WOLFSENTRY_ERROR_RERETURN(ret);                             \
00278     } while (0)
00281 #else
00282     #define WOLFSENTRY_UNLOCK_FOR_RETURN() DO_NOTHING
00283     #define WOLFSENTRY_UNLOCK_FOR_RETURN_EX(ctx) DO_NOTHING
00284     #define WOLFSENTRY_MUTEX_EX(ctx) ((void)(ctx), WOLFSENTRY_ERROR_ENCODE(OK))
00285     #define WOLFSENTRY_MUTEX_OR_RETURN() (void)wolfsentry
00286     #define WOLFSENTRY_SHARED_EX(ctx) (void)(ctx)
00287     #define WOLFSENTRY_SHARED_OR_RETURN() (void)wolfsentry
00288     #define WOLFSENTRY_PROMOTABLE_EX(ctx) (void)(ctx)
00289     #define WOLFSENTRY_PROMOTABLE_OR_RETURN() (void)wolfsentry
00290     #define WOLFSENTRY_UNLOCK_AND_RETURN(lock, ret) WOLFSENTRY_ERROR_RERETURN(ret)
00291 #endif
00292
00293 #define WOLFSENTRY_ERROR_UNLOCK_AND_RETURN(name) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_ERROR_RETURN(name); } while (0)
00295 #define WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED(x) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_ERROR_RETURN_RECODED(x); } while (0)
00297 #define WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_EX(ctx, name) do { WOLFSENTRY_UNLOCK_FOR_RETURN_EX(ctx);
      WOLFSENTRY_ERROR_RETURN(name); } while (0)
00299 #define WOLFSENTRY_ERROR_UNLOCK_AND_RETURN_RECODED_EX(ctx, x) do {
      WOLFSENTRY_UNLOCK_FOR_RETURN_EX(ctx); WOLFSENTRY_ERROR_RETURN_RECODED(x); } while (0)
00301 #define WOLFSENTRY_ERROR_UNLOCK_AND_RERETURN(x) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_ERROR_RERETURN(x); } while (0)
00303 #define WOLFSENTRY_ERROR_RERETURN_AND_UNLOCK(y) do { wolfsentry_errcode_t _yret = (y);
      WOLFSENTRY_UNLOCK_FOR_RETURN(); WOLFSENTRY_ERROR_RERETURN(_yret); } while (0)
00306 #define WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN(name) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_SUCCESS_RETURN(name); } while (0)
00308 #define WOLFSENTRY_SUCCESS_UNLOCK_AND_RETURN_RECODED(x) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_SUCCESS_RETURN_RECODED(x); } while (0)
00310 #define WOLFSENTRY_SUCCESS_UNLOCK_AND_RERETURN(x) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_SUCCESS_RERETURN(x); } while (0)
00312 #define WOLFSENTRY_SUCCESS_RERETURN_AND_UNLOCK(y) do { wolfsentry_errcode_t _yret = (y);
      WOLFSENTRY_UNLOCK_FOR_RETURN(); WOLFSENTRY_SUCCESS_RERETURN(_yret); } while (0)
00315 #define WOLFSENTRY_UNLOCK_AND_RETURN_VALUE(x) do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_RETURN_VALUE(x); } while (0)
00317 #define WOLFSENTRY_UNLOCK_AND_RETURN_VOID do { WOLFSENTRY_UNLOCK_FOR_RETURN(); WOLFSENTRY_RETURN_VOID;
      } while (0)
00320 #define WOLFSENTRY_RETURN_OK WOLFSENTRY_SUCCESS_RETURN(OK)
00322 #define WOLFSENTRY_UNLOCK_AND_RETURN_OK do { WOLFSENTRY_UNLOCK_FOR_RETURN();
      WOLFSENTRY_SUCCESS_RETURN(OK); } while (0)
00324 #define WOLFSENTRY_RERETURN_IF_ERROR(y) do { wolfsentry_errcode_t _yret = (y); if (_yret < 0)
      WOLFSENTRY_ERROR_RERETURN(_yret); } while (0)
00326 #define WOLFSENTRY_UNLOCK_AND_RERETURN_IF_ERROR(y) do { wolfsentry_errcode_t _yret = (y); if (_yret <
      0) { WOLFSENTRY_UNLOCK_FOR_RETURN(); WOLFSENTRY_ERROR_RERETURN(_yret); } } while (0)
00329 #ifdef WOLFSENTRY_ERROR_STRINGS
00330 WOLFSENTRY_API const char *wolfsentry_errcode_source_string(wolfsentry_errcode_t e);
00332 WOLFSENTRY_API const char *wolfsentry_errcode_error_string(wolfsentry_errcode_t e);
00334 WOLFSENTRY_API const char *wolfsentry_errcode_error_name(wolfsentry_errcode_t e);
00336 #endif
00337
00338 #if !defined(WOLFSENTRY_NO_STDIO_STREAMS) && !defined(WOLFSENTRY_NO_DIAG_MSGS)
00339
00340 #include <errno.h>
00341
00342 #ifdef __STRICT_ANSI__
00343 #define WOLFSENTRY_WARN(fmt,...) WOLFSENTRY_PRINTF_ERR("%s@L%d " fmt, __FILE__, __LINE__, __VA_ARGS__)
00344 #else
00345 #define WOLFSENTRY_WARN(fmt,...) WOLFSENTRY_PRINTF_ERR("%s@L%d " fmt, __FILE__, __LINE__, ##
      __VA_ARGS__)
00347 #endif
00348
00349 #define WOLFSENTRY_WARN_ON_FAILURE(...) do { wolfsentry_errcode_t _ret = (__VA_ARGS__); if (_ret < 0)
      { WOLFSENTRY_WARN(#__VA_ARGS__ ": " WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(_ret)); }}
      while(0)
00351 #define WOLFSENTRY_WARN_ON_FAILURE_LIBC(...) do { if ((__VA_ARGS__) < 0) {
      WOLFSENTRY_WARN(#__VA_ARGS__ ": %s\n", strerror(errno)); }} while(0)
00354 #else
00355
```

```
00356 #define WOLFSENTRY_WARN(fmt,...) DO_NOTHING
00357 #define WOLFSENTRY_WARN_ON_FAILURE(...) do { if ((__VA_ARGS__) < 0) {} } while (0)
00358 #define WOLFSENTRY_WARN_ON_FAILURE_LIBC(...) do { if ((__VA_ARGS__) < 0) {}} while (0)
00359
00360 #endif /* !WOLFSENTRY_NO_STDIO_STREAMS && !WOLFSENTRY_NO_DIAG_MSGS */
00361
00362 #ifdef WOLFSENTRY_CPPCHECK
00363     #undef WOLFSENTRY_ERROR_ENCODE
00364     #define WOLFSENTRY_ERROR_ENCODE(x) 0
00365     #undef WOLFSENTRY_SUCCESS_ENCODE
00366     #define WOLFSENTRY_SUCCESS_ENCODE(x) 0
00367 #endif
00368
00369 enum wolfsentry_source_id {
00370     WOLFSENTRY_SOURCE_ID_UNSET      =  0,
00371     WOLFSENTRY_SOURCE_ID_ACTIONS_C  =  1,
00372     WOLFSENTRY_SOURCE_ID_EVENTS_C   =  2,
00373     WOLFSENTRY_SOURCE_ID_WOLFSENTRY_INTERNAL_C =  3,
00374     WOLFSENTRY_SOURCE_ID_ROUTES_C   =  4,
00375     WOLFSENTRY_SOURCE_ID_WOLFSENTRY_UTIL_C     =  5,
00376     WOLFSENTRY_SOURCE_ID_KV_C       =  6,
00377     WOLFSENTRY_SOURCE_ID_ADDR_FAMILIES_C = 7,
00378     WOLFSENTRY_SOURCE_ID_JSON_LOAD_CONFIG_C = 8,
00379     WOLFSENTRY_SOURCE_ID_JSON_JSON_UTIL_C = 9,
00380     WOLFSENTRY_SOURCE_ID_LWIP_PACKET_FILTER_GLUE_C = 10,
00381     WOLFSENTRY_SOURCE_ID_ACTION_BUILTINS_C = 11,
00382
00383     WOLFSENTRY_SOURCE_ID_USER_BASE  =  112
00384 };
00385
00386 #ifdef WOLFSENTRY_ERROR_STRINGS
00387 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_source_string_set(enum wolfsentry_source_id
      wolfsentry_source_id, const char *source_string);
00389 #define WOLFSENTRY_REGISTER_SOURCE() wolfsentry_user_source_string_set(WOLFSENTRY_SOURCE_ID,__FILE__)
00391 #endif
00392
00393 enum wolfsentry_error_id {
00394     WOLFSENTRY_ERROR_ID_OK                    =    0,
00395     WOLFSENTRY_ERROR_ID_NOT_OK                =   -1,
00396     WOLFSENTRY_ERROR_ID_INTERNAL_CHECK_FATAL  =   -2,
00397     WOLFSENTRY_ERROR_ID_SYS_OP_FATAL          =   -3,
00398     WOLFSENTRY_ERROR_ID_SYS_OP_FAILED         =   -4,
00399     WOLFSENTRY_ERROR_ID_SYS_RESOURCE_FAILED   =   -5,
00400     WOLFSENTRY_ERROR_ID_INCOMPATIBLE_STATE    =   -6,
00401     WOLFSENTRY_ERROR_ID_TIMED_OUT             =   -7,
00402     WOLFSENTRY_ERROR_ID_INVALID_ARG           =   -8,
00403     WOLFSENTRY_ERROR_ID_BUSY                  =   -9,
00404     WOLFSENTRY_ERROR_ID_INTERRUPTED           =  -10,
00405     WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_BIG   =  -11,
00406     WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_SMALL =  -12,
00407     WOLFSENTRY_ERROR_ID_STRING_ARG_TOO_LONG   =  -13,
00408     WOLFSENTRY_ERROR_ID_BUFFER_TOO_SMALL      =  -14,
00409     WOLFSENTRY_ERROR_ID_IMPLEMENTATION_MISSING =  -15,
00410     WOLFSENTRY_ERROR_ID_ITEM_NOT_FOUND        =  -16,
00411     WOLFSENTRY_ERROR_ID_ITEM_ALREADY_PRESENT  =  -17,
00412     WOLFSENTRY_ERROR_ID_ALREADY_STOPPED       =  -18,
00413     WOLFSENTRY_ERROR_ID_WRONG_OBJECT          =  -19,
00414     WOLFSENTRY_ERROR_ID_DATA_MISSING          =  -20,
00415     WOLFSENTRY_ERROR_ID_NOT_PERMITTED         =  -21,
00416     WOLFSENTRY_ERROR_ID_ALREADY               =  -22,
00417     WOLFSENTRY_ERROR_ID_CONFIG_INVALID_KEY    =  -23,
00418     WOLFSENTRY_ERROR_ID_CONFIG_INVALID_VALUE  =  -24,
00419     WOLFSENTRY_ERROR_ID_CONFIG_OUT_OF_SEQUENCE =  -25,
00420     WOLFSENTRY_ERROR_ID_CONFIG_UNEXPECTED     =  -26,
00421     WOLFSENTRY_ERROR_ID_CONFIG_MISPLACED_KEY  =  -27,
00422     WOLFSENTRY_ERROR_ID_CONFIG_PARSER         =  -28,
00423     WOLFSENTRY_ERROR_ID_CONFIG_MISSING_HANDLER =  -29,
00424     WOLFSENTRY_ERROR_ID_CONFIG_JSON_VALUE_SIZE =  -30,
00425     WOLFSENTRY_ERROR_ID_OP_NOT_SUPP_FOR_PROTO  =  -31,
00426     WOLFSENTRY_ERROR_ID_WRONG_TYPE            =  -32,
00427     WOLFSENTRY_ERROR_ID_BAD_VALUE             =  -33,
00428     WOLFSENTRY_ERROR_ID_DEADLOCK_AVERTED      =  -34,
00429     WOLFSENTRY_ERROR_ID_OVERFLOW_AVERTED      =  -35,
00430     WOLFSENTRY_ERROR_ID_LACKING_MUTEX         =  -36,
00431     WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK     =  -37,
00432     WOLFSENTRY_ERROR_ID_LIB_MISMATCH          =  -38,
00433     WOLFSENTRY_ERROR_ID_LIBCONFIG_MISMATCH    =  -39,
00434     WOLFSENTRY_ERROR_ID_IO_FAILED             =  -40,
00435     WOLFSENTRY_ERROR_ID_WRONG_ATTRIBUTES      =  -41,
00436
00437     WOLFSENTRY_ERROR_ID_USER_BASE             = -128,
00438
00439     WOLFSENTRY_SUCCESS_ID_OK                  =    0,
00440     WOLFSENTRY_SUCCESS_ID_LOCK_OK_AND_GOT_RESV =    1,
00441     WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX          =    2,
00442     WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK      =    3,
00443     WOLFSENTRY_SUCCESS_ID_USED_FALLBACK       =    4,
```

```
00444     WOLFSENTRY_SUCCESS_ID_YES                 =    5,
00445     WOLFSENTRY_SUCCESS_ID_NO                  =    6,
00446     WOLFSENTRY_SUCCESS_ID_ALREADY_OK          =    7,
00447     WOLFSENTRY_SUCCESS_ID_DEFERRED            =    8,
00448     WOLFSENTRY_SUCCESS_ID_USER_BASE           =  128
00449 };
00450
00451 #ifdef WOLFSENTRY_ERROR_STRINGS
00452 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_error_string_set(enum wolfsentry_error_id
      wolfsentry_error_id, const char *message_string);
00454 #define WOLFSENTRY_REGISTER_ERROR(name, msg) wolfsentry_user_error_string_set(WOLFSENTRY_ERROR_ID_ ##
      name, msg)
00456 #endif
00457
00460 #endif /* WOLFSENTRY_ERRCODES_H */
```

10.10 wolfsentry/wolfsentry_json.h File Reference

Types and prototypes for loading/reloading configuration using JSON.

```
#include "wolfsentry.h"
#include "centijson_sax.h"
```

Macros

- #define **WOLFSENTRY**
- #define **WOLFSENTRY_MAX_JSON_NESTING** 16

  Can be overridden.

Typedefs

- typedef uint32_t **wolfsentry_config_load_flags_t**

  Type for holding flag bits from wolfsentry_config_load_flags.

Enumerations

- enum wolfsentry_config_load_flags {
  WOLFSENTRY_CONFIG_LOAD_FLAG_NONE ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES ,
  WOLFSENTRY_CONFIG_LOAD_FLAG_FINI }

  Flags to be ORd together to communicate options to wolfsentry_config_json_init()

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_centijson_errcode_translate** (wolfsentry_errcode_t centijson_errcode)

  Convert CentiJSON numeric error code to closest-corresponding wolfSentry error code.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_init** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_config_load_flags_t load_flags, struct wolfsentry_json_process_state **jps)

  Allocate and initialize a `struct wolfsentry_json_process_state` with the designated `load_flags`, to subsequently pass to `wolfsentry_config_json_feed()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_init_ex** (WOLFSENTRY_CONTEXT_ARGS_IN, wolfsentry_config_load_flags_t load_flags, const JSON_CONFIG *json_config, struct wolfsentry_json_↩ process_state **jps)

  Variant of `wolfsentry_config_json_init()` with an additional `JSON_CONFIG` argument, `json_↩ config`, for tailoring of JSON parsing dynamics.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_feed** (struct wolfsentry_json_process↩ _state *jps, const unsigned char *json_in, size_t json_in_len, char *err_buf, size_t err_buf_size)

  Pass a segment of JSON configuration into the parsing engine. Segments can be as short or as long as desired, to facilitate incremental read-in.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_centijson_errcode** (struct wolfsentry_json↩ _process_state *jps, int *json_errcode, const char **json_errmsg)

  Copy the current error code and/or human-readable error message from a `struct wolfsentry_json_↩ process_state` allocated by `wolfsentry_config_json_init()`.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_fini** (struct wolfsentry_json_process↩ _state **jps, char *err_buf, size_t err_buf_size)

  To be called when done iterating `wolfsentry_config_json_feed()`, completing the configuration load.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_oneshot** (WOLFSENTRY_CONTEXT_ARGS_IN, const unsigned char *json_in, size_t json_in_len, wolfsentry_config_load_flags_t load_flags, char *err_buf, size_t err_buf_size)

  Load a complete JSON configuration from an in-memory buffer.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_config_json_oneshot_ex** (WOLFSENTRY_CONTEXT_ARGS_IN, const unsigned char *json_in, size_t json_in_len, wolfsentry_config_load_flags_t load_flags, const JSON_CONFIG *json_config, char *err_buf, size_t err_buf_size)

  Variant of `wolfsentry_config_json_oneshot()` with an additional `JSON_CONFIG` argument, `json_↩ config`, for tailoring of JSON parsing dynamics.

10.10.1 Detailed Description

Types and prototypes for loading/reloading configuration using JSON.

Include this file in your application for JSON configuration capabilities.

10.11 wolfsentry_json.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry_json.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
```

```
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_JSON_H
00030 #define WOLFSENTRY_JSON_H
00031
00032 #include "wolfsentry.h"
00033
00034 #ifndef WOLFSENTRY
00035 #define WOLFSENTRY
00036 #endif
00037 #include "centijson_sax.h"
00038
00043 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_centijson_errcode_translate(wolfsentry_errcode_t
      centijson_errcode);
00046 #ifndef WOLFSENTRY_MAX_JSON_NESTING
00047 #define WOLFSENTRY_MAX_JSON_NESTING 16
00049 #endif
00050
00051 typedef uint32_t wolfsentry_config_load_flags_t;
00055 enum wolfsentry_config_load_flags {
00056     WOLFSENTRY_CONFIG_LOAD_FLAG_NONE             = 0U,
00058     WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH         = 1U << 0U,
00060     WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN          = 1U << 1U,
00062     WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT = 1U << 2U,
00064     WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS = 1U << 3U,
00066     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT = 1U << 4U,
00068     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST = 1U << 5U,
00070     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST = 1U << 6U,
00072     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAINDICTORDER = 1U << 7U,
00074     WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES = 1U << 8U,
00076     WOLFSENTRY_CONFIG_LOAD_FLAG_FINI             = 1U << 30U
00078 };
00079
00080 struct wolfsentry_json_process_state;
00081
00082 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_init(
00083     WOLFSENTRY_CONTEXT_ARGS_IN,
00084     wolfsentry_config_load_flags_t load_flags,
00085     struct wolfsentry_json_process_state **jps);
00088 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_init_ex(
00089     WOLFSENTRY_CONTEXT_ARGS_IN,
00090     wolfsentry_config_load_flags_t load_flags,
00091     const JSON_CONFIG *json_config,
00092     struct wolfsentry_json_process_state **jps);
00095 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_feed(
00096     struct wolfsentry_json_process_state *jps,
00097     const unsigned char *json_in,
00098     size_t json_in_len,
00099     char *err_buf,
00100     size_t err_buf_size);
00103 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_centijson_errcode(struct
      wolfsentry_json_process_state *jps, int *json_errcode, const char **json_errmsg);
00106 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_fini(
00107     struct wolfsentry_json_process_state **jps,
00108     char *err_buf,
00109     size_t err_buf_size);
00112 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_oneshot(
00113     WOLFSENTRY_CONTEXT_ARGS_IN,
00114     const unsigned char *json_in,
00115     size_t json_in_len,
00116     wolfsentry_config_load_flags_t load_flags,
00117     char *err_buf,
00118     size_t err_buf_size);
00121 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_config_json_oneshot_ex(
00122     WOLFSENTRY_CONTEXT_ARGS_IN,
00123     const unsigned char *json_in,
00124     size_t json_in_len,
00125     wolfsentry_config_load_flags_t load_flags,
00126     const JSON_CONFIG *json_config,
00127     char *err_buf,
00128     size_t err_buf_size);
00133 #endif /* WOLFSENTRY_JSON_H */
```

10.12 wolfsentry/wolfsentry_lwip.h File Reference

Prototypes for lwIP callback installation functions, for use in lwIP applications.

```
#include "lwip/init.h"
#include "lwip/filter.h"
```

Functions

- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_ethernet_callback** (WOLFSENTRY_CONTEXT_AR
  packet_filter_event_mask_t ethernet_mask)

  Install wolfSentry callbacks into lwIP for ethernet (layer 2) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_ip_callbacks** (WOLFSENTRY_CONTEXT_ARGS_
  packet_filter_event_mask_t ip_mask)

  Install wolfSentry callbacks into lwIP for IPv4/IPv6 (layer 3) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_icmp_callbacks** (WOLFSENTRY_CONTEXT_ARG
  packet_filter_event_mask_t icmp_mask)

  Install wolfSentry callbacks into lwIP for ICMP filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_tcp_callback** (WOLFSENTRY_CONTEXT_ARGS_
  packet_filter_event_mask_t tcp_mask)

  Install wolfSentry callbacks into lwIP for TCP (layer 4) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_udp_callback** (WOLFSENTRY_CONTEXT_ARGS_
  packet_filter_event_mask_t udp_mask)

  Install wolfSentry callbacks into lwIP for UDP (layer 4) filtering.
- WOLFSENTRY_API wolfsentry_errcode_t **wolfsentry_install_lwip_filter_callbacks** (WOLFSENTRY_CONTEXT_ARGS_IN,
  packet_filter_event_mask_t ethernet_mask, packet_filter_event_mask_t ip_mask, packet_filter_event_↩
  mask_t icmp_mask, packet_filter_event_mask_t tcp_mask, packet_filter_event_mask_t udp_mask)

  Install wolfSentry callbacks for all layers/protocols enabled by the supplied masks.
- WOLFSENTRY_API_VOID **wolfsentry_cleanup_lwip_filter_callbacks** (WOLFSENTRY_CONTEXT_ARGS_IN,
  void *arg)

  Disables any wolfSentry callbacks previously installed in lwIP.

10.12.1 Detailed Description

Prototypes for lwIP callback installation functions, for use in lwIP applications.

`packet_filter_event_mask_t` is passed to lwIP via the callback installation routines, to designate which events are of interest. It is set to a bitwise-OR of values from `packet_filter_event_t`, defined in `src/include/lwip/filter.h` in the lwIP source tree after applying `lwip/LWIP_PACKET_FILTER↩_API.patch`. The values are:

`FILT_BINDING` – Call into wolfSentry (filter) on binding events
`FILT_DISSOCIATE` – Call into wolfSentry on socket dissociation events
`FILT_LISTENING` – Call into wolfSentry at initiation of socket listening
`FILT_STOP_LISTENING` – Call into wolfSentry when listening is shut down
`FILT_CONNECTING` – Call into wolfSentry (filter) when connecting out
`FILT_ACCEPTING` – Call into wolfSentry (filter) when accepting an inbound connection
`FILT_CLOSED` – Call into wolfSentry when socket is closed
`FILT_REMOTE_RESET` – Call into wolfSentry when a connection was reset by the remote peer
`FILT_RECEIVING` – Call into wolfSentry (filter) for each regular inbound packet of data
`FILT_SENDING` – Call into wolfSentry (filter) for each regular outbound packet of data
`FILT_ADDR_UNREACHABLE` – Call into wolfSentry when inbound traffic attempts to reach an unknown address
`FILT_PORT_UNREACHABLE` – Call into wolfSentry when inbound traffic attempts to reach an unlistened/unbound port
`FILT_INBOUND_ERR` – Call into wolfSentry when inbound traffic results in detection of an error by lwIP
`FILT_OUTBOUND_ERR` – Call into wolfSentry when outbound traffic results in detection of an error by lwIP

10.13 wolfsentry_lwip.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry/wolfsentry_lwip.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00044 #ifndef WOLFSENTRY_LWIP_H
00045 #define WOLFSENTRY_LWIP_H
00046
00051 #include "lwip/init.h"
00052
00053 #if LWIP_PACKET_FILTER_API
00054
00055 #include "lwip/filter.h"
00056
00057 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_ethernet_callback(
00058     WOLFSENTRY_CONTEXT_ARGS_IN,
00059     packet_filter_event_mask_t ethernet_mask);
00062 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_ip_callbacks(
00063     WOLFSENTRY_CONTEXT_ARGS_IN,
00064     packet_filter_event_mask_t ip_mask);
00067 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_icmp_callbacks(
00068     WOLFSENTRY_CONTEXT_ARGS_IN,
00069     packet_filter_event_mask_t icmp_mask);
00072 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_tcp_callback(
00073     WOLFSENTRY_CONTEXT_ARGS_IN,
00074     packet_filter_event_mask_t tcp_mask);
00077 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_udp_callback(
00078     WOLFSENTRY_CONTEXT_ARGS_IN,
00079     packet_filter_event_mask_t udp_mask);
00082 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_install_lwip_filter_callbacks(
00083     WOLFSENTRY_CONTEXT_ARGS_IN,
00084     packet_filter_event_mask_t ethernet_mask,
00085     packet_filter_event_mask_t ip_mask,
00086     packet_filter_event_mask_t icmp_mask,
00087     packet_filter_event_mask_t tcp_mask,
00088     packet_filter_event_mask_t udp_mask);
00091 WOLFSENTRY_API_VOID wolfsentry_cleanup_lwip_filter_callbacks(
00092     WOLFSENTRY_CONTEXT_ARGS_IN,
00093     void *arg);
00096 #endif /* LWIP_PACKET_FILTER_API */
00097
00100 #endif /* WOLFSENTRY_LWIP_H */
```

10.14 wolfsentry/wolfsentry_settings.h File Reference

Target- and config-specific settings and abstractions for wolfSentry.

```
#include <wolfsentry/wolfsentry_options.h>
#include <inttypes.h>
#include <stdint.h>
#include <stddef.h>
#include <assert.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <time.h>
```

Data Structures

- struct wolfsentry_thread_context_public

  Right-sized, right-aligned opaque container for thread state.
- struct wolfsentry_build_settings

  struct for passing the build version and configuration

Macros

- #define **WOLFSENTRY_USER_SETTINGS_FILE** "the_path"

  Define to the path of a user settings file to be included, containing extra and override definitions and directives. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.
- #define **WOLFSENTRY_NO_ALLOCA**

  Build flag to use only implementations that avoid alloca().
- #define **WOLFSENTRY_C89**

  Build flag to use only constructs that are pedantically legal in C89.
- #define **__attribute_maybe_unused__**

  Attribute abstraction to mark a function or variable (typically a `static`) as possibly unused.
- #define **DO_NOTHING**

  Statement-type abstracted construct that executes no code.
- #define **WOLFSENTRY_NO_INTTYPES_H**

  Define to inhibit inclusion of `inttypes.h` (alternative `typedef`s or `include` must be supplied with WOLFSENTRY_USER_SETTINGS_FILE).
- #define **WOLFSENTRY_NO_STDINT_H**

  Define to inhibit inclusion of `stding.h` (alternative `typedef`s or `include` must be supplied with WOLFSENTRY_USER_SETTINGS_FILE).
- #define **WOLFSENTRY_PRINTF_ERR**(...)

  printf-like macro, expecting a format as first arg, used for rendering warning and error messages. Can be overridden in WOLFSENTRY_USER_SETTINGS_FILE.
- #define **WOLFSENTRY_SINGLETHREADED**

  Define to disable all thread handling and safety in wolfSentry.
- #define **WOLFSENTRY_USE_NONPOSIX_SEMAPHORES**

  Define if POSIX semaphore API is not available. If no non-POSIX builtin implementation is present in wolfsentry_util.c, then WOLFSENTRY_NO_SEM_BUILTIN must be set, and the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include a full semaphore implementation (shim set) in its wolfsentry_semcbs slot.
- #define **WOLFSENTRY_USE_NONPOSIX_THREADS**

  Define if POSIX thread API is not available. `WOLFSENTRY_THREAD_INCLUDE`, `WOLFSENTRY_THREAD_ID_T`, and `WOLFSENTRY_THREAD_GET_ID_HANDLER` will need to be supplied in WOLFSENTRY_USER_SETTINGS_FILE.
- #define **WOLFSENTRY_NO_GNU_ATOMICS**

  Define if gnu-style atomic intrinsics are not available. `WOLFSENTRY_ATOMIC_*()` macro definitions for intrinsics will need to be supplied in WOLFSENTRY_USER_SETTINGS_FILE (see wolfsentry_util.h).
- #define **WOLFSENTRY_NO_CLOCK_BUILTIN**

  If defined, omit built-in time primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_timecbs.
- #define **WOLFSENTRY_NO_SEM_BUILTIN**

  If defined, omit built-in semaphore primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_semcbs.
- #define **WOLFSENTRY_NO_MALLOC_BUILTIN**

  If defined, omit built-in heap allocator primitives; the wolfsentry_host_platform_interface supplied to wolfSentry APIs must include implementations of all functions in wolfsentry_allocator.
- #define **WOLFSENTRY_NO_ERROR_STRINGS**

If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.

- #define **WOLFSENTRY_NO_PROTOCOL_NAMES**

  If defined, omit APIs for rendering error codes and source code files in human readable form. They will be rendered numerically.

- #define **WOLFSENTRY_NO_ADDR_BITMASK_MATCHING**

  If defined, omit support for bitmask matching of addresses, and support only prefix matching.

- #define **WOLFSENTRY_NO_IPV6**

  If defined, omit support for IPv6.

- #define **WOLFSENTRY_MAX_BITMASK_MATCHED_AFS**

  The maximum number of distinct address families that can use bitmask matching in routes. Default value is 4.

- #define **WOLFSENTRY_NO_GETPROTOBY**

  Define this to gate out calls to getprotobyname_r() and getservbyname_r(), necessitating numeric identification of protocols (e.g. 6 for TCP) and services (e.g. 25 for SMTP) in configuration JSON documents.

- #define **WOLFSENTRY_NO_POSIX_MEMALIGN**

  Define if `posix_memalign()` is not available.

- #define **WOLFSENTRY_FLEXIBLE_ARRAY_SIZE**

  Value appropriate as a size for an array that will be allocated to a variable size. Built-in value usually works.

- #define **SIZET_FMT**

  printf-style format string appropriate for pairing with `size_t`

- #define **WOLFSENTRY_ENT_ID_FMT**

  printf-style format string appropriate for pairing with wolfsentry_ent_id_t

- #define **WOLFSENTRY_ENT_ID_NONE**

  always-invalid object ID

- #define **WOLFSENTRY_HITCOUNT_FMT**

  printf-style format string appropriate for pairing with wolfsentry_hitcount_t

- #define **__wolfsentry_wur**

  abstracted attribute designating that the return value must be checked to avoid a compiler warning

- #define **wolfsentry_static_assert**(c)

  abstracted static assert – `c` must be true, else `c` is printed

- #define **wolfsentry_static_assert2**(c, m)

  abstracted static assert – `c` must be true, else `m` is printed

- #define **WOLFSENTRY_DEADLINE_NEVER** (-1)

  Value returned in `deadline->tv_sec` and `deadline->tv_nsec` by wolfsentry_get_thread_deadline() when `thread` has no deadline set. Not allowed as explicit values passed to wolfsentry_set_deadline_abs() – use wolfsentry_clear_deadline() to clear any deadline. Can be overridden with user settings.

- #define **WOLFSENTRY_DEADLINE_NOW** (-2)

  Value returned in `deadline->tv_sec` and `deadline->tv_nsec` by wolfsentry_get_thread_deadline() when `thread` is in non-blocking mode. Not allowed as explicit values passed to wolfsentry_set_deadline_abs() – use wolfsentry_set_deadline_rel_usecs(WOLFSENTRY_CONTEXT_ARGS_OUT, 0) to put thread in non-blocking mode. Can be overridden with user settings.

- #define **WOLFSENTRY_SEMAPHORE_INCLUDE** "the_path"

  Define to the path of a header file declaring a semaphore API. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.

- #define **WOLFSENTRY_THREAD_INCLUDE** "the_path"

  Define to the path of a header file declaring a threading API. Can be an absolute or a relative path, subject to a `-I` path supplied to `make` using `EXTRA_CFLAGS`. Include quotes or <> around the path.

- #define **WOLFSENTRY_THREAD_ID_T** thread_id_type

  Define to the appropriate type analogous to POSIX `pthread_t`.

- #define **WOLFSENTRY_THREAD_GET_ID_HANDLER** pthread_self_ish_function

  Define to the name of a void function analogous to POSIX `pthread_self`, returning a value of type WOLFSENTRY_THREAD_ID_T.

- #define **WOLFSENTRY_THREAD_NO_ID** 0

- #define **WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER** {0}
- #define **WOLFSENTRY_API_VOID**

  Function attribute for declaring/defining public void API functions.
- #define **WOLFSENTRY_API**

  Function attribute for declaring/defining public API functions with return values.
- #define **WOLFSENTRY_LOCAL_VOID**

  Function attribute for declaring/defining private void functions.
- #define **WOLFSENTRY_LOCAL**

  Function attribute for declaring/defining private functions with return values.
- #define **WOLFSENTRY_MAX_ADDR_BYTES** 16

  The maximum size allowed for an address, in bytes. Can be overridden. Note that support for bitmask matching for an address family depends on WOLFSENTRY_MAX_ADDR_BYTES at least twice the max size of a bare address in that family, as the address and mask are internally stored as a single double-length byte vector. Note also that WOLFSENTRY_MAX_ADDR_BYTES entails proportional overhead if wolfSentry is built WOLFSENTRY_NO_ALLOCA or WOLFSENTRY_C89.
- #define **WOLFSENTRY_MAX_ADDR_BITS** (WOLFSENTRY_MAX_ADDR_BYTES∗8)

  The maximum size allowed for an address, in bits. Can be overridden.
- #define **WOLFSENTRY_MAX_LABEL_BYTES** 32

  The maximum size allowed for a label, in bytes. Can be overridden.
- #define **WOLFSENTRY_BUILTIN_LABEL_PREFIX** "%"

  The prefix string reserved for use in names of built-in actions and events.
- #define **WOLFSENTRY_KV_MAX_VALUE_BYTES** 16384

  The maximum size allowed for scalar user-defined values. Can be overridden.
- #define **WOLFSENTRY_CONFIG_SIGNATURE**

  Macro to use as the initializer for wolfsentry_build_settings.config and wolfsentry_host_platform_interface.caller_build_settings.

Typedefs

- typedef unsigned char **byte**

  8 bits unsigned
- typedef uint16_t **wolfsentry_addr_family_t**

  integer type for holding address family number
- typedef uint16_t **wolfsentry_proto_t**

  integer type for holding protocol number
- typedef uint16_t **wolfsentry_port_t**

  integer type for holding port number
- typedef uint32_t **wolfsentry_ent_id_t**

  integer type for holding table entry ID
- typedef uint16_t **wolfsentry_addr_bits_t**

  integer type for address prefix lengths (in bits)
- typedef uint32_t **wolfsentry_hitcount_t**

  integer type for holding hit count statistics
- typedef int64_t **wolfsentry_time_t**

  integer type for holding absolute and relative times, using microseconds in built-in implementations.
- typedef uint16_t **wolfsentry_priority_t**

  integer type for holding event priority (smaller number is higher priority)

10.14.1 Detailed Description

Target- and config-specific settings and abstractions for wolfSentry.

This file is included by wolfsentry.h.

10.15 wolfsentry_settings.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry_settings.h
00003  *
00004  * Copyright (C) 2022-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_SETTINGS_H
00030 #define WOLFSENTRY_SETTINGS_H
00031
00035 #ifdef WOLFSENTRY_FOR_DOXYGEN
00036 #define WOLFSENTRY_USER_SETTINGS_FILE "the_path"
00038 #undef WOLFSENTRY_USER_SETTINGS_FILE
00039 #endif
00040
00041 #ifdef WOLFSENTRY_USER_SETTINGS_FILE
00042     #include WOLFSENTRY_USER_SETTINGS_FILE
00043 #endif
00044
00045 #if !defined(BUILDING_LIBWOLFSENTRY) && !defined(WOLFSENTRY_USER_SETTINGS_FILE)
00046     #include <wolfsentry/wolfsentry_options.h>
00047 #endif
00048
00055 #ifdef WOLFSENTRY_FOR_DOXYGEN
00056 #define WOLFSENTRY_NO_ALLOCA
00057 #undef WOLFSENTRY_NO_ALLOCA
00058 #define WOLFSENTRY_C89
00059 #undef WOLFSENTRY_C89
00060 #endif
00061
00062 #ifdef WOLFSENTRY_C89
00063     #define WOLFSENTRY_NO_INLINE
00064     #ifndef WOLFSENTRY_NO_POSIX_MEMALIGN
00065         #define WOLFSENTRY_NO_POSIX_MEMALIGN
00066     #endif
00067     #define WOLFSENTRY_NO_DESIGNATED_INITIALIZERS
00068     #define WOLFSENTRY_NO_LONG_LONG
00069     #if !defined(WOLFSENTRY_USE_NONPOSIX_SEMAPHORES) && !defined(WOLFSENTRY_SINGLETHREADED)
00070         /* sem_timedwait() was added in POSIX 200112L */
00071         #define WOLFSENTRY_SINGLETHREADED
00072     #endif
00073 #endif
00074
00075 #ifndef __attribute_maybe_unused__
00076 #if defined(__GNUC__)
00077 #define __attribute_maybe_unused__ __attribute__((unused))
00079 #else
00080 #define __attribute_maybe_unused__
00081 #endif
00082 #endif
00083
00084 #ifdef WOLFSENTRY_NO_INLINE
00086 #define inline __attribute_maybe_unused__
00088 #endif
00089
00090 #ifndef DO_NOTHING
00091 #define DO_NOTHING do {} while (0)
00093 #endif
00094
00097 #ifdef FREERTOS
00098     #include <FreeRTOS.h>
00099     #define WOLFSENTRY_CALL_DEPTH_RETURNS_STRING
00100     #if !defined(WOLFSENTRY_NO_STDIO_STREAMS) && !defined(WOLFSENTRY_PRINTF_ERR)
00101         #define WOLFSENTRY_PRINTF_ERR(...) printf(__VA_ARGS__)
00102     #endif
00103
00104     #define FREERTOS_NANOSECONDS_PER_SECOND     1000000000L
```

```
00105     #define FREERTOS_NANOSECONDS_PER_TICK       (FREERTOS_NANOSECONDS_PER_SECOND / configTICK_RATE_HZ)
00106
00107     #if !defined(SIZE_T_32) && !defined(SIZE_T_64)
00108         /* size_t is "unsigned int" in STM32 FreeRTOS */
00109         #define SIZE_T_32
00110     #endif
00111 #endif
00112
00117 #ifdef WOLFSENTRY_FOR_DOXYGEN
00118 #define WOLFSENTRY_NO_INTTYPES_H
00120 #undef WOLFSENTRY_NO_INTTYPES_H
00121 #endif
00122 #ifndef WOLFSENTRY_NO_INTTYPES_H
00123 #include <inttypes.h>
00124 #endif
00125 #ifdef WOLFSENTRY_FOR_DOXYGEN
00126 #define WOLFSENTRY_NO_STDINT_H
00128 #undef WOLFSENTRY_NO_STDINT_H
00129 #endif
00130 #ifndef WOLFSENTRY_NO_STDINT_H
00131 #include <stdint.h>
00132 #endif
00133
00136 #if !defined(SIZE_T_32) && !defined(SIZE_T_64)
00137     #if defined(__WORDSIZE) && (__WORDSIZE == 64)
00138         #define SIZE_T_64
00139     #elif defined(INTPTR_MAX) && defined(INT64_MAX) && (INTPTR_MAX == INT64_MAX)
00140         #define SIZE_T_64
00141     #elif defined(__WORDSIZE) && (__WORDSIZE == 32)
00142         #define SIZE_T_32
00143     #elif defined(INTPTR_MAX) && defined(INT32_MAX) && (INTPTR_MAX == INT32_MAX)
00144         #define SIZE_T_32
00145     #else
00146         #error "must define SIZE_T_32 or SIZE_T_64 with user settings."
00147     #endif
00148 #elif defined(SIZE_T_32) && defined(SIZE_T_64)
00149     #error "must define SIZE_T_32 xor SIZE_T_64."
00150 #endif
00151
00156 #if !defined(WOLFSENTRY_NO_STDIO_STREAMS) && !defined(WOLFSENTRY_PRINTF_ERR)
00157     #define WOLFSENTRY_PRINTF_ERR(...) fprintf(stderr, __VA_ARGS__)
00159 #endif
00160
00167 #ifdef WOLFSENTRY_FOR_DOXYGEN
00168 #define WOLFSENTRY_SINGLETHREADED
00170 #undef WOLFSENTRY_SINGLETHREADED
00171 #endif
00172
00173 #ifndef WOLFSENTRY_SINGLETHREADED
00174
00176 #define WOLFSENTRY_THREADSAFE
00179 #ifdef WOLFSENTRY_FOR_DOXYGEN
00180
00181 #define WOLFSENTRY_USE_NONPOSIX_SEMAPHORES
00183 #undef WOLFSENTRY_USE_NONPOSIX_SEMAPHORES
00184
00185 #define WOLFSENTRY_USE_NONPOSIX_THREADS
00187 #undef WOLFSENTRY_USE_NONPOSIX_THREADS
00188
00189 #define WOLFSENTRY_NO_GNU_ATOMICS
00191 #undef WOLFSENTRY_NO_GNU_ATOMICS
00192
00193 #endif
00194
00195 #ifndef WOLFSENTRY_USE_NONPOSIX_SEMAPHORES
00196     #if defined(__MACH__) || defined(FREERTOS) || defined(_WIN32)
00197         #define WOLFSENTRY_USE_NONPOSIX_SEMAPHORES
00198     #endif
00199 #endif
00200
00201 #ifndef WOLFSENTRY_USE_NONPOSIX_THREADS
00202     #if defined(FREERTOS) || defined(_WIN32)
00203         #define WOLFSENTRY_USE_NONPOSIX_THREADS
00204     #endif
00205 #endif
00206
00209 #ifndef WOLFSENTRY_USE_NONPOSIX_SEMAPHORES
00210     #define WOLFSENTRY_USE_NATIVE_POSIX_SEMAPHORES
00211 #endif
00212
00213 #ifndef WOLFSENTRY_USE_NONPOSIX_THREADS
00214     #define WOLFSENTRY_USE_NATIVE_POSIX_THREADS
00215 #endif
00216
00217 #ifndef WOLFSENTRY_NO_GNU_ATOMICS
00218     #define WOLFSENTRY_HAVE_GNU_ATOMICS
00219 #endif
```

```
00220
00223 #endif /* !WOLFSENTRY_SINGLETHREADED */
00224
00225 #ifdef WOLFSENTRY_FOR_DOXYGEN
00226
00227 #define WOLFSENTRY_NO_CLOCK_BUILTIN
00229 #undef WOLFSENTRY_NO_CLOCK_BUILTIN
00230
00231 #define WOLFSENTRY_NO_SEM_BUILTIN
00233 #undef WOLFSENTRY_NO_SEM_BUILTIN
00234
00235 #define WOLFSENTRY_NO_MALLOC_BUILTIN
00237 #undef WOLFSENTRY_NO_MALLOC_BUILTIN
00238
00239 #define WOLFSENTRY_NO_ERROR_STRINGS
00241 #undef WOLFSENTRY_NO_ERROR_STRINGS
00242
00243 #define WOLFSENTRY_NO_PROTOCOL_NAMES
00245 #undef WOLFSENTRY_NO_PROTOCOL_NAMES
00246
00247 #define WOLFSENTRY_NO_ADDR_BITMASK_MATCHING
00249 #undef WOLFSENTRY_NO_ADDR_BITMASK_MATCHING
00250
00251 #define WOLFSENTRY_NO_IPV6
00253 #undef WOLFSENTRY_NO_IPV6
00254
00255 #endif /* WOLFSENTRY_FOR_DOXYGEN */
00256
00257 #ifndef WOLFSENTRY_MAX_BITMASK_MATCHED_AFS
00258     #define WOLFSENTRY_MAX_BITMASK_MATCHED_AFS 4
00260 #endif
00261
00264 #ifndef WOLFSENTRY_NO_CLOCK_BUILTIN
00265     #define WOLFSENTRY_CLOCK_BUILTINS
00266 #endif
00267
00268 #ifndef WOLFSENTRY_NO_MALLOC_BUILTIN
00269     #define WOLFSENTRY_MALLOC_BUILTINS
00270 #endif
00271
00272 #ifndef WOLFSENTRY_NO_SEM_BUILTIN
00273     #define WOLFSENTRY_SEM_BUILTINS
00274 #endif
00275
00276 #ifndef WOLFSENTRY_NO_ERROR_STRINGS
00277     #define WOLFSENTRY_ERROR_STRINGS
00278 #endif
00279
00280 #ifndef WOLFSENTRY_NO_PROTOCOL_NAMES
00281     #define WOLFSENTRY_PROTOCOL_NAMES
00282 #endif
00283
00284 #ifndef WOLFSENTRY_NO_JSON_DOM
00285     #define WOLFSENTRY_HAVE_JSON_DOM
00286 #endif
00287
00288 #ifndef WOLFSENTRY_NO_ADDR_BITMASK_MATCHING
00289     #define WOLFSENTRY_ADDR_BITMASK_MATCHING
00290 #endif
00291
00292 #ifndef WOLFSENTRY_NO_IPV6
00293     #define WOLFSENTRY_IPV6
00294 #endif
00295
00298 #if !defined(WOLFSENTRY_NO_GETPROTOBY) && (!defined(__GLIBC__) || !defined(__USE_MISC) ||
      defined(WOLFSENTRY_C89))
00299     /* get*by*_r() is non-standard. */
00300     #define WOLFSENTRY_NO_GETPROTOBY
00302 #endif
00303
00310 #if defined(WOLFSENTRY_USE_NATIVE_POSIX_SEMAPHORES) || defined(WOLFSENTRY_CLOCK_BUILTINS) ||
      defined(WOLFSENTRY_MALLOC_BUILTINS)
00311 #ifndef _XOPEN_SOURCE
00312 #if __STDC_VERSION__ >= 201112L
00313 #define _XOPEN_SOURCE 700
00314 #elif __STDC_VERSION__ >= 199901L
00315 #define _XOPEN_SOURCE 600
00316 #else
00317 #define _XOPEN_SOURCE 500
00318 #endif /* __STDC_VERSION__ */
00319 #endif
00320 #endif
00321
00322 #if !defined(WOLFSENTRY_NO_POSIX_MEMALIGN) && (!defined(_POSIX_C_SOURCE) || (_POSIX_C_SOURCE <
      200112L))
00323     #define WOLFSENTRY_NO_POSIX_MEMALIGN
00325 #endif
```

```
00326
00327 #if defined(__STRICT_ANSI__)
00328 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE 1
00329 #elif defined(__GNUC__) && !defined(__clang__)
00330 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE
00332 #else
00333 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE 0
00334 #endif
00335
00338 #ifndef WOLFSENTRY_NO_TIME_H
00339 #ifndef __USE_POSIX199309
00340 /* glibc needs this for struct timespec with -std=c99 */
00341 #define __USE_POSIX199309
00342 #endif
00343 #endif
00344
00347 #ifndef SIZET_FMT
00348     #ifdef SIZE_T_32
00349         #define SIZET_FMT "%u"
00350     #elif __STDC_VERSION__ >= 199901L
00351         #define SIZET_FMT "%zu"
00352     #else
00353         #define SIZET_FMT "%lu"
00355     #endif
00356 #endif
00357
00358 #ifndef WOLFSENTRY_NO_STDDEF_H
00359 #include <stddef.h>
00360 #endif
00361 #ifndef WOLFSENTRY_NO_ASSERT_H
00362 #include <assert.h>
00363 #endif
00364 #ifndef WOLFSENTRY_NO_STDIO_H
00365 #ifndef __USE_ISOC99
00366 /* kludge to make glibc snprintf() prototype visible even when -std=c89 */
00368 #define __USE_ISOC99
00370 #include <stdio.h>
00371 #undef __USE_ISOC99
00372 #else
00373 #include <stdio.h>
00374 #endif
00375 #endif
00376 #ifndef WOLFSENTRY_NO_STRING_H
00377 #include <string.h>
00378 #endif
00379 #ifndef WOLFSENTRY_NO_STRINGS_H
00380 #include <strings.h>
00381 #endif
00382 #ifndef WOLFSENTRY_NO_TIME_H
00383 #include <time.h>
00384 #endif
00385
00386 typedef unsigned char byte;
00389 typedef uint16_t wolfsentry_addr_family_t;
00392 typedef uint16_t wolfsentry_proto_t;
00394 typedef uint16_t wolfsentry_port_t;
00397 #ifdef WOLFSENTRY_ENT_ID_TYPE
00398 typedef WOLFSENTRY_ENT_ID_TYPE wolfsentry_ent_id_t;
00399 #else
00400 typedef uint32_t wolfsentry_ent_id_t;
00402 #endif
00403
00404 #ifndef WOLFSENTRY_ENT_ID_FMT
00405     #ifdef PRIu32
00406         #define WOLFSENTRY_ENT_ID_FMT "%" PRIu32
00407     #elif (defined(__WORDSIZE) && (__WORDSIZE == 32)) || \
00408         (defined(INTPTR_MAX) && defined(INT32_MAX) && (INTPTR_MAX == INT32_MAX))
00409         #define WOLFSENTRY_ENT_ID_FMT "%lu"
00410     #else
00411         #define WOLFSENTRY_ENT_ID_FMT "%u"
00413     #endif
00414 #endif
00415
00416 #define WOLFSENTRY_ENT_ID_NONE 0
00418 typedef uint16_t wolfsentry_addr_bits_t;
00420 #ifdef WOLFSENTRY_HITCOUNT_TYPE
00421 typedef WOLFSENTRY_HITCOUNT_TYPE wolfsentry_hitcount_t;
00422 #else
00423 typedef uint32_t wolfsentry_hitcount_t;
00425 #define WOLFSENTRY_HITCOUNT_FMT "%u"
00427 #endif
00428 #ifdef WOLFSENTRY_TIME_TYPE
00429 typedef WOLFSENTRY_TIME_TYPE wolfsentry_time_t;
00430 #else
00431 typedef int64_t wolfsentry_time_t;
00433 #endif
00434
```

```
00435 #ifdef WOLFSENTRY_PRIORITY_TYPE
00436 typedef WOLFSENTRY_PRIORITY_TYPE wolfsentry_priority_t;
00437 #else
00438 typedef uint16_t wolfsentry_priority_t;
00440 #endif
00441
00442 #ifndef attr_align_to
00443 #ifdef __GNUC__
00444 #define attr_align_to(x) __attribute__((aligned(x)))
00445 #elif defined(_MSC_VER)
00446 /* disable align warning, we want alignment ! */
00447 #pragma warning(disable: 4324)
00448 #define attr_align_to(x) __declspec(align(x))
00449 #else
00450 #error must supply definition for attr_align_to() macro.
00451 #endif
00452 #endif
00453
00454 #ifndef __wolfsentry_wur
00455 #ifdef __wur
00456 #define __wolfsentry_wur __wur
00457 #elif defined(__must_check)
00458 #define __wolfsentry_wur __must_check
00459 #elif defined(__GNUC__) && (__GNUC__ >= 4)
00460 #define __wolfsentry_wur __attribute__((warn_unused_result))
00462 #else
00463 #define __wolfsentry_wur
00464 #endif
00465 #endif
00466
00467 #ifndef wolfsentry_static_assert
00468 #if defined(__GNUC__) && defined(static_assert) && !defined(__STRICT_ANSI__)
00469 /* note semicolon included in expansion, so that assert can completely disappear in ISO C builds. */
00470 #define wolfsentry_static_assert(c) static_assert(c, #c);
00471 #define wolfsentry_static_assert2(c, m) static_assert(c, m);
00472 #else
00473 #define wolfsentry_static_assert(c)
00475 #define wolfsentry_static_assert2(c, m)
00477 #endif
00478 #endif /* !wolfsentry_static_assert */
00479
00486 #if defined(WOLFSENTRY_THREADSAFE)
00487
00488 #ifndef WOLFSENTRY_DEADLINE_NEVER
00489     #define WOLFSENTRY_DEADLINE_NEVER (-1)
00491 #endif
00492 #ifndef WOLFSENTRY_DEADLINE_NOW
00493     #define WOLFSENTRY_DEADLINE_NOW (-2)
00495 #endif
00496
00497 #ifdef WOLFSENTRY_USE_NATIVE_POSIX_SEMAPHORES
00498
00499 #ifdef WOLFSENTRY_SEMAPHORE_INCLUDE
00500
00501 #include WOLFSENTRY_SEMAPHORE_INCLUDE
00502
00503 #else /* !WOLFSENTRY_SEMAPHORE_INCLUDE */
00504
00505 #ifndef __USE_XOPEN2K
00506 /* kludge to force glibc sem_timedwait() prototype visible with -std=c99 */
00507 #define __USE_XOPEN2K
00508 #include <semaphore.h>
00509 #undef __USE_XOPEN2K
00510 #else
00511 #include <semaphore.h>
00512 #endif
00513
00514 #endif /* !WOLFSENTRY_SEMAPHORE_INCLUDE */
00515
00516 #elif defined(__MACH__)
00517
00518 #include <dispatch/dispatch.h>
00519 #include <semaphore.h>
00520 #define sem_t dispatch_semaphore_t
00521
00522 #elif defined(FREERTOS)
00523
00524 #include <atomic.h>
00525
00526 #ifdef WOLFSENTRY_SEMAPHORE_INCLUDE
00527 #include WOLFSENTRY_SEMAPHORE_INCLUDE
00528 #else
00529 #include <semphr.h>
00530 #endif
00531
00532 #define SEM_VALUE_MAX        0x7FFFU
00533
```

```
00534 #define sem_t StaticSemaphore_t
00535
00536 #else
00537
00544 #ifdef WOLFSENTRY_FOR_DOXYGEN
00545 #define WOLFSENTRY_SEMAPHORE_INCLUDE "the_path"
00547 #undef WOLFSENTRY_SEMAPHORE_INCLUDE
00548 #define WOLFSENTRY_THREAD_INCLUDE "the_path"
00550 #undef WOLFSENTRY_THREAD_INCLUDE
00551 #define WOLFSENTRY_THREAD_ID_T thread_id_type
00553 #undef WOLFSENTRY_THREAD_ID_T
00554 #define WOLFSENTRY_THREAD_GET_ID_HANDLER pthread_self_ish_function
00556 #undef WOLFSENTRY_THREAD_GET_ID_HANDLER
00557 #endif
00558
00565 #ifdef WOLFSENTRY_SEMAPHORE_INCLUDE
00566 #include WOLFSENTRY_SEMAPHORE_INCLUDE
00567 #endif
00568
00569 #endif
00570
00571     #ifdef WOLFSENTRY_THREAD_INCLUDE
00572         #include WOLFSENTRY_THREAD_INCLUDE
00573     #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00574         #include <pthread.h>
00575     #endif
00576     #ifdef WOLFSENTRY_THREAD_ID_T
00577         typedef WOLFSENTRY_THREAD_ID_T wolfsentry_thread_id_t;
00578     #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00579         typedef pthread_t wolfsentry_thread_id_t;
00580     #elif defined(FREERTOS)
00581         typedef TaskHandle_t wolfsentry_thread_id_t;
00582     #else
00583         #error Must supply WOLFSENTRY_THREAD_ID_T for WOLFSENTRY_THREADSAFE on non-POSIX targets.
00584     #endif
00585     /* note WOLFSENTRY_THREAD_GET_ID_HANDLER must return WOLFSENTRY_THREAD_NO_ID on failure. */
00586     #ifdef WOLFSENTRY_THREAD_GET_ID_HANDLER
00587     #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00588        #define WOLFSENTRY_THREAD_GET_ID_HANDLER pthread_self
00589     #elif defined(FREERTOS)
00590        #define WOLFSENTRY_THREAD_GET_ID_HANDLER xTaskGetCurrentTaskHandle
00591     #else
00592         #error Must supply WOLFSENTRY_THREAD_GET_ID_HANDLER for WOLFSENTRY_THREADSAFE on non-POSIX
      targets.
00593     #endif
00594
00595     struct wolfsentry_thread_context;
00596
00597     /* WOLFSENTRY_THREAD_NO_ID must be zero. */
00598     #define WOLFSENTRY_THREAD_NO_ID 0
00599
00601     struct wolfsentry_thread_context_public {
00602         uint64_t opaque[8];
00603     };
00604
00605     #define WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER {0}
00606 #endif
00607
00616 #ifdef BUILDING_LIBWOLFSENTRY
00617     #if defined(_MSC_VER) || defined(__MINGW32__) || defined(__CYGWIN__) || \
00618         defined(_WIN32_WCE)
00619         #if defined(WOLFSENTRY_DLL)
00620             #define WOLFSENTRY_API_BASE __declspec(dllexport)
00621         #else
00622             #define WOLFSENTRY_API_BASE
00623         #endif
00624         #define WOLFSENTRY_LOCAL_BASE
00625     #elif defined(HAVE_VISIBILITY) && HAVE_VISIBILITY
00626         #define WOLFSENTRY_API_BASE   __attribute__ ((visibility("default")))
00627         #define WOLFSENTRY_LOCAL_BASE __attribute__ ((visibility("hidden")))
00628     #elif defined(__SUNPRO_C) && (__SUNPRO_C >= 0x550)
00629         #define WOLFSENTRY_API_BASE   __global
00630         #define WOLFSENTRY_LOCAL_BASE __hidden
00631     #else
00632         #define WOLFSENTRY_API_BASE
00633         #define WOLFSENTRY_LOCAL_BASE
00634     #endif /* HAVE_VISIBILITY */
00635 #else /* !BUILDING_LIBWOLFSENTRY */
00636     #if defined(_MSC_VER) || defined(__MINGW32__) || defined(__CYGWIN__) || \
00637         defined(_WIN32_WCE)
00638         #if defined(WOLFSENTRY_DLL)
00639             #define WOLFSENTRY_API_BASE __declspec(dllimport)
00640         #else
00641             #define WOLFSENTRY_API_BASE
00642         #endif
00643         #define WOLFSENTRY_LOCAL_BASE
00644     #else
```

```
00645         #define WOLFSENTRY_API_BASE
00646         #define WOLFSENTRY_LOCAL_BASE
00647     #endif
00648 #endif /* !BUILDING_LIBWOLFSENTRY */
00649
00652 #define WOLFSENTRY_API_VOID WOLFSENTRY_API_BASE void
00654 #define WOLFSENTRY_API WOLFSENTRY_API_BASE __wolfsentry_wur
00657 #define WOLFSENTRY_LOCAL_VOID WOLFSENTRY_LOCAL_BASE void
00659 #define WOLFSENTRY_LOCAL WOLFSENTRY_LOCAL_BASE __wolfsentry_wur
00664 #ifndef WOLFSENTRY_NO_DESIGNATED_INITIALIZERS
00665 #define WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00666 #endif
00667
00668 #ifndef WOLFSENTRY_NO_LONG_LONG
00669 #define WOLFSENTRY_HAVE_LONG_LONG
00670 #endif
00671
00674 #ifndef WOLFSENTRY_MAX_ADDR_BYTES
00675 #define WOLFSENTRY_MAX_ADDR_BYTES 16
00677 #elif WOLFSENTRY_MAX_ADDR_BYTES * 8 > 0xffff
00678 #error WOLFSENTRY_MAX_ADDR_BYTES * 8 must fit in a uint16_t.
00679 #endif
00680
00681 #ifndef WOLFSENTRY_MAX_ADDR_BITS
00682 #define WOLFSENTRY_MAX_ADDR_BITS (WOLFSENTRY_MAX_ADDR_BYTES*8)
00684 #else
00685 #if WOLFSENTRY_MAX_ADDR_BITS > (WOLFSENTRY_MAX_ADDR_BYTES*8)
00686 #error WOLFSENTRY_MAX_ADDR_BITS is too large for given/default WOLFSENTRY_MAX_ADDR_BYTES
00687 #endif
00688 #endif
00689
00690 #ifndef WOLFSENTRY_MAX_LABEL_BYTES
00691 #define WOLFSENTRY_MAX_LABEL_BYTES 32
00693 #elif WOLFSENTRY_MAX_LABEL_BYTES > 0xff
00694 #error WOLFSENTRY_MAX_LABEL_BYTES must fit in a byte.
00695 #endif
00696
00697 #ifndef WOLFSENTRY_BUILTIN_LABEL_PREFIX
00698 #define WOLFSENTRY_BUILTIN_LABEL_PREFIX "%"
00700 #endif
00701
00702 #ifndef WOLFSENTRY_KV_MAX_VALUE_BYTES
00703 #define WOLFSENTRY_KV_MAX_VALUE_BYTES 16384
00705 #endif
00706
00707 #if defined(WOLFSENTRY_ENT_ID_TYPE) ||          \
00708     defined(WOLFSENTRY_HITCOUNT_TYPE) ||        \
00709     defined(WOLFSENTRY_TIME_TYPE) ||            \
00710     defined(WOLFSENTRY_PRIORITY_TYPE) ||        \
00711     defined(WOLFSENTRY_THREAD_ID_T) ||          \
00712     defined(SIZE_T_32) ||                       \
00713     defined(SIZE_T_64)
00714 #define WOLFSENTRY_USER_DEFINED_TYPES
00715 #endif
00716
00725 enum wolfsentry_build_flags {
00726     WOLFSENTRY_CONFIG_FLAG_ENDIANNESS_ONE = (1U « 0U),
00727     WOLFSENTRY_CONFIG_FLAG_USER_DEFINED_TYPES = (1U « 1U),
00728     WOLFSENTRY_CONFIG_FLAG_THREADSAFE = (1U « 2U),
00729     WOLFSENTRY_CONFIG_FLAG_CLOCK_BUILTINS = (1U « 3U),
00730     WOLFSENTRY_CONFIG_FLAG_MALLOC_BUILTINS = (1U « 4U),
00731     WOLFSENTRY_CONFIG_FLAG_ERROR_STRINGS = (1U « 5U),
00732     WOLFSENTRY_CONFIG_FLAG_PROTOCOL_NAMES = (1U « 6U),
00733     WOLFSENTRY_CONFIG_FLAG_NO_STDIO_STREAMS = (1U « 7U),
00734     WOLFSENTRY_CONFIG_FLAG_NO_JSON = (1U « 8U),
00735     WOLFSENTRY_CONFIG_FLAG_HAVE_JSON_DOM = (1U « 9U),
00736     WOLFSENTRY_CONFIG_FLAG_DEBUG_CALL_TRACE = (1U « 10U),
00737     WOLFSENTRY_CONFIG_FLAG_LWIP = (1U « 11U),
00738     WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS = (1U « 12U),
00739     WOLFSENTRY_CONFIG_FLAG_ADDR_BITMASKS = (1U « 13U),
00740     WOLFSENTRY_CONFIG_FLAG_MAX = WOLFSENTRY_CONFIG_FLAG_ADDR_BITMASKS,
00741     WOLFSENTRY_CONFIG_FLAG_ENDIANNESS_ZERO = (0U « 31U)
00742 };
00743
00747 struct wolfsentry_build_settings {
00748     uint32_t version;
00750     uint32_t config;
00752 };
00753
00754 #if !defined(BUILDING_LIBWOLFSENTRY) || defined(WOLFSENTRY_DEFINE_BUILD_SETTINGS)
00755
00758 #ifdef WOLFSENTRY_USER_DEFINED_TYPES
00759     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_USER_DEFINED_TYPES WOLFSENTRY_CONFIG_FLAG_USER_DEFINED_TYPES
00760 #else
00761     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_USER_DEFINED_TYPES 0
00762 #endif
00763
```

```
00764 #ifdef WOLFSENTRY_THREADSAFE
00765     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_THREADSAFE WOLFSENTRY_CONFIG_FLAG_THREADSAFE
00766 #else
00767     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_THREADSAFE 0
00768 #endif
00769
00770 #ifdef WOLFSENTRY_CLOCK_BUILTINS
00771     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_CLOCK_BUILTINS WOLFSENTRY_CONFIG_FLAG_CLOCK_BUILTINS
00772 #else
00773     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_CLOCK_BUILTINS 0
00774 #endif
00775
00776 #ifdef WOLFSENTRY_MALLOC_BUILTINS
00777     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_MALLOC_BUILTINS WOLFSENTRY_CONFIG_FLAG_MALLOC_BUILTINS
00778 #else
00779     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_MALLOC_BUILTINS 0
00780 #endif
00781
00782 #ifdef WOLFSENTRY_ERROR_STRINGS
00783     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_ERROR_STRINGS WOLFSENTRY_CONFIG_FLAG_ERROR_STRINGS
00784 #else
00785     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_ERROR_STRINGS 0
00786 #endif
00787
00788 #ifdef WOLFSENTRY_PROTOCOL_NAMES
00789     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_PROTOCOL_NAMES WOLFSENTRY_CONFIG_FLAG_PROTOCOL_NAMES
00790 #else
00791     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_PROTOCOL_NAMES 0
00792 #endif
00793
00794 #ifdef WOLFSENTRY_NO_STDIO_STREAMS
00795     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_STDIO_STREAMS WOLFSENTRY_CONFIG_FLAG_NO_STDIO_STREAMS
00796 #else
00797     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_STDIO_STREAMS 0
00798 #endif
00799
00800 #ifdef WOLFSENTRY_NO_JSON
00801     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_JSON WOLFSENTRY_CONFIG_FLAG_NO_JSON
00802 #else
00803     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_JSON 0
00804 #endif
00805
00806 #ifdef WOLFSENTRY_HAVE_JSON_DOM
00807     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_HAVE_JSON_DOM WOLFSENTRY_CONFIG_FLAG_HAVE_JSON_DOM
00808 #else
00809     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_HAVE_JSON_DOM 0
00810 #endif
00811
00812 #ifdef WOLFSENTRY_DEBUG_CALL_TRACE
00813     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_DEBUG_CALL_TRACE WOLFSENTRY_CONFIG_FLAG_DEBUG_CALL_TRACE
00814 #else
00815     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_DEBUG_CALL_TRACE 0
00816 #endif
00817
00818 #ifdef WOLFSENTRY_LWIP
00819     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_LWIP WOLFSENTRY_CONFIG_FLAG_LWIP
00820 #else
00821     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_LWIP 0
00822 #endif
00823
00824 /* with compilers that can't evaluate the below expression as a compile-time
00825  * constant, WOLFSENTRY_SHORT_ENUMS can be defined in user settings to 0 or
00826  * 1 to avoid the dependency.
00827  */
00828 #ifdef WOLFSENTRY_SHORT_ENUMS
00829 #if WOLFSENTRY_SHORT_ENUMS == 0
00830     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_SHORT_ENUMS 0
00831 #else
00832     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_SHORT_ENUMS WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS
00833 #endif
00834 #else
00835     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_SHORT_ENUMS ((sizeof(wolfsentry_init_flags_t) < sizeof(int))
      ? WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS : 0)
00836 #endif
00837
00838 #ifdef WOLFSENTRY_ADDR_BITMASK_MATCHING
00839     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_ADDR_BITMASKS WOLFSENTRY_CONFIG_FLAG_ADDR_BITMASKS
00840 #else
00841     #define _WOLFSENTRY_CONFIG_FLAG_VALUE_ADDR_BITMASKS 0
00842 #endif
00843
00847 #define WOLFSENTRY_CONFIG_SIGNATURE ( \
00848     WOLFSENTRY_CONFIG_FLAG_ENDIANNESS_ONE | \
00849     _WOLFSENTRY_CONFIG_FLAG_VALUE_USER_DEFINED_TYPES | \
00850     _WOLFSENTRY_CONFIG_FLAG_VALUE_THREADSAFE | \
00851     _WOLFSENTRY_CONFIG_FLAG_VALUE_CLOCK_BUILTINS | \
00852     _WOLFSENTRY_CONFIG_FLAG_VALUE_MALLOC_BUILTINS | \
```

```
00853     _WOLFSENTRY_CONFIG_FLAG_VALUE_ERROR_STRINGS | \
00854     _WOLFSENTRY_CONFIG_FLAG_VALUE_PROTOCOL_NAMES | \
00855     _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_STDIO_STREAMS | \
00856     _WOLFSENTRY_CONFIG_FLAG_VALUE_NO_JSON | \
00857     _WOLFSENTRY_CONFIG_FLAG_VALUE_HAVE_JSON_DOM | \
00858     _WOLFSENTRY_CONFIG_FLAG_VALUE_DEBUG_CALL_TRACE | \
00859     _WOLFSENTRY_CONFIG_FLAG_VALUE_LWIP | \
00860     _WOLFSENTRY_CONFIG_FLAG_VALUE_SHORT_ENUMS | \
00861     _WOLFSENTRY_CONFIG_FLAG_VALUE_ADDR_BITMASKS)
00862
00863 static __attribute_maybe_unused__ struct wolfsentry_build_settings wolfsentry_build_settings = {
00864 #ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00865     .version =
00866 #endif
00867     WOLFSENTRY_VERSION,
00868 #ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00869     .config =
00870 #endif
00871     WOLFSENTRY_CONFIG_SIGNATURE
00872 };
00875 #endif /* !BUILDING_LIBWOLFSENTRY || WOLFSENTRY_DEFINE_BUILD_SETTINGS */
00876
00879 #endif /* WOLFSENTRY_SETTINGS_H */
```

10.16 wolfsentry/wolfsentry_util.h File Reference

Utility and convenience macros for both internal and application use.

Macros

- #define **offsetof**(structure, element)

  Evaluates to the byte offset of `element` in `structure`.
- #define **sizeof_field**(structure, element)

  Evaluates to the size in bytes of `element` in `structure`.
- #define **instance_of_field**(structure, element)

  Evaluates to a dummy instance of `element` in `structure`, e.g. to be passed to MAX_UINT_OF().
- #define **container_of**(ptr, container_type, member_name)

  Evaluates to a pointer to the struct of type `container_type` within which `ptr` points to the member named `member_name`.
- #define **length_of_array**(x)

  Evaluates to the number of elements in `x`, which must be an array.
- #define **end_ptr_of_array**(x)

  Evaluates to a pointer to the byte immediately following the end of array `x`.
- #define **popcount32**(x)

  Evaluates to the number of set bits in `x`.
- #define **LOG2_32**(x)

  Evaluates to the floor of the base 2 logarithm of `x`, which must be a 32 bit integer.
- #define **LOG2_64**(x)

  Evaluates to the floor of the base 2 logarithm of `x`, which must be a 64 bit integer.
- #define **streq**(vs, fs, vs_len)

  Evaluates to true iff string `vs` of length `vs_len` (not including a terminating null, if any) equals null-terminated string `fs`.
- #define **strcaseeq**(vs, fs, vs_len)

  Evaluates to true iff string `vs` of length `vs_len` (not including a terminating null, if any) equals null-terminated string `fs`, neglecting case distinctions.
- #define **WOLFSENTRY_BYTE_STREAM_DECLARE_STACK**(buf, bufsiz)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_DECLARE_HEAP**(buf, bufsiz)

Byte stream helper macro.

- #define **WOLFSENTRY_BYTE_STREAM_INIT_HEAP**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_FREE_HEAP**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_RESET**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_LEN**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_HEAD**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_PTR**(buf)

  Byte stream helper macro.
- #define **WOLFSENTRY_BYTE_STREAM_SPC**(buf)

  Byte stream helper macro.
- #define **MAX_UINT_OF**(x)

  Evaluates to the largest representable `unsigned int` in a word the size of `x`.
- #define **MAX_SINT_OF**(x)

  Evaluates to the largest representable `signed int` in a word the size of `x`.
- #define **WOLFSENTRY_SET_BITS**(enumint, bits)

  Sets the designated `bits` in `enumint`.
- #define **WOLFSENTRY_CHECK_BITS**(enumint, bits)

  Evaluates to true if `bits` are all set in `enumint`.
- #define **WOLFSENTRY_CLEAR_BITS**(enumint, bits)

  Clears the designated `bits` in `enumint`.
- #define **WOLFSENTRY_MASKIN_BITS**(enumint, bits)

  Evaluates to the bits that are set in both `enumint` and `bits`.
- #define **WOLFSENTRY_MASKOUT_BITS**(enumint, bits)

  Evaluates to the bits that are set `enumint` but not set in `bits`.
- #define **WOLFSENTRY_CLEAR_ALL_BITS**(enumint)

  Clears all bits in `enumint`.
- #define **BITS_PER_BYTE** 8
- #define **WOLFSENTRY_BITS_TO_BYTES**(x)

  Evaluates to the number of bytes needed to represent `x` bits.
- #define **WOLFSENTRY_ATOMIC_INCREMENT**(i, x)

  Adds `x` to `i` thread-safely, returning the sum.
- #define **WOLFSENTRY_ATOMIC_DECREMENT**(i, x)

  Subtracts `x` from `i` thread-safely, returning the difference.
- #define **WOLFSENTRY_ATOMIC_POSTINCREMENT**(i, x)

  Adds `x` to `i` thread-safely, returning the operand `i`.
- #define **WOLFSENTRY_ATOMIC_POSTDECREMENT**(i, x)

  Subtracts `x` from `i` thread-safely, returning the operand `i`.
- #define **WOLFSENTRY_ATOMIC_STORE**(i, x)

  Sets `i` to `x`, subject to benign races from other threads.
- #define **WOLFSENTRY_ATOMIC_LOAD**(i)

  Returns the value of `i`, subject to benign races from other threads.
- #define **WOLFSENTRY_ATOMIC_CMPXCHG**(ptr, expected, desired, weak_p, success_memorder, failure↩ _memorder)

  Sets `*ptr` to `desired` and returns true iff `*ptr` has the value `*expected`, otherwise sets `*expected` to the actual value of `*ptr` and returns false.
- #define **WOLFSENTRY_ATOMIC_INCREMENT_BY_ONE**(i)

Adds 1 to `i` thread-safely, returning the sum.

- #define **WOLFSENTRY_ATOMIC_DECREMENT_BY_ONE**(i)

  Subtracts 1 from `i` thread-safely, returning the difference.

- #define **WOLFSENTRY_ATOMIC_TEST_AND_SET**(i, expected, intended)

  Sets `i` to `intended` and returns true iff `i` has the value `expected`, otherwise sets `expected` to the actual value of `i` and returns false.

- #define **WOLFSENTRY_ATOMIC_UPDATE_FLAGS**(i, set_i, clear_i, pre_i, post_i)

  Sets bits `set_i` in `i`, clears bits `clear_i` in `i`, and sets `pre_i` to the value of `i` before any changes, and `post_i` to the value of `i` after changes.

- #define **WOLFSENTRY_ATOMIC_RESET**(i, pre_i)

  Clears all bits in `i`, saving the previous value of `i` in `pre_i`.

- #define **WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY**(i, x, out)

  Adds `x` to unsigned integer `i`, guarding against overflow, saving the sum to `out`. If overflow would occur, error is indicated by saving `0` to `out`, and `i` is left unchanged.

- #define **WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY_BY_ONE**(i, out)

  Increments unsigned integer `i` by one, guarding against overflow, saving the result to `out`. If overflow would occur, error is indicated by saving `0` to `out`, and `i` is left unchanged.

- #define **WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY**(i, x, out)

  Subtracts `x` from unsigned integer `i`, guarding against underflow, saving the difference to `out`. If underflow would occur, error is indicated by saving a max-value integer (all-1s) to `out`, and `i` is left unchanged.

- #define **WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY_BY_ONE**(i, out)

  Decrements unsigned integer `i` by 1, guarding against underflow, saving the difference to `out`. If underflow would occur, error is indicated by saving a max-value integer (all-1s) to `out`, and `i` is left unchanged.

10.16.1 Detailed Description

Utility and convenience macros for both internal and application use.

Included by `wolfsentry.h`.

10.17 wolfsentry_util.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfsentry_util.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_UTIL_H
00030 #define WOLFSENTRY_UTIL_H
00031
00032 #ifndef offsetof
00033 /* gcc and clang define this in stddef.h to use sanitizer-safe builtins. */
00034 #define offsetof(structure, element) ((uintptr_t)&(((structure *)0)->element))
```

```
00036 #endif
00037 #ifndef sizeof_field
00038 #define sizeof_field(structure, element) sizeof(((structure *)0)->element)
00040 #endif
00041 #ifndef instance_of_field
00042 #define instance_of_field(structure, element) (((structure *)0)->element)
00044 #endif
00045 #ifndef container_of
00046 #define container_of(ptr, container_type, member_name) ((container_type *)(void *)(((byte *)(ptr)) -
      offsetof(container_type, member_name)))
00048 #endif
00049 #ifndef length_of_array
00050 #define length_of_array(x) (sizeof (x) / sizeof (x)[0])
00052 #endif
00053 #ifndef end_ptr_of_array
00054 #define end_ptr_of_array(x) (&(x)[length_of_array(x)])
00056 #endif
00057
00058 #ifndef popcount32
00059 #ifdef __GNUC__
00060 #define popcount32(x) __builtin_popcount(x)
00062 #else
00063 #error Must supply binding for popcount32() on non-__GNUC__ targets.
00064 #endif
00065 #endif
00066
00067 #if defined(__GNUC__) && !defined(WOLFSENTRY_NO_BUILTIN_CLZ)
00068 #ifndef LOG2_32
00069 #define LOG2_32(x) (31 - __builtin_clz((unsigned int)(x)))
00071 #endif
00072 #ifndef LOG2_64
00073 #define LOG2_64(x) ((sizeof(unsigned long long) * 8ULL) - (unsigned long
      long)__builtin_clzll((unsigned long long)(x)) - 1ULL)
00075 #endif
00076 #endif
00077
00078 #define streq(vs,fs,vs_len) (((vs_len) == strlen(fs)) && (memcmp(vs,fs,vs_len) == 0))
00080 #define strcaseeq(vs,fs,vs_len) (((vs_len) == strlen(fs)) && (strncasecmp(vs,fs,vs_len) == 0))
00083 #define WOLFSENTRY_BYTE_STREAM_DECLARE_STACK(buf, bufsiz) static const size_t buf ## siz = (bufsiz);
      unsigned char (buf)[bufsiz], *buf ## _p; size_t buf ## spc
00085 #define WOLFSENTRY_BYTE_STREAM_DECLARE_HEAP(buf, bufsiz) static const size_t buf ## siz = (bufsiz);
      unsigned char *(buf), *buf ## _p; size_t buf ## spc
00087 #define WOLFSENTRY_BYTE_STREAM_INIT_HEAP(buf) ((buf) = (unsigned char *)WOLFSENTRY_MALLOC(buf ## siz))
00089 #define WOLFSENTRY_BYTE_STREAM_FREE_HEAP(buf) WOLFSENTRY_FREE(buf)
00091 #define WOLFSENTRY_BYTE_STREAM_RESET(buf) do { (buf ## _p) = (buf); (buf ## spc) = (buf ## siz); }
      while (0)
00093 #define WOLFSENTRY_BYTE_STREAM_LEN(buf) ((buf ## siz) - (buf ## spc))
00095 #define WOLFSENTRY_BYTE_STREAM_HEAD(buf) (buf)
00097 #define WOLFSENTRY_BYTE_STREAM_PTR(buf) (&(buf ## _p))
00099 #define WOLFSENTRY_BYTE_STREAM_SPC(buf) (&(buf ## spc))
00102 #define MAX_UINT_OF(x) ((((uint64_t)1 « ((sizeof(x) * (uint64_t)BITS_PER_BYTE) - (uint64_t)1)) -
      (uint64_t)1) | ((uint64_t)1 « ((sizeof(x) * (uint64_t)BITS_PER_BYTE) - (uint64_t)1)))
00104 #define MAX_SINT_OF(x) ((int64_t)((((uint64_t)1 « ((sizeof(x) * (uint64_t)BITS_PER_BYTE) -
      (uint64_t)2)) - (uint64_t)1) | ((uint64_t)1 « ((sizeof(x) * (uint64_t)BITS_PER_BYTE) - (uint64_t)2))))
00107 #define WOLFSENTRY_SET_BITS(enumint, bits) ((enumint) |= (bits))
00109 #define WOLFSENTRY_CHECK_BITS(enumint, bits) (((enumint) & (bits)) == (bits))
00111 #define WOLFSENTRY_CLEAR_BITS(enumint, bits) ((enumint) &= ~(uint32_t)(bits))
00113 #define WOLFSENTRY_MASKIN_BITS(enumint, bits) ((enumint) & (bits))
00115 #define WOLFSENTRY_MASKOUT_BITS(enumint, bits) ((enumint) & ~(uint32_t)(bits))
00117 #define WOLFSENTRY_CLEAR_ALL_BITS(enumint) ((enumint) = 0)
00120 #ifndef BITS_PER_BYTE
00121 #define BITS_PER_BYTE 8
00122 #endif
00123
00124 #define WOLFSENTRY_BITS_TO_BYTES(x) (((x) + 7U) » 3U)
00127 /* helpers for stringifying the expanded value of a macro argument rather than its literal text: */
00129 #define _qq(x) #x
00130 #define _q(x) _qq(x)
00133 #ifdef WOLFSENTRY_THREADSAFE
00134
00135 #ifdef WOLFSENTRY_HAVE_GNU_ATOMICS
00136
00137 #define WOLFSENTRY_ATOMIC_INCREMENT(i, x) __atomic_add_fetch(&(i),x,__ATOMIC_SEQ_CST)
00139 #define WOLFSENTRY_ATOMIC_DECREMENT(i, x) __atomic_sub_fetch(&(i),x,__ATOMIC_SEQ_CST)
00141 #define WOLFSENTRY_ATOMIC_POSTINCREMENT(i, x) __atomic_fetch_add(&(i),x,__ATOMIC_SEQ_CST)
00143 #define WOLFSENTRY_ATOMIC_POSTDECREMENT(i, x) __atomic_fetch_sub(&(i),x,__ATOMIC_SEQ_CST)
00145 #define WOLFSENTRY_ATOMIC_STORE(i, x) __atomic_store_n(&(i), x, __ATOMIC_RELEASE)
00147 #define WOLFSENTRY_ATOMIC_LOAD(i) __atomic_load_n(&(i), __ATOMIC_CONSUME)
00149 #define WOLFSENTRY_ATOMIC_CMPXCHG(ptr, expected, desired, weak_p, success_memorder, failure_memorder)
      __atomic_compare_exchange_n(ptr, expected, desired, weak_p, success_memorder, failure_memorder)
00152 #else
00153
00154 #if !defined(WOLFSENTRY_ATOMIC_INCREMENT) || !defined(WOLFSENTRY_ATOMIC_DECREMENT) || \
00155     !defined(WOLFSENTRY_ATOMIC_POSTINCREMENT) || !defined(WOLFSENTRY_ATOMIC_POSTDECREMENT) || \
00156     !defined(WOLFSENTRY_ATOMIC_STORE) || !defined(WOLFSENTRY_ATOMIC_LOAD) || \
00157     !defined(WOLFSENTRY_ATOMIC_CMPXCHG)
00158 #error Missing required atomic implementation(s)
```

```
00159 #endif
00160
00161 #endif /* WOLFSENTRY_HAVE_GNU_ATOMICS */
00162
00163 #define WOLFSENTRY_ATOMIC_INCREMENT_BY_ONE(i) WOLFSENTRY_ATOMIC_INCREMENT(i, 1)
00165 #define WOLFSENTRY_ATOMIC_DECREMENT_BY_ONE(i) WOLFSENTRY_ATOMIC_DECREMENT(i, 1)
00168 /* caution, _TEST_AND_SET() alters arg2 (and returns false) on failure. */
00169 #define WOLFSENTRY_ATOMIC_TEST_AND_SET(i, expected, intended)           \
00170     WOLFSENTRY_ATOMIC_CMPXCHG(                                          \
00171         &(i),                                                           \
00172         &(expected),                                                    \
00173         intended,                                                       \
00174         0 /* weak */,                                                  \
00175         __ATOMIC_SEQ_CST /* success_memmodel */,                       \
00176         __ATOMIC_SEQ_CST /* failure_memmodel */);
00179 #define WOLFSENTRY_ATOMIC_UPDATE_FLAGS(i, set_i, clear_i, pre_i, post_i)\
00180 do {                                                                    \
00181     *(pre_i) = (i);                                                     \
00182     for (;;) {                                                          \
00183         *(post_i) = (*(pre_i) | (set_i)) & ~(clear_i);                  \
00184         if (*(post_i) == *(pre_i))                                      \
00185             break;                                                      \
00186         if (WOLFSENTRY_ATOMIC_CMPXCHG(                                  \
00187                 &(i),                                                   \
00188                 (pre_i),                                                \
00189                 *(post_i),                                              \
00190                 0 /* weak */,                                          \
00191                 __ATOMIC_SEQ_CST /* success_memmodel */,               \
00192                 __ATOMIC_SEQ_CST /* failure_memmodel */))               \
00193             break;                                                      \
00194     }                                                                   \
00195 } while (0)
00198 #define WOLFSENTRY_ATOMIC_RESET(i, pre_i)                               \
00199 do {                                                                    \
00200     *(pre_i) = (i);                                                     \
00201     for (;;) {                                                          \
00202         if (*(pre_i) == 0)                                              \
00203             break;                                                      \
00204         if (WOLFSENTRY_ATOMIC_CMPXCHG(                                  \
00205                 &(i),                                                   \
00206                 (pre_i),                                                \
00207                 0,                                                      \
00208                 0 /* weak */,                                          \
00209                 __ATOMIC_SEQ_CST /* success_memmodel */,               \
00210                 __ATOMIC_SEQ_CST /* failure_memmodel */))               \
00211             break;                                                      \
00212     }                                                                   \
00213 } while (0)
00216 #define WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY(i, x, out)          \
00217 do {                                                                    \
00218     __typeof__(i) _pre_i = (i);                                         \
00219     __typeof__(i) _post_i = _pre_i;                                     \
00220     for (;;) {                                                          \
00221         if (MAX_UINT_OF(i) - _pre_i < (x)) {                            \
00222             _post_i = 0;                                                \
00223             break;                                                      \
00224         }                                                               \
00225         _post_i = (__typeof__(i))(_pre_i + (x));                        \
00226         if (_post_i == _pre_i)                                          \
00227             break;                                                      \
00228         if (WOLFSENTRY_ATOMIC_CMPXCHG(                                  \
00229                 &(i),                                                   \
00230                 &_pre_i,                                                \
00231                 _post_i,                                                \
00232                 0 /* weak */,                                          \
00233                 __ATOMIC_SEQ_CST /* success_memmodel */,               \
00234                 __ATOMIC_SEQ_CST /* failure_memmodel */))               \
00235             break;                                                      \
00236     }                                                                   \
00237     (out) = _post_i;                                                    \
00238 } while(0)
00241 #define WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY_BY_ONE(i, out)      \
00242     WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY(i, 1U, out)
00245 #define WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY(i, x, out)          \
00246 do {                                                                    \
00247     __typeof__(i) _pre_i = (i);                                         \
00248     __typeof__(i) _post_i = _pre_i;                                     \
00249     for (;;) {                                                          \
00250         if (_pre_i < (x)) {                                             \
00251             _post_i = MAX_UINT_OF(i);                                   \
00252             break;                                                      \
00253         }                                                               \
00254         _post_i = (__typeof__(i))(_pre_i - (x));                        \
00255         if (_post_i == _pre_i)                                          \
00256             break;                                                      \
00257         if (WOLFSENTRY_ATOMIC_CMPXCHG  (                                \
00258                 &(i),                                                   \
```

```
00259                 &_pre_i,                                                  \
00260                 _post_i,                                                  \
00261                 0 /* weak */,                                            \
00262                 __ATOMIC_SEQ_CST /* success_memmodel */,                  \
00263                 __ATOMIC_SEQ_CST /* failure_memmodel */))                 \
00264             break;                                                        \
00265     }                                                                     \
00266     (out) = _post_i;                                                      \
00267 } while(0)
00270 #define WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY_BY_ONE(i, out)       \
00271     WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY(i, 1U, out)
00274 #else /* !WOLFSENTRY_THREADSAFE */
00275
00276 #define WOLFSENTRY_ATOMIC_INCREMENT(i, x) ((i) += (x))
00277 #define WOLFSENTRY_ATOMIC_INCREMENT_BY_ONE(i) (++(i))
00278 #define WOLFSENTRY_ATOMIC_DECREMENT(i, x) ((i) -= (x))
00279 #define WOLFSENTRY_ATOMIC_DECREMENT_BY_ONE(i) (--(i))
00280 #define WOLFSENTRY_ATOMIC_STORE(i, x) ((i)=(x))
00281 #define WOLFSENTRY_ATOMIC_LOAD(i) (i)
00282
00283 #define WOLFSENTRY_ATOMIC_UPDATE_FLAGS(i, set_i, clear_i, pre_i, post_i)\
00284 do {                                                                      \
00285     *(pre_i) = (i);                                                       \
00286     *(post_i) = (*(pre_i) | (set_i)) & ~(clear_i);                        \
00287     if (*(post_i) != *(pre_i))                                            \
00288         (i) = *(post_i);                                                  \
00289 } while (0)
00290
00291 #define WOLFSENTRY_ATOMIC_RESET(i, pre_i) do { *(pre_i) = (i); (i) = 0; } while (0)
00292
00293 #define WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY(i, x, out)           \
00294     do {                                                                  \
00295         if (((x) > MAX_UINT_OF(i)) || ((MAX_UINT_OF(i) - (i) < (x))))    \
00296             (out) = 0U;                                                   \
00297         else                                                              \
00298             (out) = (i) = (__typeof__(i))((i) + (x));                     \
00299     } while (0)
00300
00301 #define WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY_BY_ONE(i, out)       \
00302     WOLFSENTRY_ATOMIC_INCREMENT_UNSIGNED_SAFELY(i, 1U, out)
00303
00304 #define WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY(i, x, out)           \
00305     do {                                                                  \
00306         if (((x) > MAX_UINT_OF(i)) || ((i) < (x)))                        \
00307             (out) = MAX_UINT_OF(i);                                       \
00308         else                                                              \
00309             (out) = (i) = (__typeof__(i))((i) - (x));                     \
00310     } while (0)
00311
00312 #define WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY_BY_ONE(i, out)       \
00313     WOLFSENTRY_ATOMIC_DECREMENT_UNSIGNED_SAFELY(i, 1U, out)
00314
00315 #endif /* WOLFSENTRY_THREADSAFE */
00316
00317 #endif /* WOLFSENTRY_UTIL_H */
```

10.18 wolfsentry/wolfssl_test.h File Reference

Macros and helper functions for wolfSSL –enable-wolfsentry.

```
#include <wolfsentry/wolfsentry_util.h>
#include <wolfsentry/wolfsentry_json.h>
```

Data Structures

- struct wolfsentry_data

Macros

- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX**(x) (x)
- #define **WOLFSENTRY_CONTEXT_ARGS_OUT_EX4**(x, y) (x)
- #define **tcp_connect**(sockfd, ip, port, udp, sctp, ssl) tcp_connect_with_wolfSentry(sockfd, ip, port, udp, sctp, ssl, wolfsentry)

10.18.1 Detailed Description

Macros and helper functions for wolfSSL –enable-wolfsentry.

This file is included by wolfssl/test.h when defined(WOLFSSL_WOLFSENTRY_HOOKS).

10.19 wolfssl_test.h

Go to the documentation of this file.

```
00001 /*
00002  * wolfssl_test.h
00003  *
00004  * Copyright (C) 2021-2023 wolfSSL Inc.
00005  *
00006  * This file is part of wolfSentry.
00007  *
00008  * wolfSentry is free software; you can redistribute it and/or modify
00009  * it under the terms of the GNU General Public License as published by
00010  * the Free Software Foundation; either version 2 of the License, or
00011  * (at your option) any later version.
00012  *
00013  * wolfSentry is distributed in the hope that it will be useful,
00014  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00015  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016  * GNU General Public License for more details.
00017  *
00018  * You should have received a copy of the GNU General Public License
00019  * along with this program; if not, write to the Free Software
00020  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
00021  */
00022
00029 #ifndef WOLFSENTRY_WOLFSSL_TEST_H
00030 #define WOLFSENTRY_WOLFSSL_TEST_H
00031
00032 #include <wolfsentry/wolfsentry_util.h>
00033
00034 #if !defined(NO_FILESYSTEM) && !defined(WOLFSENTRY_NO_JSON)
00035 #include <wolfsentry/wolfsentry_json.h>
00036 #endif
00037
00038 #if defined(WOLFSENTRY_VERSION_GE)
00039 #if WOLFSENTRY_VERSION_GE(0, 8, 0)
00040 #define HAVE_WOLFSENTRY_API_0v8
00041 #endif
00042 #endif
00043
00044 #ifndef HAVE_WOLFSENTRY_API_0v8
00045 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX(x) (x)
00046 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(x, y) (x)
00047 #endif
00048
00049 struct wolfsentry_data {
00050     WOLFSENTRY_SOCKADDR(128) remote;
00051     WOLFSENTRY_SOCKADDR(128) local;
00052     wolfsentry_route_flags_t flags;
00053     void *heap;
00054     int alloctype;
00055 };
00056
00057 static void free_wolfsentry_data(struct wolfsentry_data *data) {
00058     XFREE(data, data->heap, data->alloctype);
00059 }
00060
00061 static struct wolfsentry_context *wolfsentry = NULL;
00062
00063 static int wolfsentry_data_index = -1;
00064
00065 static WC_INLINE int wolfsentry_store_endpoints(
00066     WOLFSSL *ssl,
00067     SOCKADDR_IN_T *remote,
00068     SOCKADDR_IN_T *local,
00069     int proto,
00070     wolfsentry_route_flags_t flags,
00071     struct wolfsentry_data **wolfsentry_data_out)
00072 {
00073     struct wolfsentry_data *wolfsentry_data = (struct wolfsentry_data *)XMALLOC(
00074         sizeof *wolfsentry_data, NULL, DYNAMIC_TYPE_SOCKADDR);
00075     if (wolfsentry_data == NULL)
```

```
00076         return WOLFSSL_FAILURE;
00077
00078     wolfsentry_data->heap = NULL;
00079     wolfsentry_data->alloctype = DYNAMIC_TYPE_SOCKADDR;
00080
00081 #ifdef TEST_IPV6
00082     if ((sizeof wolfsentry_data->remote.addr < sizeof remote->sin6_addr) ||
00083         (sizeof wolfsentry_data->local.addr < sizeof local->sin6_addr))
00084         return WOLFSSL_FAILURE;
00085     wolfsentry_data->remote.sa_family = wolfsentry_data->local.sa_family = remote->sin6_family;
00086     wolfsentry_data->remote.sa_port = ntohs(remote->sin6_port);
00087     wolfsentry_data->local.sa_port = ntohs(local->sin6_port);
00088     if (WOLFSENTRY_MASKIN_BITS(flags, WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD)) {
00089         wolfsentry_data->remote.addr_len = 0;
00090         XMEMSET(wolfsentry_data->remote.addr, 0, sizeof remote->sin6_addr);
00091     } else {
00092         wolfsentry_data->remote.addr_len = sizeof remote->sin6_addr * BITS_PER_BYTE;
00093         XMEMCPY(wolfsentry_data->remote.addr, &remote->sin6_addr, sizeof remote->sin6_addr);
00094     }
00095     if (WOLFSENTRY_MASKIN_BITS(flags, WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD)) {
00096         wolfsentry_data->local.addr_len = 0;
00097         XMEMSET(wolfsentry_data->local.addr, 0, sizeof local->sin6_addr);
00098     } else {
00099         wolfsentry_data->local.addr_len = sizeof local->sin6_addr * BITS_PER_BYTE;
00100         XMEMCPY(wolfsentry_data->local.addr, &local->sin6_addr, sizeof local->sin6_addr);
00101     }
00102 #else
00103     if ((sizeof wolfsentry_data->remote.addr < sizeof remote->sin_addr) ||
00104         (sizeof wolfsentry_data->local.addr < sizeof local->sin_addr))
00105         return WOLFSSL_FAILURE;
00106     wolfsentry_data->remote.sa_family = wolfsentry_data->local.sa_family = remote->sin_family;
00107     wolfsentry_data->remote.sa_port = ntohs(remote->sin_port);
00108     wolfsentry_data->local.sa_port = ntohs(local->sin_port);
00109     if (WOLFSENTRY_MASKIN_BITS(flags, WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD)) {
00110         wolfsentry_data->remote.addr_len = 0;
00111         XMEMSET(wolfsentry_data->remote.addr, 0, sizeof remote->sin_addr);
00112     } else {
00113         wolfsentry_data->remote.addr_len = sizeof remote->sin_addr * BITS_PER_BYTE;
00114         XMEMCPY(wolfsentry_data->remote.addr, &remote->sin_addr, sizeof remote->sin_addr);
00115     }
00116     if (WOLFSENTRY_MASKIN_BITS(flags, WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD)) {
00117         wolfsentry_data->local.addr_len = 0;
00118         XMEMSET(wolfsentry_data->local.addr, 0, sizeof local->sin_addr);
00119     } else {
00120         wolfsentry_data->local.addr_len = sizeof local->sin_addr * BITS_PER_BYTE;
00121         XMEMCPY(wolfsentry_data->local.addr, &local->sin_addr, sizeof local->sin_addr);
00122     }
00123 #endif
00124     wolfsentry_data->remote.sa_proto = wolfsentry_data->local.sa_proto = proto;
00125     wolfsentry_data->remote.interface = wolfsentry_data->local.interface = 0;
00126     wolfsentry_data->flags = flags;
00127
00128     if (wolfSSL_set_ex_data_with_cleanup(
00129             ssl, wolfsentry_data_index, wolfsentry_data,
00130             (wolfSSL_ex_data_cleanup_routine_t)free_wolfsentry_data) !=
00131         WOLFSSL_SUCCESS) {
00132         free_wolfsentry_data(wolfsentry_data);
00133         return WOLFSSL_FAILURE;
00134     }
00135
00136     if (wolfsentry_data_out != NULL)
00137         *wolfsentry_data_out = wolfsentry_data;
00138
00139     return WOLFSSL_SUCCESS;
00140 }
00141
00142 static int wolfSentry_NetworkFilterCallback(
00143     WOLFSSL *ssl,
00144     struct wolfsentry_context *_wolfsentry,
00145     wolfSSL_netfilter_decision_t *decision)
00146 {
00147     struct wolfsentry_data *data;
00148     char inet_ntop_buf[INET6_ADDRSTRLEN], inet_ntop_buf2[INET6_ADDRSTRLEN];
00149     wolfsentry_errcode_t ret;
00150     wolfsentry_action_res_t action_results;
00151
00152 #if defined(WOLFSENTRY_THREADSAFE) && defined(HAVE_WOLFSENTRY_API_0v8)
00153     WOLFSENTRY_THREAD_HEADER(WOLFSENTRY_THREAD_FLAG_NONE);
00154     if (WOLFSENTRY_THREAD_GET_ERROR < 0) {
00155         fprintf(stderr, "wolfsentry thread init error: "
00156                 WOLFSENTRY_ERROR_FMT "\n",
00157                 WOLFSENTRY_ERROR_FMT_ARGS(WOLFSENTRY_THREAD_GET_ERROR));
00158         return WOLFSSL_FAILURE;
00159     }
00160 #endif /* WOLFSENTRY_THREADSAFE && HAVE_WOLFSENTRY_API_0v8 */
00161
00162     if ((data = wolfSSL_get_ex_data(ssl, wolfsentry_data_index)) == NULL)
```

```
00163         return WOLFSSL_FAILURE;
00164
00165     ret = wolfsentry_route_event_dispatch(
00166         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(_wolfsentry),
00167         (const struct wolfsentry_sockaddr *)&data->remote,
00168         (const struct wolfsentry_sockaddr *)&data->local,
00169         data->flags,
00170         NULL /* event_label */,
00171         0 /* event_label_len */,
00172         NULL /* caller_context */,
00173         NULL /* id */,
00174         NULL /* inexact_matches */,
00175         &action_results);
00176
00177     if (ret >= 0) {
00178         if (WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_REJECT))
00179             *decision = WOLFSSL_NETFILTER_REJECT;
00180         else if (WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_ACCEPT))
00181             *decision = WOLFSSL_NETFILTER_ACCEPT;
00182         else
00183             *decision = WOLFSSL_NETFILTER_PASS;
00184     } else {
00185         fprintf(stderr, "wolfsentry_route_event_dispatch error "
00186                 WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
00187         *decision = WOLFSSL_NETFILTER_PASS;
00188     }
00189
00190     printf("wolfSentry got network filter callback: family=%d proto=%d rport=%d"
00191            " lport=%d raddr=%s laddr=%s interface=%d; decision=%d (%s)\n",
00192            data->remote.sa_family,
00193            data->remote.sa_proto,
00194            data->remote.sa_port,
00195            data->local.sa_port,
00196            inet_ntop(data->remote.sa_family, data->remote.addr, inet_ntop_buf,
00197                      sizeof inet_ntop_buf),
00198            inet_ntop(data->local.sa_family, data->local.addr, inet_ntop_buf2,
00199                      sizeof inet_ntop_buf2),
00200            data->remote.interface,
00201            *decision,
00202            *decision == WOLFSSL_NETFILTER_REJECT ? "REJECT" :
00203            *decision == WOLFSSL_NETFILTER_ACCEPT ? "ACCEPT" :
00204            *decision == WOLFSSL_NETFILTER_PASS ? "PASS" :
00205            "???");
00206
00207 #if defined(WOLFSENTRY_THREADSAFE) && defined(HAVE_WOLFSENTRY_API_0v8)
00208     ret = WOLFSENTRY_THREAD_TAILER(WOLFSENTRY_THREAD_FLAG_NONE);
00209     if (ret < 0) {
00210         fprintf(stderr, "wolfsentry thread exit error: "
00211                 WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
00212     }
00213 #endif
00214
00215     return WOLFSSL_SUCCESS;
00216 }
00217
00218 static int wolfsentry_setup(
00219     struct wolfsentry_context **_wolfsentry,
00220     const char *_wolfsentry_config_path,
00221     wolfsentry_route_flags_t route_flags)
00222 {
00223     wolfsentry_errcode_t ret;
00224
00225 #ifdef HAVE_WOLFSENTRY_API_0v8
00226 #ifdef WOLFSENTRY_THREADSAFE
00227     WOLFSENTRY_THREAD_HEADER(WOLFSENTRY_THREAD_FLAG_NONE);
00228     if (WOLFSENTRY_THREAD_GET_ERROR < 0) {
00229         fprintf(stderr, "wolfsentry thread init error: "
00230                 WOLFSENTRY_ERROR_FMT "\n",
00231                 WOLFSENTRY_ERROR_FMT_ARGS(WOLFSENTRY_THREAD_GET_ERROR));
00232         err_sys("unable to initialize wolfSentry thread context");
00233     }
00234 #endif
00235     ret =  wolfsentry_init(wolfsentry_build_settings,
00236                            WOLFSENTRY_CONTEXT_ARGS_OUT_EX(NULL /* hpi */),
00237                            NULL /* default config */,
00238                            _wolfsentry);
00239 #else
00240     ret =  wolfsentry_init(NULL /* hpi */, NULL /* default config */,
00241                            _wolfsentry);
00242 #endif
00243     if (ret < 0) {
00244         fprintf(stderr, "wolfsentry_init() returned " WOLFSENTRY_ERROR_FMT "\n",
00245                 WOLFSENTRY_ERROR_FMT_ARGS(ret));
00246         err_sys("unable to initialize wolfSentry");
00247     }
00248
00249     if (wolfsentry_data_index < 0)
```

```
00250         wolfsentry_data_index = wolfSSL_get_ex_new_index(0, NULL, NULL, NULL,
00251                                                          NULL);
00252
00253 #if !defined(NO_FILESYSTEM) && !defined(WOLFSENTRY_NO_JSON)
00254     if (_wolfsentry_config_path != NULL) {
00255         unsigned char buf[512];
00256         char err_buf[512];
00257         struct wolfsentry_json_process_state *jps;
00258
00259         FILE *f = fopen(_wolfsentry_config_path, "r");
00260
00261         if (f == NULL) {
00262             fprintf(stderr, "fopen(%s): %s\n",_wolfsentry_config_path,strerror(errno));
00263             err_sys("unable to open wolfSentry config file");
00264         }
00265
00266         if ((ret = wolfsentry_config_json_init(
00267                   WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry),
00268                   WOLFSENTRY_CONFIG_LOAD_FLAG_NONE,
00269                   &jps)) < 0) {
00270             fprintf(stderr, "wolfsentry_config_json_init() returned "
00271                     WOLFSENTRY_ERROR_FMT "\n",
00272                     WOLFSENTRY_ERROR_FMT_ARGS(ret));
00273             err_sys("error while initializing wolfSentry config parser");
00274         }
00275
00276         for (;;) {
00277             size_t n = fread(buf, 1, sizeof buf, f);
00278             if ((n < sizeof buf) && ferror(f)) {
00279                 fprintf(stderr,"fread(%s): %s\n",_wolfsentry_config_path, strerror(errno));
00280                 err_sys("error while reading wolfSentry config file");
00281             }
00282
00283             ret = wolfsentry_config_json_feed(jps, buf, n, err_buf, sizeof err_buf);
00284             if (ret < 0) {
00285                 fprintf(stderr, "%.*s\n", (int)sizeof err_buf, err_buf);
00286                 err_sys("error while loading wolfSentry config file");
00287             }
00288             if ((n < sizeof buf) && feof(f))
00289                 break;
00290         }
00291         fclose(f);
00292
00293         if ((ret = wolfsentry_config_json_fini(&jps, err_buf, sizeof err_buf)) < 0) {
00294             fprintf(stderr, "%.*s\n", (int)sizeof err_buf, err_buf);
00295             err_sys("error while loading wolfSentry config file");
00296         }
00297
00298     } else
00299 #endif /* !NO_FILESYSTEM && !WOLFSENTRY_NO_JSON */
00300     {
00301         struct wolfsentry_route_table *table;
00302
00303 #ifdef WOLFSENTRY_THREADSAFE
00304         ret = WOLFSENTRY_SHARED_EX(*_wolfsentry);
00305         if (ret < 0) {
00306             fprintf(stderr, "wolfsentry shared lock op failed: "
00307                     WOLFSENTRY_ERROR_FMT ".\n",
00308                     WOLFSENTRY_ERROR_FMT_ARGS(ret));
00309             return ret;
00310         }
00311 #endif
00312
00313         if ((ret = wolfsentry_route_get_main_table(
00314                   WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry),
00315                   &table)) < 0)
00316         {
00317             fprintf(stderr, "wolfsentry_route_get_main_table() returned "
00318                     WOLFSENTRY_ERROR_FMT "\n",
00319                     WOLFSENTRY_ERROR_FMT_ARGS(ret));
00320 #ifdef WOLFSENTRY_THREADSAFE
00321             WOLFSENTRY_WARN_ON_FAILURE(
00322                 wolfsentry_context_unlock(
00323                     WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00324 #endif
00325             return ret;
00326         }
00327
00328         if (WOLFSENTRY_MASKIN_BITS(route_flags, WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT)) {
00329             WOLFSENTRY_SOCKADDR(128) remote, local;
00330             wolfsentry_ent_id_t id;
00331             wolfsentry_action_res_t action_results;
00332
00333             if ((ret = wolfsentry_route_table_default_policy_set(
00334                     WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry),
00335                     table,
00336                     WOLFSENTRY_ACTION_RES_ACCEPT))
```

```
00337                 < 0) {
00338                 fprintf(stderr,
00339                         "wolfsentry_route_table_default_policy_set() returned "
00340                         WOLFSENTRY_ERROR_FMT "\n",
00341                         WOLFSENTRY_ERROR_FMT_ARGS(ret));
00342 #ifdef WOLFSENTRY_THREADSAFE
00343                 WOLFSENTRY_WARN_ON_FAILURE(
00344                     wolfsentry_context_unlock(
00345                         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00346 #endif
00347                 return ret;
00348             }
00349
00350             XMEMSET(&remote, 0, sizeof remote);
00351             XMEMSET(&local, 0, sizeof local);
00352 #ifdef TEST_IPV6
00353             remote.sa_family = local.sa_family = AF_INET6;
00354             remote.addr_len = 128;
00355             XMEMCPY(remote.addr, "\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\001",
      16);
00356 #else
00357             remote.sa_family = local.sa_family = AF_INET;
00358             remote.addr_len = 32;
00359             XMEMCPY(remote.addr, "\177\000\000\001", 4);
00360 #endif
00361
00362             if ((ret = wolfsentry_route_insert
00363                  (WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry),
00364                   NULL /* caller_context */,
00365                   (const struct wolfsentry_sockaddr *)&remote,
00366                   (const struct wolfsentry_sockaddr *)&local,
00367                   route_flags                                    |
00368                   WOLFSENTRY_ROUTE_FLAG_GREENLISTED              |
00369                   WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD    |
00370                   WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD|
00371                   WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD |
00372                   WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD   |
00373                   WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD        |
00374                   WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD  |
00375                   WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD,
00376                   0 /* event_label_len */, 0 /* event_label */, &id,
00377                   &action_results)) < 0) {
00378                 fprintf(stderr, "wolfsentry_route_insert() returned "
00379                         WOLFSENTRY_ERROR_FMT "\n",
00380                         WOLFSENTRY_ERROR_FMT_ARGS(ret));
00381 #ifdef WOLFSENTRY_THREADSAFE
00382                 WOLFSENTRY_WARN_ON_FAILURE(
00383                     wolfsentry_context_unlock(
00384                         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00385 #endif
00386                 return ret;
00387             }
00388         } else if (WOLFSENTRY_MASKIN_BITS(route_flags, WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN)) {
00389             WOLFSENTRY_SOCKADDR(128) remote, local;
00390             wolfsentry_ent_id_t id;
00391             wolfsentry_action_res_t action_results;
00392
00393             if ((ret = wolfsentry_route_table_default_policy_set(
00394                       WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry), table,
00395                       WOLFSENTRY_ACTION_RES_REJECT|WOLFSENTRY_ACTION_RES_STOP))
00396                 < 0) {
00397                 fprintf(stderr,
00398                         "wolfsentry_route_table_default_policy_set() returned "
00399                         WOLFSENTRY_ERROR_FMT "\n",
00400                         WOLFSENTRY_ERROR_FMT_ARGS(ret));
00401 #ifdef WOLFSENTRY_THREADSAFE
00402                 WOLFSENTRY_WARN_ON_FAILURE(
00403                     wolfsentry_context_unlock(
00404                         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00405 #endif
00406                 return ret;
00407             }
00408
00409             XMEMSET(&remote, 0, sizeof remote);
00410             XMEMSET(&local, 0, sizeof local);
00411 #ifdef TEST_IPV6
00412             remote.sa_family = local.sa_family = AF_INET6;
00413             remote.addr_len = 128;
00414             XMEMCPY(remote.addr, "\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\001",
      16);
00415 #else
00416             remote.sa_family = local.sa_family = AF_INET;
00417             remote.addr_len = 32;
00418             XMEMCPY(remote.addr, "\177\000\000\001", 4);
00419 #endif
00420
00421             if ((ret = wolfsentry_route_insert
```

```
00422                   (WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry),
00423                    NULL /* caller_context */,
00424                    (const struct wolfsentry_sockaddr *)&remote,
00425                    (const struct wolfsentry_sockaddr *)&local,
00426                    route_flags                                    |
00427                    WOLFSENTRY_ROUTE_FLAG_GREENLISTED              |
00428                    WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD    |
00429                    WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD|
00430                    WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD |
00431                    WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD   |
00432                    WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD        |
00433                    WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD  |
00434                    WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD,
00435                    0 /* event_label_len */, 0 /* event_label */, &id,
00436                    &action_results)) < 0) {
00437                 fprintf(stderr, "wolfsentry_route_insert() returned "
00438                         WOLFSENTRY_ERROR_FMT "\n",
00439                         WOLFSENTRY_ERROR_FMT_ARGS(ret));
00440 #ifdef WOLFSENTRY_THREADSAFE
00441                 WOLFSENTRY_WARN_ON_FAILURE(
00442                     wolfsentry_context_unlock(
00443                         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00444 #endif
00445                 return ret;
00446             }
00447         }
00448 #ifdef WOLFSENTRY_THREADSAFE
00449         WOLFSENTRY_WARN_ON_FAILURE(
00450             wolfsentry_context_unlock(
00451                 WOLFSENTRY_CONTEXT_ARGS_OUT_EX(*_wolfsentry)));
00452 #endif
00453     }
00454
00455 #if defined(WOLFSENTRY_THREADSAFE) && defined(HAVE_WOLFSENTRY_API_0v8)
00456     ret = WOLFSENTRY_THREAD_TAILER(WOLFSENTRY_THREAD_FLAG_NONE);
00457     if (ret < 0) {
00458         fprintf(stderr, "wolfsentry thread exit error: "
00459                 WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
00460     }
00461 #endif
00462
00463     return 0;
00464 }
00465
00466 static WC_INLINE int tcp_connect_with_wolfSentry(
00467     SOCKET_T* sockfd,
00468     const char* ip,
00469     word16 port,
00470     int udp,
00471     int sctp,
00472     WOLFSSL* ssl,
00473     struct wolfsentry_context *_wolfsentry)
00474 {
00475     SOCKADDR_IN_T remote_addr;
00476     struct wolfsentry_data *wolfsentry_data;
00477     char inet_ntop_buf[INET6_ADDRSTRLEN], inet_ntop_buf2[INET6_ADDRSTRLEN];
00478     wolfsentry_errcode_t ret;
00479     wolfsentry_action_res_t action_results;
00480     wolfSSL_netfilter_decision_t decision;
00481
00482 #if defined(WOLFSENTRY_THREADSAFE) && defined(HAVE_WOLFSENTRY_API_0v8)
00483     WOLFSENTRY_THREAD_HEADER(WOLFSENTRY_THREAD_FLAG_NONE);
00484     if (WOLFSENTRY_THREAD_GET_ERROR < 0) {
00485         fprintf(stderr, "wolfsentry thread init error: "
00486                 WOLFSENTRY_ERROR_FMT "\n",
00487                 WOLFSENTRY_ERROR_FMT_ARGS(WOLFSENTRY_THREAD_GET_ERROR));
00488         err_sys("unable to initialize wolfSentry thread context");
00489     }
00490 #endif
00491
00492     build_addr(&remote_addr, ip, port, udp, sctp);
00493
00494     {
00495         SOCKADDR_IN_T local_addr;
00496 #ifdef TEST_IPV6
00497         local_addr.sin6_port = 0;
00498 #else
00499         local_addr.sin_port = 0;
00500 #endif
00501         ((struct sockaddr *)&local_addr)->sa_family = ((struct sockaddr *)&remote_addr)->sa_family;
00502
00503         if (wolfsentry_store_endpoints(
00504                 ssl, &remote_addr, &local_addr,
00505                 udp ? IPPROTO_UDP : IPPROTO_TCP,
00506                 WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT|
00507                 WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD|
00508                 WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD, &wolfsentry_data) != WOLFSSL_SUCCESS)
```

```
00509             return WOLFSSL_FAILURE;
00510     }
00511
00512     ret = wolfsentry_route_event_dispatch(
00513         WOLFSENTRY_CONTEXT_ARGS_OUT_EX(_wolfsentry),
00514         (const struct wolfsentry_sockaddr *)&wolfsentry_data->remote,
00515         (const struct wolfsentry_sockaddr *)&wolfsentry_data->local,
00516         wolfsentry_data->flags,
00517         NULL /* event_label */,
00518         0    /* event_label_len */,
00519         NULL /* caller_context */,
00520         NULL /* id */,
00521         NULL /* inexact_matches */,
00522         &action_results);
00523
00524     if (ret < 0) {
00525         fprintf(stderr, "wolfsentry_route_event_dispatch error "
00526                 WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
00527         decision = WOLFSSL_NETFILTER_PASS;
00528     } else {
00529         if (WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_REJECT))
00530             decision = WOLFSSL_NETFILTER_REJECT;
00531         else if (WOLFSENTRY_MASKIN_BITS(action_results, WOLFSENTRY_ACTION_RES_ACCEPT))
00532             decision = WOLFSSL_NETFILTER_ACCEPT;
00533         else
00534             decision = WOLFSSL_NETFILTER_PASS;
00535     }
00536
00537     printf("wolfSentry callin from tcp_connect_with_wolfSentry: family=%d proto=%d rport=%d"
00538            " lport=%d raddr=%s laddr=%s interface=%d; decision=%d (%s)\n",
00539            wolfsentry_data->remote.sa_family,
00540            wolfsentry_data->remote.sa_proto,
00541            wolfsentry_data->remote.sa_port,
00542            wolfsentry_data->local.sa_port,
00543            inet_ntop(wolfsentry_data->remote.sa_family, wolfsentry_data->remote.addr, inet_ntop_buf,
00544                      sizeof inet_ntop_buf),
00545            inet_ntop(wolfsentry_data->local.sa_family, wolfsentry_data->local.addr, inet_ntop_buf2,
00546                      sizeof inet_ntop_buf2),
00547            wolfsentry_data->remote.interface,
00548            decision,
00549            decision == WOLFSSL_NETFILTER_REJECT ? "REJECT" :
00550            decision == WOLFSSL_NETFILTER_ACCEPT ? "ACCEPT" :
00551            decision == WOLFSSL_NETFILTER_PASS ?   "PASS" :
00552            "???");
00553
00554     if (decision == WOLFSSL_NETFILTER_REJECT)
00555         return SOCKET_FILTERED_E;
00556
00557     if (udp) {
00558         wolfSSL_dtls_set_peer(ssl, &remote_addr, sizeof(remote_addr));
00559     }
00560     tcp_socket(sockfd, udp, sctp);
00561
00562     if (!udp) {
00563         if (connect(*sockfd, (const struct sockaddr*)&remote_addr, sizeof(remote_addr)) != 0)
00564             err_sys_with_errno("tcp connect failed");
00565     }
00566
00567 #if defined(WOLFSENTRY_THREADSAFE) && defined(HAVE_WOLFSENTRY_API_0v8)
00568     ret = WOLFSENTRY_THREAD_TAILER(WOLFSENTRY_THREAD_FLAG_NONE);
00569     if (ret < 0) {
00570         fprintf(stderr, "wolfsentry thread exit error: "
00571                 WOLFSENTRY_ERROR_FMT "\n", WOLFSENTRY_ERROR_FMT_ARGS(ret));
00572     }
00573 #endif
00574
00575     return WOLFSSL_SUCCESS;
00576 }
00577
00578 #define tcp_connect(sockfd, ip, port, udp, sctp, ssl) \
00579     tcp_connect_with_wolfSentry(sockfd, ip, port, udp, sctp, ssl, wolfsentry)
00580
00581 #endif /* !WOLFSENTRY_WOLFSSL_TEST_H */
```

Index