

## wolfSentry Embedded IDPS v1.5.0 API Reference

Generated by Doxygen 1.9.7



|   |           |
|---|-----------|
| <b>1 wolfSentry: the wolfSSL embedded firewall and IDPS</b>                       | <b>1</b>  |
| 1.1 Description   | 1         |
| 1.2 Documentation   | 1         |
| 1.3 Dependencies  | 1         |
| <b>2 Building and Initializing wolfSentry for an application on FreeRTOS/IWIP</b> | <b>3</b>  |
| <b>3 Configuring wolfSentry using a JSON document</b>                             | <b>7</b>  |
| <b>4 wolfSentry Release History and Change Log</b>                                | <b>17</b> |
| <b>5 Module Index</b>   | <b>37</b> |
| 5.1 Modules   | 37        |
| <b>6 Data Structure Index</b>   | <b>39</b> |
| 6.1 Data Structures   | 39        |
| <b>7 File Index</b>   | <b>41</b> |
| 7.1 File List   | 41        |
| <b>8 Module Documentation</b>   | <b>43</b> |
| 8.1 Core Types and Macros   | 43        |
| 8.1.1 Detailed Description  | 44        |
| 8.2 Startup/Configuration/Shutdown Subsystem                                      | 44        |
| 8.2.1 Detailed Description  | 48        |
| 8.2.2 Enumeration Type Documentation  | 48        |
| 8.2.2.1 wolfsentry_clone_flags_t  | 48        |
| 8.2.2.2 wolfsentry_config_load_flags  | 48        |
| 8.2.2.3 wolfsentry_init_flags_t   | 49        |
| 8.2.3 Function Documentation  | 49        |
| 8.2.3.1 wolfsentry_context_clone()  | 49        |
| 8.2.3.2 wolfsentry_context_enable_actions()                                       | 50        |
| 8.2.3.3 wolfsentry_context_exchange()   | 50        |
| 8.2.3.4 wolfsentry_context_flush()  | 51        |
| 8.2.3.5 wolfsentry_context_free()   | 51        |
| 8.2.3.6 wolfsentry_context_inhibit_actions()                                      | 51        |
| 8.2.3.7 wolfsentry_defaultconfig_get()  | 51        |
| 8.2.3.8 wolfsentry_defaultconfig_update()   | 52        |
| 8.2.3.9 wolfsentry_init()   | 52        |
| 8.2.3.10 wolfsentry_shutdown()  | 53        |
| 8.3 Diagnostics, Control Flow Helpers, and Compiler Attribute Helpers             | 53        |
| 8.3.1 Detailed Description  | 58        |
| 8.3.2 Macro Definition Documentation  | 58        |
| 8.3.2.1 WOLFSENTRY_DEBUG_CALL_TRACE   | 58        |
| 8.4 Route/Rule Subsystem  | 58        |

|  |    |
|--|----|
| 8.4.1 Detailed Description                                 | 64 |
| 8.4.2 Enumeration Type Documentation                       | 64 |
| 8.4.2.1 wolfsentry_format_flags_t                          | 64 |
| 8.4.2.2 wolfsentry_route_flags_t                           | 65 |
| 8.4.3 Function Documentation                               | 66 |
| 8.4.3.1 wolfsentry_route_bulk_clear_insert_action_status() | 66 |
| 8.4.3.2 wolfsentry_route_bulk_insert_actions()             | 66 |
| 8.4.3.3 wolfsentry_route_delete()                          | 67 |
| 8.4.3.4 wolfsentry_route_delete_by_id()                    | 67 |
| 8.4.3.5 wolfsentry_route_drop_reference()                  | 68 |
| 8.4.3.6 wolfsentry_route_event_dispatch()                  | 68 |
| 8.4.3.7 wolfsentry_route_export()                          | 69 |
| 8.4.3.8 wolfsentry_route_exports_render()                  | 70 |
| 8.4.3.9 wolfsentry_route_flush_table()                     | 70 |
| 8.4.3.10 wolfsentry_route_get_addrs()                      | 70 |
| 8.4.3.11 wolfsentry_route_get_flags()                      | 71 |
| 8.4.3.12 wolfsentry_route_get_main_table()                 | 71 |
| 8.4.3.13 wolfsentry_route_get_metadata()                   | 72 |
| 8.4.3.14 wolfsentry_route_get_private_data()               | 72 |
| 8.4.3.15 wolfsentry_route_get_reference()                  | 72 |
| 8.4.3.16 wolfsentry_route_insert()                         | 73 |
| 8.4.3.17 wolfsentry_route_parent_event()                   | 74 |
| 8.4.3.18 wolfsentry_route_render()                         | 74 |
| 8.4.3.19 wolfsentry_route_set_wildcard()                   | 75 |
| 8.4.3.20 wolfsentry_route_stale_purge()                    | 75 |
| 8.4.3.21 wolfsentry_route_table_default_policy_get()       | 75 |
| 8.4.3.22 wolfsentry_route_table_default_policy_set()       | 76 |
| 8.4.3.23 wolfsentry_route_table_fallthrough_route_get()    | 76 |
| 8.4.3.24 wolfsentry_route_table_iterate_current()          | 77 |
| 8.4.3.25 wolfsentry_route_table_iterate_end()              | 77 |
| 8.4.3.26 wolfsentry_route_table_iterate_next()             | 78 |
| 8.4.3.27 wolfsentry_route_table_iterate_prev()             | 78 |
| 8.4.3.28 wolfsentry_route_table_iterate_seek_to_head()     | 78 |
| 8.4.3.29 wolfsentry_route_table_iterate_seek_to_tail()     | 79 |
| 8.4.3.30 wolfsentry_route_table_iterate_start()            | 79 |
| 8.4.3.31 wolfsentry_route_update_flags()                   | 80 |
| 8.5 Action Subsystem                                       | 80 |
| 8.5.1 Detailed Description                                 | 82 |
| 8.5.2 Typedef Documentation                                | 82 |
| 8.5.2.1 wolfsentry_action_callback_t                       | 82 |
| 8.5.3 Enumeration Type Documentation                       | 83 |
| 8.5.3.1 wolfsentry_action_flags_t                          | 83 |

|  |     |
|--|-----|
| 8.5.3.2 wolfsentry_action_res_t . . . . .                | 83  |
| 8.5.3.3 wolfsentry_action_type_t . . . . .               | 84  |
| 8.5.4 Function Documentation . . . . .                   | 84  |
| 8.5.4.1 wolfsentry_action_delete() . . . . .             | 84  |
| 8.5.4.2 wolfsentry_action_drop_reference() . . . . .     | 85  |
| 8.5.4.3 wolfsentry_action_flush_all() . . . . .          | 85  |
| 8.5.4.4 wolfsentry_action_get_flags() . . . . .          | 86  |
| 8.5.4.5 wolfsentry_action_get_label() . . . . .          | 86  |
| 8.5.4.6 wolfsentry_action_get_reference() . . . . .      | 86  |
| 8.5.4.7 wolfsentry_action_insert() . . . . .             | 87  |
| 8.5.4.8 wolfsentry_action_update_flags() . . . . .       | 88  |
| 8.6 Event Subsystem . . . . .                            | 88  |
| 8.6.1 Detailed Description . . . . .                     | 90  |
| 8.6.2 Enumeration Type Documentation . . . . .           | 90  |
| 8.6.2.1 wolfsentry_event_flags_t . . . . .               | 90  |
| 8.6.2.2 wolfsentry_eventconfig_flags_t . . . . .         | 90  |
| 8.6.3 Function Documentation . . . . .                   | 91  |
| 8.6.3.1 wolfsentry_event_action_append() . . . . .       | 91  |
| 8.6.3.2 wolfsentry_event_action_delete() . . . . .       | 91  |
| 8.6.3.3 wolfsentry_event_action_insert_after() . . . . . | 92  |
| 8.6.3.4 wolfsentry_event_action_list_done() . . . . .    | 92  |
| 8.6.3.5 wolfsentry_event_action_list_next() . . . . .    | 93  |
| 8.6.3.6 wolfsentry_event_action_list_start() . . . . .   | 93  |
| 8.6.3.7 wolfsentry_event_action_prepend() . . . . .      | 94  |
| 8.6.3.8 wolfsentry_event_delete() . . . . .              | 95  |
| 8.6.3.9 wolfsentry_event_drop_reference() . . . . .      | 95  |
| 8.6.3.10 wolfsentry_event_flush_all() . . . . .          | 95  |
| 8.6.3.11 wolfsentry_event_get_config() . . . . .         | 96  |
| 8.6.3.12 wolfsentry_event_get_flags() . . . . .          | 96  |
| 8.6.3.13 wolfsentry_event_get_label() . . . . .          | 97  |
| 8.6.3.14 wolfsentry_event_get_reference() . . . . .      | 97  |
| 8.6.3.15 wolfsentry_event_insert() . . . . .             | 97  |
| 8.6.3.16 wolfsentry_event_set_aux_event() . . . . .      | 98  |
| 8.6.3.17 wolfsentry_event_update_config() . . . . .      | 98  |
| 8.6.3.18 wolfsentry_eventconfig_check() . . . . .        | 100 |
| 8.6.3.19 wolfsentry_eventconfig_init() . . . . .         | 100 |
| 8.7 Address Family Subsystem . . . . .                   | 101 |
| 8.7.1 Detailed Description . . . . .                     | 104 |
| 8.8 User-Defined Value Subsystem . . . . .               | 104 |
| 8.8.1 Detailed Description . . . . .                     | 108 |
| 8.8.2 Typedef Documentation . . . . .                    | 108 |
| 8.8.2.1 wolfsentry_kv_validator_t . . . . .              | 108 |

|  |     |
|--|-----|
| 8.8.3 Function Documentation                             | 108 |
| 8.8.3.1 wolfsentry_user_value_get_bytes()                | 108 |
| 8.8.3.2 wolfsentry_user_value_get_json()                 | 108 |
| 8.8.3.3 wolfsentry_user_value_get_string()               | 108 |
| 8.9 Object Subsystem                                     | 109 |
| 8.9.1 Detailed Description                               | 109 |
| 8.9.2 Enumeration Type Documentation                     | 109 |
| 8.9.2.1 wolfsentry_object_type_t                         | 109 |
| 8.9.3 Function Documentation                             | 110 |
| 8.9.3.1 wolfsentry_get_object_id()                       | 110 |
| 8.9.3.2 wolfsentry_get_object_type()                     | 110 |
| 8.9.3.3 wolfsentry_table_n_deletes()                     | 111 |
| 8.9.3.4 wolfsentry_table_n_inserts()                     | 111 |
| 8.10 Thread Synchronization Subsystem                    | 111 |
| 8.10.1 Detailed Description                              | 116 |
| 8.10.2 Enumeration Type Documentation                    | 116 |
| 8.10.2.1 wolfsentry_lock_flags_t                         | 116 |
| 8.10.2.2 wolfsentry_thread_flags_t                       | 117 |
| 8.10.3 Function Documentation                            | 117 |
| 8.10.3.1 wolfsentry_lock_alloc()                         | 117 |
| 8.10.3.2 wolfsentry_lock_destroy()                       | 118 |
| 8.10.3.3 wolfsentry_lock_free()                          | 118 |
| 8.10.3.4 wolfsentry_lock_get_flags()                     | 119 |
| 8.10.3.5 wolfsentry_lock_have_either()                   | 119 |
| 8.10.3.6 wolfsentry_lock_have_mutex()                    | 120 |
| 8.10.3.7 wolfsentry_lock_have_shared()                   | 120 |
| 8.10.3.8 wolfsentry_lock_have_shared2mutex_reservation() | 121 |
| 8.10.3.9 wolfsentry_lock_init()                          | 121 |
| 8.10.3.10 wolfsentry_lock_mutex()                        | 122 |
| 8.10.3.11 wolfsentry_lock_mutex2shared()                 | 122 |
| 8.10.3.12 wolfsentry_lock_mutex_abstimed()               | 123 |
| 8.10.3.13 wolfsentry_lock_mutex_timed()                  | 123 |
| 8.10.3.14 wolfsentry_lock_shared()                       | 124 |
| 8.10.3.15 wolfsentry_lock_shared2mutex()                 | 124 |
| 8.10.3.16 wolfsentry_lock_shared2mutex_abandon()         | 125 |
| 8.10.3.17 wolfsentry_lock_shared2mutex_abstimed()        | 125 |
| 8.10.3.18 wolfsentry_lock_shared2mutex_redeem()          | 126 |
| 8.10.3.19 wolfsentry_lock_shared2mutex_redeem_abstimed() | 126 |
| 8.10.3.20 wolfsentry_lock_shared2mutex_redeem_timed()    | 127 |
| 8.10.3.21 wolfsentry_lock_shared2mutex_reserve()         | 127 |
| 8.10.3.22 wolfsentry_lock_shared2mutex_timed()           | 128 |
| 8.10.3.23 wolfsentry_lock_shared_abstimed()              | 128 |

|  |            |
|--|------------|
| 8.10.3.24 wolfsentry_lock_shared_timed()                 | 129        |
| 8.10.3.25 wolfsentry_lock_unlock()                       | 130        |
| 8.11 Allocator (Heap) Functions and Callbacks            | 130        |
| 8.11.1 Detailed Description                              | 131        |
| 8.12 Time Functions and Callbacks                        | 131        |
| 8.12.1 Detailed Description                              | 132        |
| 8.13 Semaphore Function Callbacks                        | 132        |
| 8.13.1 Detailed Description                              | 133        |
| 8.13.2 Typedef Documentation                             | 133        |
| 8.13.2.1 sem_destroy_cb_t                                | 133        |
| 8.13.2.2 sem_init_cb_t                                   | 133        |
| 8.13.2.3 sem_post_cb_t                                   | 133        |
| 8.13.2.4 sem_timedwait_cb_t                              | 133        |
| 8.13.2.5 sem_trywait_cb_t                                | 133        |
| 8.13.2.6 sem_wait_cb_t                                   | 134        |
| 8.14 lwIP Callback Activation Functions                  | 134        |
| 8.14.1 Detailed Description                              | 134        |
| <b>9 Data Structure Documentation</b>                    | <b>135</b> |
| 9.1 JSON_CALLBACKS Struct Reference                      | 135        |
| 9.2 JSON_CONFIG Struct Reference                         | 135        |
| 9.3 JSON_DOM_PARSER Struct Reference                     | 135        |
| 9.4 JSON_INPUT_POS Struct Reference                      | 136        |
| 9.5 JSON_PARSER Struct Reference                         | 136        |
| 9.6 JSON_VALUE Struct Reference                          | 136        |
| 9.7 wolfsentry_allocator Struct Reference                | 137        |
| 9.7.1 Detailed Description                               | 137        |
| 9.8 wolfsentry_build_settings Struct Reference           | 137        |
| 9.8.1 Detailed Description                               | 137        |
| 9.8.2 Field Documentation                                | 137        |
| 9.8.2.1 config   | 137        |
| 9.8.2.2 version  | 138        |
| 9.9 wolfsentry_eventconfig Struct Reference              | 138        |
| 9.9.1 Detailed Description                               | 138        |
| 9.10 wolfsentry_host_platform_interface Struct Reference | 139        |
| 9.10.1 Detailed Description                              | 139        |
| 9.10.2 Field Documentation                               | 139        |
| 9.10.2.1 allocator                                       | 139        |
| 9.10.2.2 caller_build_settings                           | 139        |
| 9.10.2.3 semcbs  | 139        |
| 9.10.2.4 timecbs   | 139        |
| 9.11 wolfsentry_kv_pair Struct Reference                 | 140        |

|   |            |
|---|------------|
| 9.11.1 Detailed Description                             | 140        |
| 9.11.2 Field Documentation                              | 140        |
| 9.11.2.1 b  | 140        |
| 9.12 wolfsentry_route_endpoint Struct Reference         | 140        |
| 9.12.1 Detailed Description                             | 141        |
| 9.13 wolfsentry_route_exports Struct Reference          | 141        |
| 9.13.1 Detailed Description                             | 142        |
| 9.14 wolfsentry_route_metadata_exports Struct Reference | 142        |
| 9.14.1 Detailed Description                             | 142        |
| 9.15 wolfsentry_semcbs Struct Reference                 | 142        |
| 9.15.1 Detailed Description                             | 143        |
| 9.16 wolfsentry_sockaddr Struct Reference               | 143        |
| 9.16.1 Detailed Description                             | 143        |
| 9.17 wolfsentry_thread_context_public Struct Reference  | 144        |
| 9.17.1 Detailed Description                             | 144        |
| 9.18 wolfsentry_timecbs Struct Reference                | 144        |
| 9.18.1 Detailed Description                             | 144        |
| <b>10 File Documentation</b>                            | <b>145</b> |
| 10.1 centijson_dom.h                                    | 145        |
| 10.2 centijson_sax.h                                    | 147        |
| 10.3 centijson_value.h                                  | 151        |
| 10.4 wolfsentry/wolfsentry.h File Reference             | 158        |
| 10.4.1 Detailed Description                             | 180        |
| 10.5 wolfsentry.h                                       | 180        |
| 10.6 wolfsentry/wolfsentry_af.h File Reference          | 200        |
| 10.6.1 Detailed Description                             | 202        |
| 10.7 wolfsentry_af.h                                    | 203        |
| 10.8 wolfsentry/wolfsentry_errcodes.h File Reference    | 204        |
| 10.8.1 Detailed Description                             | 208        |
| 10.9 wolfsentry_errcodes.h                              | 209        |
| 10.10 wolfsentry/wolfsentry_json.h File Reference       | 214        |
| 10.10.1 Detailed Description                            | 215        |
| 10.11 wolfsentry_json.h                                 | 215        |
| 10.12 wolfsentry/wolfsentry_lwip.h File Reference       | 216        |
| 10.12.1 Detailed Description                            | 217        |
| 10.13 wolfsentry_lwip.h                                 | 217        |
| 10.14 wolfsentry/wolfsentry_settings.h File Reference   | 218        |
| 10.14.1 Detailed Description                            | 221        |
| 10.15 wolfsentry_settings.h                             | 221        |
| 10.16 wolfsentry/wolfsentry_util.h File Reference       | 228        |
| 10.16.1 Detailed Description                            | 230        |



|   |            |
|---|------------|
| 10.17 <a href="#">wolfsentry_util.h</a> . . . . . | 230        |
| <b>Index</b>                                      | <b>235</b> |



# Chapter 1

## wolfSentry: the wolfSSL embedded firewall and IDPS

### 1.1 Description

wolfSentry is the wolfSSL embedded firewall and IDPS (Intrusion Detection and Prevention System). It is normally built as a library that is linked in with an application, which can run on bare metal or in a multiuser runtime.

At a high level, wolfSentry is a dynamically configurable logic hub, arbitrarily associating user-defined events with user-defined actions, contextualized by connection attributes, tracking the evolution of the client-server relationship. At a low level, wolfSentry is an embedded firewall engine (both static and fully dynamic), with wildcard-capable lookup of known hosts/netblocks, and unlimited extensibility through callbacks for address families and action handlers, private data segments for each tracked peer, and a dictionary of user-defined configuration and state nodes, including freeform deep-tree JSON.

The wolfSentry engine is dynamically configurable programmatically through an API, or from a textual input file in JSON supplied to the engine. Callbacks implement application-specific functionalities such as deep packet inspection, orchestrations, and remote logging.

### 1.2 Documentation

Basic application integration on FreeRTOS-lwIP is documented, with usable code fragments, by [doc/freertos-lwip-app.md](#).

The JSON configuration blob is documented in detail by [doc/json\\_configuration.md](#).

The latest changes and additions are noted in the [ChangeLog.md](#) at the top of the repository.

### 1.3 Dependencies

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

```
make STATIC=1 SINGLETHREADED=1 NO_STDIO=1 EXTRA_CFLAGS='-DWOLFSENTRY_NO_CLOCK_BUILTIN  
-DWOLFSENTRY_NO_MALLOC_BUILTIN'
```

generates a `libwolfSentry.a` that depends on only `inet_ntop()` and a handful of basic string functions. Allocator and time callbacks must then be set in a struct `wolfSentry_host_platform_interface` supplied to `wolfSentry_init()`.



## Chapter 2

# Building and Initializing wolfSentry for an application on FreeRTOS/lwIP

Building the wolfSentry library for FreeRTOS with lwIP 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,
    .max_connection_count = 10,
    .derogatory_threshold_for_penaltybox = 4,
    .penaltybox_duration = 300,
    .route_idle_time_for_purge = 0,
    .flags = WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY,
    .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,
    /* default alignment -- same as sizeof(void *). */
    /* by default, don't allow more than 10 simultaneous
     * connections that match the same route.
     */
    /* after 4 derogatory events matching the same route,
     * put the route in penalty box status.
     */
    /* keep routes in penalty box status for 5 minutes.
     * denominated in seconds when passing to
     * wolfSentry_init().
     */
    /* 0 to disable -- autopurge doesn't usually make
     * much sense as a default config.
     */
    /* automatically clear
     * derogatory count for a route when a commendable
     * event matches the route.
     */

```

```

        .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().
 */
wolfentry_errcode_t activate_wolfentry_lwip(const char *json_config, int json_config_len)
{
    wolfentry_errcode_t ret;
    char err_buf[512]; /* buffer for detailed error messages from
        * wolfentry_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 (wolfentry_lwip_ctx != NULL) {
        printf("activate_wolfentry_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 "wolfentry" is not the correct arg
     * to pass. This is used here to pass a null pointer for the host platform
     * interface ("hpi").
     */
    ret = wolfentry_init(
        wolfentry_build_settings,
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(NULL /* hpi */),
        &demo_config,
        &wolfentry_lwip_ctx);
    if (ret < 0) {
        printf("wolfentry_init() failed: " WOLFSENTRY_ERROR_FMT "\n",
            WOLFSENTRY_ERROR_FMT_ARGS(ret));
        goto out;
    }

    /* Insert user-defined actions here, if any. */
    ret = wolfentry_action_insert(
        WOLFSENTRY_CONTEXT_ARGS_OUT_EX(wolfentry_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),

#ifdef LWIP_ARP || LWIP_ETHERNET
        LWIP_ALL_EVENTS, /* ethernet_mask */
#else
        0,
#endif
#ifdef LWIP_IPV4 || LWIP_IPV6
        LWIP_ALL_EVENTS, /* ip_mask */
#else
        0,
#endif
#ifdef LWIP_ICMP || LWIP_ICMP6
        LWIP_ALL_EVENTS, /* icmp_mask */
#else
        0,
#endif
#ifdef LWIP_TCP
        LWIP_ALL_EVENTS, /* tcp_mask */
#else
        0,
#endif
#ifdef 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 `wolfentry_init()` (which can be passed in `NULL`, signifying built-in defaults). Note that times (`config.penaltybox_duration` and `config.route_idle_time_for_purge`) shall be passed to `wolfentry_init()` denominated in seconds, notwithstanding the `wolfentry_time_t` type of the members.

## JSON load flags

The `flags` argument to `wolfentry_config_json_init()` and `wolfentry_config_json_one_shot()`, 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 `wolfentry_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_MAINTAIN_DICT_ORDER` – When processing user-defined JSON values, store sequence information so that subsequent calls to `wolfentry_kv_render_value()` or `json_dom_dump(..., JSON_DOM_DUMP_PREFER_DICT_ORDER)` 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.

```
{
  "wolfentry-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
  },
  "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,
        "port" : endpoint_port
    },
    "local" : {
        "interface" : uint8,
        "address" : route_address,
        "prefix-bits" : uint16,
        "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

**wolf Sentry 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.
- **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 `wolfentry_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 fallback default policies for dispatch routines such as `wolfentry_route_event_dispatch()`.

- **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.
  - **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. `wolfentry_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

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

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_port_clause "," ]
  -","
"}"

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

route_address_clause = DQUOTE %s"address" 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...0177777777777777777777 > 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 -0100000000000000000000...0777777777777777777777 >
    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 1.5.0 (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 `wolfentry_route_default_policy_set()` and `wolfentry_route_default_policy_get()` implicitly accessing the main route table.

Added public functions `wolfentry_get_object_type()` and `wolfentry_object_release()`, companions to existing `wolfentry_object_checkout()` and `wolfentry_get_object_id()`.

Added `wolfentry_lock_size()` to facilitate caller-allocated `wolfentry_rwlock`s.

`WOLFENTRY_CONTEXT_ARGS_OUT` is now the first argument to utility routines `wolfentry_object_checkout()`, `wolfentry_defaultconfig_get()`, and `wolfentry_defaultconfig_update()`, rather than a bare `wolfentry` context pointer.

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

Removed unneeded routine `wolfentry_config_json_set_default_config()`.

Improved `wolfentry_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 `wolfentry_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 `wolfentry_config_json_init_ex()`, error if `json_config.max_key_len` is greater than `WOLFENTRY_MAX_LABEL_BYTES` (required for memory safety).

In `wolfentry_config_json_init_ex()`, call `wolfentry_defaultconfig_get()` to initialize `jps->default_config` with settings previously passed to `wolfentry_init()`.

`src/kv.c`: Fixed `_KV_STRING` and `_KV_BYTES` cases in `wolfentry_kv_value_eq_1()` (inadvertently inverted `memcmp()`), and fixed `_KV_NONE` case to return true.

Fixed `wolfentry_kv_render_value()` for `_KV_JSON` case to pass `JSON_DOM_DUMP_PREFERDICTORDER` to `json_dom_dump()`.

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

In `wolfentry_init_ex()`, correctly convert user-supplied `route_idle_time_for_purge` from seconds to `wolfentry_time_t`.

Pass `route_table->default_event` to `wolfentry_route_event_dispatch_0()` if caller-supplied trigger event is null (changed in `wolfentry_route_event_dispatch_1()`, `wolfentry_route_event_dispatch_by_id_1()`, and `wolfentry_route_event_dispatch_by_route_1()`).

In `wolfentry_route_lookup_0()`, fixed scoping of `WOLFENTRY_ACTION_RES_EXCLUDE_↵`  
`REJECT_ROUTES` to only check `WOLFENTRY_ROUTE_FLAG_PENALTYBOXED`, not `WOLFENTRY_↵`  
`ROUTE_FLAG_PORT_RESET`.

In `wolfentry_route_delete_0()`, properly set `WOLFENTRY_ROUTE_FLAG_PENDING_DELETE`.

In `wolfentry_route_event_dispatch_0()` and `wolfentry_route_event_dispatch_1()`, properly set `WOLFENTRY_ACTION_RES_ERROR` at end if `ret < 0`.

In `wolfentry_route_event_dispatch_1()`, properly set `WOLFENTRY_ACTION_RES_↵`  
`FALLTHROUGH` when `route_table->default_policy` is used.

Added missing `action_results` reset to `wolfentry_route_delete_for_filter()`.

In `wolfentry_lock_init()`, properly forbid all inapplicable flags.

Fixed `wolfentry_eventconfig_update_1()` to copy over all relevant elements.

Fixed and updated expression for `WOLFENTRY_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 `WOLFENTRY_UNITTEST_BENCHMARKS` sections in `test_static_routes()` and `test_↵`  
`json()`;
- Add to `test_init()` tests of `wolfentry_errcode_source_string()` and `wolfentry_errcode_error_s`
- Add to `test_static_routes()` tests of `wolfentry_route_default_policy_set()` and  
`wolfentry_get_object_type()`, `wolfentry_object_checkout()`, and `wolfentry_object_relea`

## 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 `wolfentry_event_set_aux_event()` and `wolfentry_event_get_aux_event()`.

Added flag filters and controls to struct `wolfentry_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 `WOLFENTRY_ACTION_RES_*` (action result) flags to support filtering matches by generic traffic type:

- `WOLFENTRY_ACTION_RES_SENDING`
- `WOLFENTRY_ACTION_RES_RECEIVED`
- `WOLFENTRY_ACTION_RES_BINDING`
- `WOLFENTRY_ACTION_RES_LISTENING`
- `WOLFENTRY_ACTION_RES_STOPPED_LISTENING`
- `WOLFENTRY_ACTION_RES_CONNECTING_OUT`
- `WOLFENTRY_ACTION_RES_CLOSED`
- `WOLFENTRY_ACTION_RES_UNREACHABLE`
- `WOLFENTRY_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 `wolfentry/wolfentry.h`.

Added `wolfentry_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 `wolfentry_route_exports`, and related helper functions to prepare it:

- `wolfentry_route_insert_by_exports_into_table()`
- `wolfentry_route_insert_by_exports()`
- `wolfentry_route_insert_by_exports_into_table_and_check_out()`
- `wolfentry_route_insert_by_exports_and_check_out()`
- `wolfentry_route_reset_metadata_exports()`

Added convenience accessor/validator functions for routes:

- `wolfentry_route_get_addrs()`
- `wolfentry_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 `wolfentry_event`, rather than through a "subevent". The related APIs (`wolfentry_event_action_prepend()`, `wolfentry_event_action_append()`, `wolfentry_event_acti↵wolfentry_event_action_delete()`, `wolfentry_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 `wolfentry_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 `wolfentry_route_metadata_exports`, changed `.connection_count`, `.derogatory_↵_count`, and `.commendable_count`, from `wolfentry_hitcount_t` to `uint16_t`, to match internal representations. Similarly, in struct `wolfentry_route_exports`, changed `.parent_event_↵label_len` from `size_t` to `int` to match `label_len` arg type.

Added `wolfentry_table_ent_get_by_id()` to the public API.

Renamed public API `wolfentry_action_res_decode()` as `wolfentry_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 `wolfentry_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:

- `wolfentry_event_set_aux_event()`
- `wolfentry_event_get_aux_event()`
- `wolfentry_event_get_label()`
- `wolfentry_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 `wolfentry_route_get_reference()`, and corresponding lock assertion in `wolfentry_table_cursor_init()`.

Fixed logic in address matching to properly match zero-length addresses when performing 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 `wolfentry_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.

- `wolfentry_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_RENDER_FLUSH()` and retrying the `wolfentry_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 `__wolfentry_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 `wolfentry_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 `wolfentry_lock_unlock()`, and effected myriad cleanups to improve clarity and portability.

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

Fixed route metadata coherency in transactional configuration updates: add `wolfentry_route_copy_↵ metadata()`, and call it from `wolfentry_context_exchange()`.

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

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

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

Copy `fallthrough_route` pointer in `wolfentry_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 `WOLFENTRY_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-init` checks to `thread/lock` routines in `src/wolfentry_util.c`, and corresponding unit test coverage for all null/uninit arg permutations.

Added `assert` in release recipe to assure that `wolfentry.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 `wolfentry_route_render_flags()`, now used in `wolfentry_route_render()` and `wolfentry_route_exports_render()`.

Refactored `wolfentry_route_lookup_0()` to consistently return the highest-priority matching route, breaking ties using `compare_match_exactness()`.

Added `DEBUG_ROUTE_LOOKUP` code paths in `wolfentry_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 `wolfentry_route_init()`, `wolfentry_route_new()`, and `wolfentry_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 `wolfentry_install_lwip_filter_ethernet_callback()`, `wolfentry_install_lwip_filter_ip_callbacks()`, `wolfentry_install_lwip_filter_icmp_callbacks()`, `wolfentry_install_lwip_filter_tcp_callback()`, `wolfentry_install_lwip_filter_udp_callbacks()` and the all-on-one `wolfentry_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 `wolfentry_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 `wolfentry_route_event_dispatch()`), ignore failure in `wolfentry_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 `wolfentry_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 `wolfentry_action_list_{append,prepend,insert}_1()`, and added missing `point_action` lookup in `wolfentry_action_list_insert_after()`.

Fixed potential use-after-free defect in `wolfentry_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 `wolfentry_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 `wolfentry_init_ex()` and `wolfentry_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/wolfentry_util.c`.
- `wolfentry/wolfentry_settings.h` was added, containing setup code previously in `wolfentry/wolfentry.h`.
- Error IDs in `enum wolfentry_error_id` are all now negative, and a new `WOLFENTRY_<-SUCCESS_ID_*` namespace was added, with positive values and supporting macros.

**New public utility APIs, macros, types, etc.**

- `WOLFSENTRY_VERSION_*` macros, for version testing
- `wolfentry_init_thread_context()`, `wolfentry_alloc_thread_context()`, `wolfentry_get_thread_id()`, `wolfentry_get_thread_user_context()`, `wolfentry_get_thread_flags()`, `wolfentry_destroy_thread_context()`, `wolfentry_free_thread_context()`, `wolfentry_set_deadline_rel_usecs()`, `wolfentry_set_deadline_abs()`, `wolfentry_clear_deadline()`, `wolfentry_set_thread_readonly()`, `wolfentry_set_thread_readwrite()`
- `WOLFSENTRY_DEADLINE_NEVER` and `WOLFSENTRY_DEADLINE_NOW`, used internally and for testing values returned by `wolfentry_get_thread_deadline()`
- Many new values in the `WOLFSENTRY_LOCK_FLAG_*` set.
- `wolfentry_lock_*`() APIs now firmed, and new `wolfentry_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 `wolfentry_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 `wolfentry_config_json_init_ex()` and `wolfentry_config_json_onehot_ex()`, accepting a `JSON_CONFIG *` (accepted as `const`).



## New APIs for JSON KVs

- `wolfentry_user_value_store_json()`
- `wolfentry_user_value_get_json()`
- `WOLFENTRY_KV_V_JSON()`
- `wolfentry_config_json_init_ex()`
- `wolfentry_config_json_one_shot_ex()`

## New config load flags controlling JSON KV parsing

- `WOLFENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT`
- `WOLFENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST`
- `WOLFENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST`
- `WOLFENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAIN_DICT_ORDER`

## 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:

- `wolfentry_user_value_set_mutability()`
- `wolfentry_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.
- `wolfentry_kv_render_value()` now has a `struct wolfentry_context *` as its first argument (necessitated by addition of freeform JSON rendering).
- Added new API routine `wolfentry_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

- `wolfentry_get_allocator()`
- `wolfentry_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:**

- `wolfentry_route_increment_derogatory_count()`
- `wolfentry_route_increment_commendable_count()`
- `wolfentry_route_reset_derogatory_count()`
- `wolfentry_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
- `wolfentry_route_table_max_purgeable_routes_get()`
- `wolfentry_route_table_max_purgeable_routes_set()`
- `wolfentry_route_stale_purge_one()`

**Noteworthy Changes and Additions**

- New API `wolfentry_route_insert_and_check_out()`, allowing efficient update of route state after insert; also related new API `wolfentry_object_checkout()`.
- New APIs `wolfentry_route_event_dispatch_by_route()` and `wolfentry_route_event_dispatch_by_id()` analogous to the `_by_id()` variants, but accepting a struct `wolfentry_route` pointer directly.
- `wolfentry_route_init()` and `wolfentry_route_new()` now allow (and ignore) nonzero supplied values in wildcarded `wolfentry_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 `wolfentry_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 `wolfentry_init_ex()` with `wolfentry_init_flags_t` argument.
- Added runtime error-checking on lock facility.

## Bug Fixes

Fix missing assignment in `wolfentry_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 `wolfentry_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 `wolfentry_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 falthrough 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 (`wolfentry_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 `WOLFENTRY_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 `wolfentry_config_json_init()`, properly copy the `load_flags` from the caller into the `_json↵_process_state`.
- The JSON SAX API routines (`wolfentry/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 `wolfentry_options.h`, synchronizing applications with wolfSentry library options as built.
- New APIs `wolfentry_route_event_dispatch_[by_id]with_initiated_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: `wolfentry_free_aligned_cb_t`, `wolfentry_allocator.free_aligned`, `wolfentry_builtin_free_aligned()`, `wolfentry_free_aligned()`, and `WOLFSENTRY↵_FREE_ALIGNED()`.
- Semaphore wrappers for FreeRTOS, for use by the `wolfentry_lock_*`() shareable-upgradeable lock facility.

## Bug Fixes

- `wolfentry_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 [douzzzer@wolfssl.com](mailto:douzzzer@wolfssl.com)



## Chapter 5

# Module Index

### 5.1 Modules

Here is a list of all modules:

|   |     |
|---|-----|
| Core Types and Macros . . . . .   | 43  |
| Startup/Configuration/Shutdown Subsystem . . . . .                          | 44  |
| Diagnostics, Control Flow Helpers, and Compiler Attribute Helpers . . . . . | 53  |
| Route/Rule Subsystem . . . . .  | 58  |
| Action Subsystem . . . . .  | 80  |
| Event Subsystem . . . . .   | 88  |
| Address Family Subsystem . . . . .  | 101 |
| User-Defined Value Subsystem . . . . .                                      | 104 |
| Object Subsystem . . . . .  | 109 |
| Thread Synchronization Subsystem . . . . .                                  | 111 |
| Allocator (Heap) Functions and Callbacks . . . . .                          | 130 |
| Time Functions and Callbacks . . . . .                                      | 131 |
| Semaphore Function Callbacks . . . . .                                      | 132 |
| lwIP Callback Activation Functions . . . . .                                | 134 |





## Chapter 6

# Data Structure Index

### 6.1 Data Structures

Here are the data structures with brief descriptions:

|   |     |
|---|-----|
| <a href="#">JSON_CALLBACKS</a>  | 135 |
| <a href="#">JSON_CONFIG</a>   | 135 |
| <a href="#">JSON_DOM_PARSER</a>   | 135 |
| <a href="#">JSON_INPUT_POS</a>  | 136 |
| <a href="#">JSON_PARSER</a>   | 136 |
| <a href="#">JSON_VALUE</a>  | 136 |
| <a href="#">wolfsentry_allocator</a>  |     |
| Struct for passing shims that abstract the native implementation of the heap allocator                              | 137 |
| <a href="#">wolfsentry_build_settings</a>   |     |
| Struct for passing the build version and configuration  | 137 |
| <a href="#">wolfsentry_eventconfig</a>  |     |
| Struct for representing event configuration   | 138 |
| <a href="#">wolfsentry_host_platform_interface</a>  |     |
| Struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores | 139 |
| <a href="#">wolfsentry_kv_pair</a>  |     |
| Public structure for passing user-defined values in/out of wolfSentry   | 140 |
| <a href="#">wolfsentry_route_endpoint</a>   |     |
| Struct for exporting socket addresses, with fixed-length fields   | 140 |
| <a href="#">wolfsentry_route_exports</a>  |     |
| Struct for exporting a route for access by applications   | 141 |
| <a href="#">wolfsentry_route_metadata_exports</a>   |     |
| Struct for exporting route metadata for access by applications  | 142 |
| <a href="#">wolfsentry_semcbs</a>   |     |
| Struct for passing shims that abstract the native implementation of counting semaphores                             | 142 |
| <a href="#">wolfsentry_sockaddr</a>   |     |
| Struct for passing socket addresses into <code>wolfsentry_route_*</code> () API routines                            | 143 |
| <a href="#">wolfsentry_thread_context_public</a>  |     |
| Right-sized, right-aligned opaque container for thread state  | 144 |
| <a href="#">wolfsentry_timecbs</a>  |     |
| Struct for passing shims that abstract the native implementation of time functions                                  | 144 |



# Chapter 7

## File Index

### 7.1 File List

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

|   |     |
|---|-----|
| wolfentry/ <a href="#">centijson_dom.h</a>  | 145 |
| wolfentry/ <a href="#">centijson_sax.h</a>  | 147 |
| wolfentry/ <a href="#">centijson_value.h</a>                                      | 151 |
| wolfentry/ <a href="#">wolfentry.h</a>  |     |
| The main include file for wolfSentry applications                                 | 158 |
| wolfentry/ <a href="#">wolfentry_af.h</a>   |     |
| Definitions for address families  | 200 |
| wolfentry/ <a href="#">wolfentry_errcodes.h</a>                                   |     |
| Definitions for diagnostics   | 204 |
| wolfentry/ <a href="#">wolfentry_json.h</a>                                       |     |
| Types and prototypes for loading/reloading configuration using JSON               | 214 |
| wolfentry/ <a href="#">wolfentry_lwip.h</a>                                       |     |
| Prototypes for lwIP callback installation functions, for use in lwIP applications | 216 |
| wolfentry/ <a href="#">wolfentry_settings.h</a>                                   |     |
| Target- and config-specific settings and abstractions for wolfSentry              | 218 |
| wolfentry/ <a href="#">wolfentry_util.h</a>                                       |     |
| Utility and convenience macros for both internal and application use              | 228 |



## Chapter 8

# Module 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\_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\_ENT\_ID\_FMT**  
*printf-style format string appropriate for pairing with `wolfentry_ent_id_t`*
- **#define WOLFSENTRY\_ENT\_ID\_NONE**  
*always-invalid object ID*
- **#define WOLFSENTRY\_HITCOUNT\_FMT**  
*printf-style format string appropriate for pairing with `wolfentry_hitcount_t`*
- **#define \_\_wolfentry\_wur**  
*abstracted attribute designating that the return value must be checked to avoid a compiler warning*
- **#define wolfentry\_static\_assert(c)**  
*abstracted static assert – `c` must be true, else `c` is printed*
- **#define wolfentry\_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. Incurs 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 **wolfentry\_addr\_family\_t**  
*integer type for holding address family number*
- typedef uint16\_t **wolfentry\_proto\_t**  
*integer type for holding protocol number*
- typedef uint16\_t **wolfentry\_port\_t**  
*integer type for holding port number*
- typedef uint32\_t **wolfentry\_ent\_id\_t**  
*integer type for holding table entry ID*
- typedef uint16\_t **wolfentry\_addr\_bits\_t**  
*integer type for address prefix lengths (in bits)*
- typedef uint32\_t **wolfentry\_hitcount\_t**  
*integer type for holding hit count statistics*
- typedef int64\_t **wolfentry\_time\_t**  
*integer type for holding absolute and relative times, using microseconds in built-in implementations.*
- typedef uint16\_t **wolfentry\_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 [wolfentry\\_host\\_platform\\_interface](#)  
*struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores*
- struct [wolfentry\\_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 [wolfentry.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 [wolfentry.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 [wolfentry.h](#).
- **#define WOLFSENTRY\_VERSION\_EQ(major, minor, tiny)**  
Helper macro that is true if the given version equals that in [wolfentry.h](#).
- **#define WOLFSENTRY\_VERSION\_LT(major, minor, tiny)**  
Helper macro that is true if the given version is less than that in [wolfentry.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 [wolfentry.h](#).
- **#define WOLFSENTRY\_MAX\_JSON\_NESTING 16**  
Can be overridden.
- **#define WOLFSENTRY\_USER\_SETTINGS\_FILE "the\_path"**  
Define [WOLFSENTRY\\_USER\\_SETTINGS\\_FILE](#) 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`.
- **#define WOLFSENTRY\_NO\_INTTYPES\_H**  
Define to inhibit inclusion of `inttypes.h` (alternative typedefs or include must be supplied with [WOLFSENTRY\\_USER\\_SETTINGS\\_FILE](#)).
- **#define WOLFSENTRY\_NO\_STDINT\_H**  
Define to inhibit inclusion of `stdint.h` (alternative typedefs 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 `wolfentry_util.c`, then the [wolfentry\\_host\\_platform\\_interface](#) supplied to wolfSentry APIs must include a full semaphore implementation (shim set) in its [wolfentry\\_semcbcs](#) 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\_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 [wolfentry\\_util.h](#)).
- **#define WOLFSENTRY\_NO\_CLOCK\_BUILTIN**  
If defined, omit built-in time primitives; the [wolfentry\\_host\\_platform\\_interface](#) supplied to wolfSentry APIs must include implementations of all functions in `struct wolfentry_timecbcs`.
- **#define WOLFSENTRY\_NO\_MALLOC\_BUILTIN**  
If defined, omit built-in heap allocator primitives; the [wolfentry\\_host\\_platform\\_interface](#) supplied to wolfSentry APIs must include implementations of all functions in `struct wolfentry_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.

## Typedefs

- typedef void(\* **wolfentry\_cleanup\_callback\_t**) (**WOLFSENTRY\_CONTEXT\_ARGS\_IN**, void \*cleanup\_↔ arg)

Function type to pass to [wolfentry\\_cleanup\\_push\(\)](#)

- typedef uint32\_t **wolfentry\_config\_load\_flags\_t**

Type for holding flag bits from [wolfentry\\_config\\_load\\_flags](#).

## Enumerations

- enum **wolfentry\_init\_flags\_t** {  
[WOLFSENTRY\\_INIT\\_FLAG\\_NONE](#) ,  
[WOLFSENTRY\\_INIT\\_FLAG\\_LOCK\\_SHARED\\_ERROR\\_CHECKING](#) }

flags to pass to [wolfentry\\_init\\_ex\(\)](#), to be OR'd together.

- enum **wolfentry\_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 [wolfentry\\_context\\_clone\(\)](#) and other cloning functions.

- enum **wolfentry\_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\\_MAINTAININDICTORDER](#) ,  
[WOLFSENTRY\\_CONFIG\\_LOAD\\_FLAG\\_FLUSH\\_ONLY\\_ROUTES](#) ,  
[WOLFSENTRY\\_CONFIG\\_LOAD\\_FLAG\\_FINI](#) }

Flags to be OR'd together to communicate options to [wolfentry\\_config\\_json\\_init\(\)](#)

## Functions

- WOLFSENTRY\_API struct **wolfentry\_build\_settings** **wolfentry\_get\_build\_settings** (void)

Return the [wolfentry\\_build\\_settings](#) of the library as built.

- WOLFSENTRY\_API **wolfentry\_errcode\_t** **wolfentry\_build\_settings\_compatible** (struct **wolfentry\_build\_settings** caller\_build\_settings)

Return success if the application and library were built with mutually compatible wolfSentry version and configuration.

- WOLFSENTRY\_API struct **wolfentry\_host\_platform\_interface** \* **wolfentry\_get\_hpi** (struct wolfentry\_↔ context \*wolfentry)

Return a pointer to the [wolfentry\\_host\\_platform\\_interface](#) associated with the supplied [wolfentry\\_context](#), mainly for passing to [wolfentry\\_alloc\\_thread\\_context\(\)](#), [wolfentry\\_free\\_thread\\_context\(\)](#), [wolfentry\\_lock\\_init\(\)](#), and [wolfentry\\_lock\\_alloc\(\)](#).



- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_cleanup\\_push](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_cleanup\\_callback\\_t](#) handler, void \*arg)  
*Register handler to be called at shutdown with arg arg.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_cleanup\\_all](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
*Iteratively call [wolfentry\\_cleanup\\_pop\(\)](#), executing each handler as it is popped, passing it the arg with which it was registered.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_init\\_ex](#) (struct [wolfentry\\_build\\_settings](#) caller\_build\_settings, [WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)(const struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi), const struct [wolfentry\\_eventconfig](#) \*config, struct [wolfentry\\_context](#) \*\*wolfentry, [wolfentry\\_init\\_flags\\_t](#) flags)  
*Variant of [wolfentry\\_init\(\)](#) that accepts a flags argument, for additional control over configuration.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_init](#) (struct [wolfentry\\_build\\_settings](#) caller\_build\_settings, [WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)(const struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi), const struct [wolfentry\\_eventconfig](#) \*config, struct [wolfentry\\_context](#) \*\*wolfentry)  
*Allocates and initializes the wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_defaultconfig\\_get](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_eventconfig](#) \*config)  
*Get the default config from a wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_defaultconfig\\_update](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct [wolfentry\\_eventconfig](#) \*config)  
*Updates mutable fields of the default config (all but [wolfentry\\_eventconfig::route\\_private\\_data\\_size](#) and [wolfentry\\_eventconfig::route\\_private\\_data\\_alignment](#))*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_flush](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
*Flushes the route, event, and user value tables from the wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_free](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)(struct [wolfentry\\_context](#) \*\*wolfentry))  
*Frees the wolfentry context and the tables within it. The wolfentry context will be a pointer to NULL upon success.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_shutdown](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)(struct [wolfentry\\_context](#) \*\*wolfentry))  
*Shut down wolfentry, freeing all resources. Gets an exclusive lock on the context, then calls [wolfentry\\_context\\_free\(\)](#).*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_inhibit\\_actions](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
*Disable automatic dispatch of actions on the wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_enable\\_actions](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
*Re-enable automatic dispatch of actions on the wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_clone](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_context](#) \*\*clone, [wolfentry\\_clone\\_flags\\_t](#) flags)  
*Clones a wolfentry context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_exchange](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_context](#) \*wolfentry2)  
*Swaps information between two wolfentry contexts.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_centijson\\_errcode\\_translate](#) ([wolfentry\\_errcode\\_t](#) centijson\_errcode)  
*Convert CentiJSON numeric error code to closest-corresponding wolfentry error code.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_json\\_init](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_config\\_load\\_flags\\_t](#) load\_flags, struct [wolfentry\\_json\\_process\\_state](#) \*\*jps)  
*Allocate and initialize a struct [wolfentry\\_json\\_process\\_state](#) with the designated load\_flags, to subsequently pass to [wolfentry\\_config\\_json\\_feed\(\)](#).*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_json\\_init\\_ex](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_config\\_load\\_flags\\_t](#) load\_flags, const [JSON\\_CONFIG](#) \*json\_config, struct [wolfentry\\_json](#) \*\*process\_state \*\*jps)

Variant of `wolfentry_config_json_init()` with an additional `JSON_CONFIG` argument, `json_config`, for tailoring of JSON parsing dynamics.

- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_config_json_feed` (struct `wolfentry_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 `wolfentry_errcode_t wolfentry_config_centijson_errcode` (struct `wolfentry_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 `wolfentry_json_process_state` allocated by `wolfentry_config_json_init()`.

- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_config_json_fini` (struct `wolfentry_json_process_state` \*\*jps, char \*err\_buf, size\_t err\_buf\_size)

To be called when done iterating `wolfentry_config_json_feed()`, completing the configuration load.

- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_config_json_oneshot` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const unsigned char \*json\_in, size\_t json\_in\_len, `wolfentry_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 `wolfentry_errcode_t wolfentry_config_json_oneshot_ex` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const unsigned char \*json\_in, size\_t json\_in\_len, `wolfentry_config_load_flags_t` load\_flags, const `JSON_CONFIG` \*json\_config, char \*err\_buf, size\_t err\_buf\_size)

Variant of `wolfentry_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 wolfentry\_clone\_flags\_t

enum `wolfentry_clone_flags_t`

Flags to be OR'd together to control the dynamics of `wolfentry_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 <code>wolfentry_init()</code> . 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 wolfentry\_config\_load\_flags

enum `wolfentry_config_load_flags`

Flags to be OR'd together to communicate options to `wolfentry_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_↔<br>THEN_COMMIT              | Test the load operation before replacing the current configuration.   |
| WOLFSENTRY_CONFIG_LOAD_FLAG_NO_↔<br>ROUTES_OR_EVENTS           | Skip routes and events in the supplied configuration.   |
| WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↔<br>DOM_DUPKEY_ABORT         | When loading JSON user values, treat as an error when duplicate keys are found.   |
| WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↔<br>DOM_DUPKEY_USEFIRST      | When loading JSON user values, when duplicate keys are found, keep the first one.   |
| WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↔<br>DOM_DUPKEY_USELAST       | When loading JSON user values, when duplicate keys are found, keep the last one.  |
| WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_↔<br>DOM_MAINTAINDICTIONORDER | When loading JSON user values, store extra sequence information so that dictionaries are rendered in same sequence by <code>json_dom_dump()</code> and <code>wolfentry_kv_render_value()</code> . |
| WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_↔<br>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 wolfentry\_init\_flags\_t

```
enum wolfentry_init_flags_t
```

flags to pass to `wolfentry_init_ex()`, to be OR'd together.

## Enumerator

|  |   |
|--|---|
| WOLFSENTRY_INIT_FLAG_NONE                            | Default behavior.   |
| WOLFSENTRY_INIT_FLAG_LOCK_SHARED_↔<br>ERROR_CHECKING | Enables supplementary error checking on shared lock usage (not currently implemented) |

## 8.2.3 Function Documentation

## 8.2.3.1 wolfentry\_context\_clone()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_context_clone (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_context ** clone,
    wolfentry_clone_flags_t flags )
```

Clones a wolfentry context.

## Parameters

|              |  |
|--------------|--|
| <i>clone</i> | the destination wolfsentry context, should be a pointer to a NULL pointer as this function will malloc   |
| <i>flags</i> | 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

|                    |  |
|--------------------|--|
| <i>wolfsentry2</i> | 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 `wolfentry_context_flush()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_context_flush (
    WOLFENTRY_CONTEXT_ARGS_IN )
```

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

##### Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

##### See also

[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.2.3.5 `wolfentry_context_free()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_context_free (
    WOLFENTRY_CONTEXT_ARGS_IN_EX(struct wolfentry_context **wolfentry) )
```

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

##### Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true, and `*wolfentry` is NULL, on success.

##### See also

`wolfentry_context_shutdown`

[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)

#### 8.2.3.6 `wolfentry_context_inhibit_actions()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_context_inhibit_actions (
    WOLFENTRY_CONTEXT_ARGS_IN )
```

Disable automatic dispatch of actions on the wolfentry context.

##### Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

##### See also

[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.2.3.7 `wolfentry_defaultconfig_get()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_defaultconfig_get (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_eventconfig * config )
```

Get the default config from a wolfentry context.

## Parameters

|               |  |
|---------------|--|
| <i>config</i> | 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 wolfentry\_defaultconfig\_update()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_defaultconfig_update (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const struct wolfentry_eventconfig * config )
```

Updates mutable fields of the default config (all but [wolfentry\\_eventconfig::route\\_private\\_data\\_size](#) and [wolfentry\\_eventconfig::route\\_private\\_data\\_alignment](#))

## Parameters

|               |                                |
|---------------|--------------------------------|
| <i>config</i> | 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 wolfentry\_init()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_init (
    struct wolfentry_build_settings caller_build_settings,
    WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfentry_host_platform_interface
    *hpi) ,
    const struct wolfentry_eventconfig * config,
    struct wolfentry_context ** wolfentry )
```

Allocates and initializes the wolfentry context.

## Parameters

|                              |   |
|------------------------------|---|
| <i>caller_build_settings</i> | Pass <a href="#">wolfentry_build_settings</a> here (definition is in <a href="#">wolfentry_settings.h</a> ) |
| <i>config</i>                | a pointer to a <a href="#">wolfentry_eventconfig</a> to use (can be NULL)                                   |
| <i>wolfentry</i>             | a pointer to the <a href="#">wolfentry_context</a> to initialize  |

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

struct [wolfentry\\_host\\_platform\\_interface](#)  
[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN\\_EX](#)

## 8.2.3.10 wolfentry\_shutdown()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_shutdown (
    WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfentry_context **wolfentry) )
```

Shut down wolfSentry, freeing all resources. Gets an exclusive lock on the context, then calls [wolfentry\\_context\\_free\(\)](#).

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true, and \*wolfentry is NULL, on success.

## See also

[wolfentry\\_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 wolfentry\_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 wolfentry\_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 wolfentry\_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 wolfentry\_errcode\_t*
- **#define WOLFSENTRY\_ERROR\_DECODE\_SOURCE\_ID(x)**  
*Extract the bare source file ID from an encoded wolfentry\_errcode\_t*
- **#define WOLFSENTRY\_ERROR\_DECODE\_LINE\_NUMBER(x)**  
*Extract the bare source line number from an encoded wolfentry\_errcode\_t*
- **#define WOLFSENTRY\_ERROR\_RECODE(x)**  
*Take an encoded wolfentry\_errcode\_t and recode it with the current source ID and line number.*
- **#define WOLFSENTRY\_ERROR\_CODE\_IS(x, name)**

- Take an encoded `wolfentry_errcode_t` `x` and test if its error code matches short-form error name (e.g. `INVALID_ARG`).
- **#define WOLFENTRY\_SUCCESS\_CODE\_IS(x, name)**  
Take an encoded `wolfentry_errcode_t` `x` and test if its error code matches short-form success name (e.g. `OK`).
  - **#define WOLFENTRY\_IS\_FAILURE(x)**  
Evaluates to true if `x` is a `wolfentry_errcode_t` that encodes a failure.
  - **#define WOLFENTRY\_IS\_SUCCESS(x)**  
Evaluates to true if `x` is a `wolfentry_errcode_t` that encodes a success.
  - **#define WOLFENTRY\_ERROR\_FMT**  
Convenience string-constant macro for formatting a `wolfentry_errcode_t` for rendering by a `printf`-type function.
  - **#define WOLFENTRY\_ERROR\_FMT\_ARGS(x)**  
Convenience macro supplying args to match the format directives in `WOLFENTRY_ERROR_FMT`.
  - **#define WOLFENTRY\_ERROR\_ENCODE(name)**  
Compute a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form error name (e.g. `INVALID_ARG`).
  - **#define WOLFENTRY\_SUCCESS\_ENCODE(x)**  
Compute a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form success name (e.g. `OK`).
  - **#define WOLFENTRY\_DEBUG\_CALL\_TRACE**  
Define to build the library or application to output codepoint and error code info at each return point.
  - **#define WOLFENTRY\_ERROR\_RETURN(x)**  
Return a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form error name (e.g. `INVALID_ARG`).
  - **#define WOLFENTRY\_SUCCESS\_RETURN(x)**  
Return a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form success name (e.g. `OK`).
  - **#define WOLFENTRY\_ERROR\_RETURN\_RECODED(x)**  
Take an encoded `wolfentry_errcode_t`, recode it with the current source ID and line number, and return it.
  - **#define WOLFENTRY\_ERROR\_REReturn(x)**  
Return an encoded `wolfentry_errcode_t`.
  - **#define WOLFENTRY\_RETURN\_VALUE(x)**  
Return an arbitrary value.
  - **#define WOLFENTRY\_RETURN\_VOID**  
Return from a void function.
  - **#define WOLFENTRY\_SUCCESS\_RETURN\_RECODED(x)**  
Take an encoded `wolfentry_errcode_t`, recode it with the current source ID and line number, and return it.
  - **#define WOLFENTRY\_SUCCESS\_REReturn(x)**  
Return an encoded `wolfentry_errcode_t`.
  - **#define WOLFENTRY\_UNLOCK\_FOR\_RETURN\_EX(ctx)**  
Unlock a previously locked `wolfentry_context`, and if the unlock fails, return the error.
  - **#define WOLFENTRY\_UNLOCK\_FOR\_RETURN()**  
Unlock the current context, and if the unlock fails, return the error.
  - **#define WOLFENTRY\_UNLOCK\_AND\_UNRESERVE\_FOR\_RETURN\_EX(ctx)**  
Unlock a previously locked `wolfentry_context`, and abandon a held promotion reservation if any (see `wolfentry_lock_unlock()`), and if the operation fails, return the error.
  - **#define WOLFENTRY\_UNLOCK\_AND\_UNRESERVE\_FOR\_RETURN()**  
Unlock the current context, and abandon a held promotion reservation if any (see `wolfentry_lock_unlock()`), and if the operation fails, return the error.
  - **#define WOLFENTRY\_MUTEX\_EX(ctx)**  
Get a mutex on a `wolfentry_context`, evaluating to the resulting `wolfentry_errcode_t`.
  - **#define WOLFENTRY\_MUTEX\_OR\_RETURN()**



- Get a mutex on the current context, and on failure, return the `wolfentry_errcode_t`.*

  - **#define WOLFENTRY\_SHARED\_EX(ctx)**

*Get a shared lock on a `wolfentry_context`, evaluating to the resulting `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_SHARED\_OR\_RETURN()**

*Get a shared lock on the current context, and on failure, return the `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_PROMOTABLE\_EX(ctx)**

*Get a mutex on a `wolfentry_context`, evaluating to the resulting `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_PROMOTABLE\_OR\_RETURN()**

*Get a shared lock with mutex promotion reservation on the current context, and on failure, return the `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_UNLOCK\_AND\_RETURN(ret)**

*Unlock the current context, and return the supplied `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_ERROR\_UNLOCK\_AND\_RETURN(name)**

*Unlock the current context, and return a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form error name (e.g. `INVALID_ARG`).*
  - **#define WOLFENTRY\_ERROR\_UNLOCK\_AND\_RETURN\_RECODED(x)**

*Unlock the current context, then take an encoded `wolfentry_errcode_t` `x`, recode it with the current source ID and line number, and return it.*
  - **#define WOLFENTRY\_ERROR\_UNLOCK\_AND\_RETURN\_EX(ctx, name)**

*Unlock a previously locked `wolfentry_context` `ctx`, and return a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form error name (e.g. `INVALID_ARG`).*
  - **#define WOLFENTRY\_ERROR\_UNLOCK\_AND\_RETURN\_RECODED\_EX(ctx, x)**

*Unlock a previously locked `wolfentry_context` `ctx`, then take an encoded `wolfentry_errcode_t` `x`, recode it with the current source ID and line number, and return it.*
  - **#define WOLFENTRY\_ERROR\_UNLOCK\_AND\_REReturn(x)**

*Unlock the current context, and return an encoded `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_ERROR\_REReturn\_AND\_UNLOCK(y)**

*Calculate the `wolfentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_SUCCESS\_UNLOCK\_AND\_RETURN(name)**

*Unlock the current context, and return a `wolfentry_errcode_t` encoding the current source ID and line number, and the designated short-form success name (e.g. `INVALID_ARG`).*
  - **#define WOLFENTRY\_SUCCESS\_UNLOCK\_AND\_RETURN\_RECODED(x)**

*Unlock the current context, then take an encoded `wolfentry_errcode_t` `x`, recode it with the current source ID and line number, and return it.*
  - **#define WOLFENTRY\_SUCCESS\_UNLOCK\_AND\_REReturn(x)**

*Unlock the current context, and return an encoded `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_SUCCESS\_REReturn\_AND\_UNLOCK(y)**

*Calculate the `wolfentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfentry_errcode_t`.*
  - **#define WOLFENTRY\_UNLOCK\_AND\_RETURN\_VALUE(x)**

*Unlock the current context, and return a value `x`.*
  - **#define WOLFENTRY\_UNLOCK\_AND\_RETURN\_VOID**

*Unlock the current context, and return void.*
  - **#define WOLFENTRY\_RETURN\_OK**

*Return a `wolfentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.*
  - **#define WOLFENTRY\_UNLOCK\_AND\_RETURN\_OK**

*Unlock the current context, and return a `wolfentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.*
  - **#define WOLFENTRY\_REReturn\_IF\_ERROR(y)**

*If `wolfentry_errcode_t` `y` is a failure code, return it.*
  - **#define WOLFENTRY\_UNLOCK\_AND\_REReturn\_IF\_ERROR(y)**

*If `wolfentry_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` or `WOLFSENTRY_NO_DIAG_MSGS` is set, `DO_NOTHING`.
- **#define WOLFSENTRY\_WARN\_ON\_FAILURE(...)**  
Evaluate the supplied expression, and if the resulting `wolfentry_errcode_t` encodes an error, render the expression and the decoded error using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO` 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` or `WOLFSENTRY_NO_DIAG_MSGS` is set, don't render a warning.
- **#define WOLFSENTRY\_REGISTER\_SOURCE()**  
Helper macro to call `wolfentry_user_source_string_set()` with appropriate arguments.
- **#define WOLFSENTRY\_REGISTER\_ERROR(name, msg)**  
Helper macro to call `wolfentry_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 wolfentry\_errcode\_t**  
The structured result code type for `wolfentry`. It encodes a failure or success code, a source code file ID, and a line number.

## Enumerations

- **enum wolfentry\_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 wolfentry\_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_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_USER_BASE = 128 }

```

## Functions

- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_source\_string** ([wolfentry\\_errcode\\_t](#) e)

*Return the name of the source code file associated with `wolfentry_errcode_t` e, or "unknown user defined source", or "unknown source".*

- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_error\_string** ([wolfentry\\_errcode\\_t](#) e)

*Return a description of the failure or success code associated with `wolfentry_errcode_t` e, or various "unknown" strings if not known.*

- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_error\_name** ([wolfentry\\_errcode\\_t](#) e)

*Return the short name of the failure or success code associated with `wolfentry_errcode_t` e, or `wolfentry_errcode_error_string(e)` if not known.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_user\_source\_string\_set** (enum `wolfentry_errcode_t` source\_id `wolfentry_source_id`, const char \*source\_string)

*Register a source code file so that `wolfentry_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_error\\_string\\_set](#) (enum [wolfentry\\_error\\_id](#) ↩  
id [wolfentry\\_error\\_id](#), const char \*message\_string)

*Register an error (negative) or success (positive) code, and corresponding message, so that [wolfentry\\_errcode\\_error\\_string](#) 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 [wolfentry\\_settings.h](#), subject to override.

## 8.4 Route/Rule Subsystem

### Data Structures

- struct [wolfentry\\_route\\_endpoint](#)  
*struct for exporting socket addresses, with fixed-length fields*
- struct [wolfentry\\_route\\_metadata\\_exports](#)  
*struct for exporting route metadata for access by applications*
- struct [wolfentry\\_route\\_exports](#)  
*struct for exporting a route for access by applications*
- struct [wolfentry\\_sockaddr](#)  
*struct for passing socket addresses into [wolfentry\\_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 [wolfentry\\_route\\_table\\_default\\_policy\\_set\(\)](#)*
- #define **WOLFSENTRY\_ROUTE\_WILDCARD\_FLAGS**  
*Bit mask for the wildcard bits in a [wolfentry\\_route\\_flags\\_t](#).*
- #define **WOLFSENTRY\_ROUTE\_IMMUTABLE\_FLAGS**  
*Bit mask for the bits in a [wolfentry\\_route\\_flags\\_t](#) that can't change after the implicated route has been inserted in the route table.*
- #define **WOLFSENTRY\_SOCKADDR(n)**  
*Macro to instantiate a [wolfentry\\_sockaddr](#) with an `addr` field sized to hold `n` bits of address data. Cast to [struct wolfentry\\_sockaddr](#) to pass as API argument.*

## Enumerations

- enum `wolfentry_route_flags_t` {  
`WOLFENTRY_ROUTE_FLAG_NONE` = 0U ,  
`WOLFENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD` ,  
`WOLFENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS` ,  
`WOLFENTRY_ROUTE_FLAG_DIRECTION_IN` ,  
`WOLFENTRY_ROUTE_FLAG_DIRECTION_OUT` ,  
`WOLFENTRY_ROUTE_FLAG_IN_TABLE` ,  
`WOLFENTRY_ROUTE_FLAG_PENDING_DELETE` ,  
`WOLFENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED` ,  
`WOLFENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED` ,  
`WOLFENTRY_ROUTE_FLAG_PENALTYBOXED` ,  
`WOLFENTRY_ROUTE_FLAG_GREENLISTED` ,  
`WOLFENTRY_ROUTE_FLAG_DONT_COUNT_HITS` ,  
`WOLFENTRY_ROUTE_FLAG_DONT_COUNT_CURRENT_CONNECTIONS` ,  
`WOLFENTRY_ROUTE_FLAG_PORT_RESET` }

*bit field specifying attributes of a route/rule*

- enum `wolfentry_format_flags_t` {  
`WOLFENTRY_FORMAT_FLAG_NONE` ,  
`WOLFENTRY_FORMAT_FLAG_ALWAYS_NUMERIC` }

*bit field with options for rendering*

## Functions

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_check_flags_sensical` (`wolfentry_route_flags_t` flags)

*Check the self-consistency of flags.*

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_insert_into_table` (`WOLFENTRY_CONTEXT_ARGS_IN`, struct `wolfentry_route_table` \*route\_table, void \*caller\_arg, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, `wolfentry_ent_id_t` \*id, `wolfentry_action_res_t` \*action\_results)

*Variant of `wolfentry_route_insert()` that takes an explicit `route_table`.*

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_insert_by_exports_into_table` (`WOLFENTRY_CONTEXT_ARGS_IN`, struct `wolfentry_route_table` \*route\_table, void \*caller\_arg, const struct `wolfentry_route_exports` \*route\_exports, `wolfentry_ent_id_t` \*id, `wolfentry_action_res_t` \*action\_results)

*Variant of `wolfentry_route_insert()` that accepts the new route as `wolfentry_route_exports`, and takes an explicit `route_table`.*

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_insert` (`WOLFENTRY_CONTEXT_ARGS_IN`, void \*caller\_arg, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, `wolfentry_ent_id_t` \*id, `wolfentry_action_res_t` \*action\_results)

*Insert a route into the route table.*

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_insert_by_exports` (`WOLFENTRY_CONTEXT_ARGS_IN`, void \*caller\_arg, const struct `wolfentry_route_exports` \*route\_exports, `wolfentry_ent_id_t` \*id, `wolfentry_action_res_t` \*action\_results)

*Variant of `wolfentry_route_insert()` that accepts the new route as `wolfentry_route_exports`.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_into\\_table\\_and\\_check\\_out](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*route\_table, void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label\_len, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
 Variant of [wolfentry\\_route\\_insert\(\)](#) that takes an explicit *route\_table*, and returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_by\\_exports\\_into\\_table\\_and\\_check\\_out](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*route\_table, void \*caller\_arg, const struct [wolfentry\\_route\\_exports](#) \*route\_exports, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
 Variant of [wolfentry\\_route\\_insert\(\)](#) that accepts the new route as *wolfentry\_route\_exports*, takes an explicit *route\_table*, and returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_and\\_check\\_out](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label\_len, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
 Variant of [wolfentry\\_route\\_insert\(\)](#) that returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_by\\_exports\\_and\\_check\\_out](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_route\\_exports](#) \*route\_exports, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
 Variant of [wolfentry\\_route\\_insert\(\)](#) that accepts the new route as *wolfentry\_route\_exports* and returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete\\_from\\_table](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*route\_table, void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label\_len, [wolfentry\\_action\\_res\\_t](#) \*action\_results, int \*n\_deleted)  
 Variant of [wolfentry\\_route\\_delete\(\)](#) that takes an explicit *route\_table*.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*trigger\_label, int trigger\_label\_len, [wolfentry\\_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 [wolfentry\\_route\\_delete\\_by\\_id\(\)](#). The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete\\_by\\_id](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, [wolfentry\\_ent\\_id\\_t](#) id, const char \*trigger\_label, int trigger\_label\_len, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_get\\_main\\_table](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*\*table)  
 Get a pointer to the internal route table. Caller must have a lock on the context at entry.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_start](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*\*cursor)  
 Open a cursor to iterate through a routes table. Caller must have a lock on the context at entry.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_seek\\_to\\_head](#) (const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor)  
 Reset the cursor to the beginning of a table.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_seek\\_to\\_tail](#) (const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor)  
 Move the cursor to the end of a table.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_current](#) (const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor, struct [wolfentry\\_route](#) \*\*route)



*Get the current position for the table cursor.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_prev](#) (const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*cursor, struct wolfentry\_route \*\*route)

*Get the previous position for the table cursor.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_next](#) (const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*cursor, struct wolfentry\_route \*\*route)

*Get the next position for the table cursor.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_end](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*\*cursor)

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

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_default\\_policy\\_set](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route\_table \*table, [wolfentry\\_action\\_res\\_t](#) default\_policy)

*Set a table's default policy.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_default\\_policy\\_set](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_action\\_res\\_t](#) default\_policy)

*variant of [wolfentry\\_route\\_table\\_default\\_policy\\_set\(\)](#) that uses the main route table implicitly, and takes care of context locking.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_default\\_policy\\_get](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route\_table \*table, [wolfentry\\_action\\_res\\_t](#) \*default\_policy)

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

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_default\\_policy\\_get](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_action\\_res\\_t](#) \*default\_policy)

*variant of [wolfentry\\_route\\_table\\_default\\_policy\\_get\(\)](#) that uses the main route table implicitly. Caller must have a lock on the context at entry.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_get\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route\_table \*table, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label\_len, int exact\_p, [wolfentry\\_route\\_flags\\_t](#) \*inexact\_matches, struct wolfentry\_route \*\*route)

*Increments a reference counter for a route.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_drop\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route \*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

*Decrease a reference counter for a route.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_clear\\_default\\_event](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route\_table \*table)

*Clear an event previously set by [wolfentry\\_route\\_table\\_set\\_default\\_event\(\)](#).*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_set\\_default\\_event](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_get\\_default\\_event](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route\_table \*table, char \*event\_label, int \*event\_label\_len)

*Get the event, if any, set by [wolfentry\\_route\\_table\\_set\\_default\\_event\(\)](#)*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_fallthrough\\_route\\_get](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_route\_table \*route\_table, const struct wolfentry\_route \*\*fallthrough\_route)

*Retrieve the default route in a route table, chiefly to pass to [wolfentry\\_route\\_update\\_flags\(\)](#).*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_get\\_addrs](#) (const struct wolfentry\_route \*route, [wolfentry\\_addr\\_family\\_t](#) \*af, [wolfentry\\_addr\\_bits\\_t](#) \*local\_addr\_len, const byte \*\*local\_addr, [wolfentry\\_addr\\_bits\\_t](#) \*remote\_addr\_len, const byte \*\*remote\_addr)

*Extract numeric address family and binary address pointers from a [wolfentry\\_route](#)*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_export](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route \*route, struct [wolfentry\\_route\\_exports](#) \*route\_exports)

*Exports a route.*

- WOLFSENTRY\_API const struct wolfentry\_event \* [wolfentry\\_route\\_parent\\_event](#) (const struct wolfentry\_route \*route)

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

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_with_table` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route_table` \*route\_table, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_ent_id_t` \*id, `wolfentry_route_flags_t` \*inexact\_matches, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_route_event_dispatch()` that accepts an explicit `route_table`.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch` (WOLFENTRY\_CONTEXT\_ARGS\_IN, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_ent_id_t` \*id, `wolfentry_route_flags_t` \*inexact\_matches, `wolfentry_action_res_t` \*action\_results)

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

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_with_table_with_init_result` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route_table` \*route\_table, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_ent_id_t` \*id, `wolfentry_route_flags_t` \*inexact\_matches, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_route_event_dispatch()` that accepts an explicit `route_table`, and doesn't clear `action_results` on entry.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_with_init_result` (WOLFENTRY\_CONTEXT\_ARGS\_IN, const struct `wolfentry_sockaddr` \*remote, const struct `wolfentry_sockaddr` \*local, `wolfentry_route_flags_t` flags, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_ent_id_t` \*id, `wolfentry_route_flags_t` \*inexact\_matches, `wolfentry_action_res_t` \*action\_results)

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

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_by_id` (WOLFENTRY\_CONTEXT\_ARGS\_IN, `wolfentry_ent_id_t` id, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_route_event_dispatch()` that preselects the matched route by ID, mainly for use by application code that tracks ID/session relationships.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_by_id_with_init_result` (WOLFENTRY\_CONTEXT\_ARGS\_IN, `wolfentry_ent_id_t` id, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_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.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_by_route` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route` \*route, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_route_event_dispatch()` that preselects the matched route by ID, mainly for use by application code that tracks route/session relationships.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_event_dispatch_by_route_with_init_result` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route` \*route, const char \*event\_label, int event\_label\_len, void \*caller\_arg, `wolfentry_action_res_t` \*action\_results)

Variant of `wolfentry_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.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_table_max_purgeable_routes_get` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route_table` \*table, `wolfentry_hitcount_t` \*max\_purgeable\_routes)

Retrieve the current limit for ephemeral routes in `table`. Caller must have a lock on the context at entry.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_table_max_purgeable_routes_set` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route_table` \*table, `wolfentry_hitcount_t` max\_purgeable\_routes)

Set the limit for ephemeral routes in `table`. Caller must have a mutex on the context at entry.

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_route_stale_purge` (WOLFENTRY\_CONTEXT\_ARGS\_IN, struct `wolfentry_route_table` \*table, `wolfentry_action_res_t` \*action\_results)



- Purges stale (expired) routes from `table`.*
- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_stale_purge_one` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route_table *table`, `wolfentry_action_res_t *action_results`)  
*Variant of `wolfentry_route_stale_purge()` that purges at most one stale route, to limit time spent working.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_stale_purge_one_opportunistically` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route_table *table`, `wolfentry_action_res_t *action_results`)  
*Variant of `wolfentry_route_stale_purge()` that purges at most one stale route, and only if the context lock is uncontended.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_flush_table` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route_table *table`, `wolfentry_action_res_t *action_results`)  
*Flush routes from a given table.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_bulk_clear_insert_action_status` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `wolfentry_action_res_t *action_results`)  
*Clears the `WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED` flag on all routes in the table.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_bulk_insert_actions` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `wolfentry_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 `wolfentry_errcode_t wolfentry_route_get_private_data` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `void **private_data`, `size_t *private_data_size`)  
*Gets the private data for a given route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_get_flags` (`const struct wolfentry_route *route`, `wolfentry_route_flags_t *flags`)  
*Gets the flags for a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_get_metadata` (`const struct wolfentry_route *route`, `struct wolfentry_route_metadata_exports *metadata`)  
*Gets the metadata for a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_reset_metadata_exports` (`struct wolfentry_route_exports *route_exports`)  
*clear metadata counts (`wolfentry_route_metadata_exports::purge_after`, `wolfentry_route_metadata_exports::connection_count`, `wolfentry_route_metadata_exports::derogatory_count`, and `wolfentry_route_metadata_exports::commendable_count`) in `wolfentry_route_exports` to prepare for use with `wolfentry_route_insert_by_exports()`*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_update_flags` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `wolfentry_route_flags_t flags_to_set`, `wolfentry_route_flags_t flags_to_clear`, `wolfentry_route_flags_t *flags_before`, `wolfentry_route_flags_t *flags_after`, `wolfentry_action_res_t *action_results`)  
*Update the route flags.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_increment_derogatory_count` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `int count_to_add`, `int *new_derogatory_count_ptr`)  
*Increase the derogatory event count of a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_increment_commendable_count` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `int count_to_add`, `int *new_commendable_count`)  
*Increase the commendable event count of a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_reset_derogatory_count` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `int *old_derogatory_count_ptr`)  
*Reset the derogatory event count of a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_reset_commendable_count` (`WOLFSENTRY_CONTEXT_ARGS_IN`, `struct wolfentry_route *route`, `int *old_commendable_count_ptr`)  
*Reset the commendable event count of a route.*
  - WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_route_set_wildcard` (`struct wolfentry_route *route`, `wolfentry_route_flags_t wildcards_to_set`)  
*Set wildcard flags for a route.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_format\\_address](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_flag\\_assoc\\_by\\_flag](#) ([wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_flag\\_assoc\\_by\\_name](#) (const char \*name, int len, [wolfentry\\_route\\_flags\\_t](#) \*flag)  
*Retrieve the numeric value of a route flag, given its name.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_format\\_json](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route](#) \*r, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_start](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*\*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)  
*Start a rendering loop to export the route table contents as a JSON document that is valid input for [wolfentry\\_config\\_json\\_feed\(\)](#) or [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_next](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)  
*Render a route within a loop started with [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_end](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*\*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)  
*Finish a rendering loop started with [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_render\\_flags](#) ([wolfentry\\_route\\_flags\\_t](#) flags, FILE \*f)  
*Render route flags in human-readable form to a stream.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_render](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route](#) \*r, FILE \*f)  
*Renders route information to a file pointer.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_exports\\_render](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route\\_exports](#) \*r, FILE \*f)  
*Renders route exports information to a file pointer.*

## 8.4.1 Detailed Description

## 8.4.2 Enumeration Type Documentation

### 8.4.2.1 [wolfentry\\_format\\_flags\\_t](#)

enum [wolfentry\\_format\\_flags\\_t](#)

bit field with options for rendering

## Enumerator

|  |   |
|--|---|
| WOLFSENTRY_FORMAT_FLAG_NONE                | Default rendering behavior.   |
| WOLFSENTRY_FORMAT_FLAG_ALWAYS_↔<br>NUMERIC | When rendering address families and protocols, always render as bare integers. Currently honored by <a href="#">wolfentry_route_format_json()</a> . |

## 8.4.2.2 wolfentry\_route\_flags\_t

```
enum wolfentry_route_flags_t
```

bit field specifying attributes of a route/rule

## Enumerator

|  |  |
|--|--|
| WOLFSENTRY_ROUTE_FLAG_NONE                           | No attributes  |
| WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_↔<br>WILDCARD        | Address family is wildcard – match all traffic in specified direction(s), optionally with specified interfaces.  |
| WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_↔<br>ADDR_WILDCARD   | Remote address is wildcard – match any remote address.   |
| WOLFSENTRY_ROUTE_FLAG_SA_PROTO_↔<br>WILDCARD         | Protocol is wildcard – match any protocol.   |
| WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_↔<br>PORT_WILDCARD    | Local port is wildcard – match any local port.   |
| WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_↔<br>ADDR_WILDCARD    | Local address is wildcard – match any local address.   |
| WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_↔<br>PORT_WILDCARD   | Remote port is wildcard – match any remote port.   |
| WOLFSENTRY_ROUTE_FLAG_REMOTE_↔<br>INTERFACE_WILDCARD | Ingestion interface is wildcard – match any ingestion interface.   |
| WOLFSENTRY_ROUTE_FLAG_LOCAL_↔<br>INTERFACE_WILDCARD  | Local interface (usually same as remote interface) is wildcard – match any local interface.  |
| WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_↔<br>_WILDCARD    | Match regardless of parent event mismatch.   |
| WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_↔<br>_NUMBERS     | Interpret port names using TCP/UDP mappings (available unless build option <a href="#">WOLFSENTRY_NO_GETPROTOBY</a> is defined)  |
| WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN                   | Match inbound traffic.   |
| WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT                  | Match outbound traffic (if <a href="#">WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN</a> and <a href="#">WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT</a> are both set, traffic in both directions is matched) |
| 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_↔<br>ACTIONS_CALLED     | Internal use – records that route insertion actions have been completed.   |
| WOLFSENTRY_ROUTE_FLAG_DELETE_↔<br>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.  |

## Enumerator

|  |  |
|--|--|
| WOLFSENTRY_ROUTE_FLAG_DONT_COUNT↔<br>HITS                | Don't keep traffic statistics for this rule (avoid counting overhead)  |
| WOLFSENTRY_ROUTE_FLAG_DONT_COUNT↔<br>CURRENT_CONNECTIONS | Don't keep concurrent connection count for this rule (don't impose connection limit, even if set in the applicable <a href="#">wolfentry_eventconfig</a> )   |
| WOLFSENTRY_ROUTE_FLAG_PORT_RESET                         | If traffic is rejected by this rule, set <a href="#">WOLFSENTRY_ACTION_RES_PORT_RESET</a> in the returned <a href="#">wolfentry_action_res_t</a> , prompting generation by the network stack of a TCP reset, ICMP unreachable, or other applicable reply packet. |

### 8.4.3 Function Documentation

#### 8.4.3.1 wolfentry\_route\_bulk\_clear\_insert\_action\_status()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_bulk_clear_insert_action_status (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    wolfentry_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

[wolfentry\\_route\\_bulk\\_insert\\_actions\(\)](#)

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.2 wolfentry\_route\_bulk\_insert\_actions()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_bulk_insert_actions (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    wolfentry_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

[wolfentry\\_route\\_bulk\\_clear\\_insert\\_action\\_status\(\)](#)

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

### 8.4.3.3 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

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

#### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

### 8.4.3.4 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

|                          |   |
|--------------------------|---|
| <i>caller_arg</i>        | an arbitrary pointer to be passed to callbacks  |
| <i>id</i>                | the object ID, as returned by <a href="#">wolfsentry_route_insert()</a> or <a href="#">wolfsentry_get_object_id()</a> |
| <i>trigger_label</i>     | a label for a trigger event (or null)   |
| <i>trigger_label_len</i> | the length of the trigger_label parameter   |
| <i>action_results</i>    | 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](#)

**8.4.3.5 wolfentry\_route\_drop\_reference()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_drop_reference (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_route * route,
    wolfentry_action_res_t * action_results )
```

Decrease a reference counter for a route.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <i>route</i>          | the route to drop the reference for |
| <i>action_results</i> | a pointer to results of the action  |

**Returns**

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

**8.4.3.6 wolfentry\_route\_event\_dispatch()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const struct wolfentry_sockaddr * remote,
    const struct wolfentry_sockaddr * local,
    wolfentry_route_flags_t flags,
    const char * event_label,
    int event_label_len,
    void * caller_arg,
    wolfentry_ent_id_t * id,
    wolfentry_route_flags_t * inexact_matches,
    wolfentry_action_res_t * action_results )
```

Submit an event into wolfentry and pass it through the filters. The *action\_results* are cleared on entry, and can be checked to see what actions wolfentry 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

|                        |   |
|------------------------|---|
| <i>remote</i>          | the remote sockaddr details   |
| <i>local</i>           | the local sockaddr details  |
| <i>flags</i>           | the flags for the event, set to <a href="#">WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN</a> for an incoming event            |
| <i>event_label</i>     | an optional label for a trigger event   |
| <i>event_label_len</i> | the length of event_label   |
| <i>caller_arg</i>      | an arbitrary pointer to be passed to action callbacks   |
| <i>id</i>              | an optional pointer to a <a href="#">wolfentry_ent_id_t</a> that will be set to the ID of the matched route, if any |
| <i>inexact_matches</i> | details for inexact matches   |
| <i>action_results</i>  | a pointer to a <a href="#">wolfentry_action_res_t</a> , 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](#)

## 8.4.3.7 wolfentry\_route\_export()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_export (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const struct wolfentry_route * route,
    struct wolfentry_route_exports * route_exports )
```

Exports a route.

`route_exports` remains valid only as long as the wolfentry lock is held (shared or exclusive), unless the route was obtained via [wolfentry\\_route\\_get\\_reference\(\)](#), in which case it's valid until [wolfentry\\_route\\_drop\\_reference\(\)](#).

## Parameters

|                      |                           |
|----------------------|---------------------------|
| <i>route</i>         | the route to export       |
| <i>route_exports</i> | the struct to export into |

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.8 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

|          |                             |
|----------|-----------------------------|
| <i>r</i> | the route exports to render |
| <i>f</i> | the pointer to render to    |

##### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

##### See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.9 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

|                       |   |
|-----------------------|---|
| <i>table</i>          | the table to purge  |
| <i>action_results</i> | 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](#)

#### 8.4.3.10 wolfsentry\_route\_get\_addrs()

```
WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs (
    const struct wolfsentry_route * route,
```



```

wolfentry_addr_family_t * af,
wolfentry_addr_bits_t * local_addr_len,
const byte ** local_addr,
wolfentry_addr_bits_t * remote_addr_len,
const byte ** remote_addr )

```

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

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

#### 8.4.3.11 `wolfentry_route_get_flags()`

```

WOLFENTRY_API wolfentry_errcode_t wolfentry_route_get_flags (
    const struct wolfentry_route * route,
    wolfentry_route_flags_t * flags )

```

Gets the flags for a route.

##### Parameters

|              |                                |
|--------------|--------------------------------|
| <i>route</i> | the route to get the flags for |
| <i>flags</i> | a pointer to receive the flags |

##### Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

#### 8.4.3.12 `wolfentry_route_get_main_table()`

```

WOLFENTRY_API wolfentry_errcode_t wolfentry_route_get_main_table (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_route_table ** table )

```

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

##### Parameters

|              |  |
|--------------|--|
| <i>table</i> | a pointer to a pointer to a table which will be filled |
|--------------|--|

##### Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

##### See also

[WOLFENTRY\\_SHARED\\_OR\\_RETURN\(\)](#)  
[WOLFENTRY\\_UNLOCK\\_AND\\_RETURN\(\)](#)  
[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.13 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

|                 |  |
|-----------------|--|
| <i>route</i>    | the route to get the metadata for              |
| <i>metadata</i> | a pointer to a pointer to receive the metadata |

##### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### 8.4.3.14 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

|                          |   |
|--------------------------|---|
| <i>route</i>             | the route to get the data from                    |
| <i>private_data</i>      | a pointer to a pointer that will receive the data |
| <i>private_data_size</i> | a pointer that will receive the size of the data  |

##### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

##### See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.15 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,
```

```

wolfentry_route_flags_t flags,
const char * event_label,
int event_label_len,
int exact_p,
wolfentry_route_flags_t * inexact_matches,
struct wolfentry_route ** route )

```

Increments a reference counter for a route.

#### Parameters

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

#### Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### See also

[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.16 wolfentry\_route\_insert()

```

WOLFENTRY_API wolfentry_errcode_t wolfentry_route_insert (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    void * caller_arg,
    const struct wolfentry_sockaddr * remote,
    const struct wolfentry_sockaddr * local,
    wolfentry_route_flags_t flags,
    const char * event_label,
    int event_label_len,
    wolfentry_ent_id_t * id,
    wolfentry_action_res_t * action_results )

```

Insert a route into the route table.

#### Parameters

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

**Returns**

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

**8.4.3.17 wolfentry\_route\_parent\_event()**

```
WOLFSENTRY_API const struct wolfentry_event * wolfentry_route_parent_event (
    const struct wolfentry_route * route )
```

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

**Parameters**

|              |                        |
|--------------|------------------------|
| <i>route</i> | a pointer to the route |
|--------------|------------------------|

**Returns**

a pointer to the parent event

**See also**

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

**8.4.3.18 wolfentry\_route\_render()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_render (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const struct wolfentry_route * r,
    FILE * f )
```

Renders route information to a file pointer.

**Parameters**

|          |                          |
|----------|--------------------------|
| <i>r</i> | the route to render      |
| <i>f</i> | the pointer to render to |

**Returns**

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.19 wolfentry\_route\_set\_wildcard()

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_route_set_wildcard (
    struct wolfentry_route * route,
    wolfentry_route_flags_t wildcards_to_set )
```

Set wildcard flags for a route.

##### Parameters

|                         |                                |
|-------------------------|--------------------------------|
| <i>route</i>            | the route to set the flags for |
| <i>wildcards_to_set</i> | the wildcards to be set        |

##### Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### 8.4.3.20 wolfentry\_route\_stale\_purge()

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_route_stale_purge (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_route_table * table,
    wolfentry_action_res_t * action_results )
```

Purges stale (expired) routes from `table`.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>table</i>          | the table to purge from   |
| <i>action_results</i> | the result bit field, pooling results from all constituent operations |

##### Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

##### See also

[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.21 wolfentry\_route\_table\_default\_policy\_get()

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_route_table_default_policy_get (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_route_table * table,
    wolfentry_action_res_t * default_policy )
```

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

## Parameters

|                       |                                 |
|-----------------------|---------------------------------|
| <i>table</i>          | the table to set the policy for |
| <i>default_policy</i> | 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](#)

#### 8.4.3.22 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

|                       |                                 |
|-----------------------|---------------------------------|
| <i>table</i>          | the table to set the policy for |
| <i>default_policy</i> | the policy to set               |

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

#### 8.4.3.23 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\(\)](#)

#### 8.4.3.24 wolfentry\_route\_table\_iterate\_current()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_table_iterate_current (
    const struct wolfentry_route_table * table,
    struct wolfentry_cursor * cursor,
    struct wolfentry_route ** route )
```

Get the current position for the table cursor.

Parameters

|               |   |
|---------------|---|
| <i>table</i>  | the table for the cursor                      |
| <i>cursor</i> | a pointer for the cursor                      |
| <i>route</i>  | a pointer to a pointer for the returned route |

Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### 8.4.3.25 wolfentry\_route\_table\_iterate\_end()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_route_table_iterate_end (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const struct wolfentry_route_table * table,
    struct wolfentry_cursor ** cursor )
```

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

Parameters

|               |   |
|---------------|---|
| <i>table</i>  | the table for the cursor                      |
| <i>cursor</i> | a pointer 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](#)

#### 8.4.3.26 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

|               |   |
|---------------|---|
| <i>table</i>  | the table for the cursor                      |
| <i>cursor</i> | a pointer for the cursor                      |
| <i>route</i>  | a pointer to a pointer for the returned route |

##### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### 8.4.3.27 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

|               |   |
|---------------|---|
| <i>table</i>  | the table for the cursor                      |
| <i>cursor</i> | a pointer for the cursor                      |
| <i>route</i>  | a pointer to a pointer for the returned route |

##### Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

#### 8.4.3.28 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

|               |                          |
|---------------|--------------------------|
| <i>table</i>  | the table for the cursor |
| <i>cursor</i> | a pointer for the cursor |



## Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**8.4.3.29 wolfentry\_route\_table\_iterate\_seek\_to\_tail()**

```
WOLFENTRY_API wolfentry\_errcode\_t wolfentry_route_table_iterate_seek_to_tail (
    const struct wolfentry_route_table * table,
    struct wolfentry_cursor * cursor )
```

Move the cursor to the end of a table.

## Parameters

|               |                          |
|---------------|--------------------------|
| <i>table</i>  | the table for the cursor |
| <i>cursor</i> | a pointer for the cursor |

## Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**8.4.3.30 wolfentry\_route\_table\_iterate\_start()**

```
WOLFENTRY_API wolfentry\_errcode\_t wolfentry_route_table_iterate_start (
    WOLFENTRY\_CONTEXT\_ARGS\_IN ,
    const struct wolfentry_route_table * table,
    struct wolfentry_cursor ** cursor )
```

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

## Parameters

|               |  |
|---------------|--|
| <i>table</i>  | a pointer to the table to open the cursor on |
| <i>cursor</i> | a pointer to a pointer for the cursor        |

## Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFENTRY\\_SHARED\\_OR\\_RETURN\(\)](#)  
[WOLFENTRY\\_UNLOCK\\_AND\\_RETURN\(\)](#)  
[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

### 8.4.3.31 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

|                       |  |
|-----------------------|--|
| <i>route</i>          | the route to update the flags for                              |
| <i>flags_to_set</i>   | new flags to set   |
| <i>flags_to_clear</i> | old flags to clear   |
| <i>flags_before</i>   | a pointer that will be filled with the flags before the change |
| <i>flags_after</i>    | a pointer that will be filled with flags after the change      |
| <i>action_results</i> | 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 bits in [wolfsentry\\_action\\_res\\_t](#).*

#### 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 `wolfentry_action_flags_t` {  
`WOLFENTRY_ACTION_FLAG_NONE` ,  
`WOLFENTRY_ACTION_FLAG_DISABLED` }  
*enum for communicating attributes of an action object*
- enum `wolfentry_action_type_t` {  
`WOLFENTRY_ACTION_TYPE_NONE` ,  
`WOLFENTRY_ACTION_TYPE_POST` ,  
`WOLFENTRY_ACTION_TYPE_INSERT` ,  
`WOLFENTRY_ACTION_TYPE_MATCH` ,  
`WOLFENTRY_ACTION_TYPE_UPDATE` ,  
`WOLFENTRY_ACTION_TYPE_DELETE` ,  
`WOLFENTRY_ACTION_TYPE_DECISION` }  
*enum communicating (to action handlers and internal logic) what type of action is being evaluated*
- enum `wolfentry_action_res_t` {  
`WOLFENTRY_ACTION_RES_NONE` ,  
`WOLFENTRY_ACTION_RES_ACCEPT` ,  
`WOLFENTRY_ACTION_RES_REJECT` ,  
`WOLFENTRY_ACTION_RES_CONNECT` ,  
`WOLFENTRY_ACTION_RES_DISCONNECT` ,  
`WOLFENTRY_ACTION_RES_DEROGATORY` ,  
`WOLFENTRY_ACTION_RES_COMMENDABLE` ,  
`WOLFENTRY_ACTION_RES_STOP` ,  
`WOLFENTRY_ACTION_RES_DEALLOCATED` ,  
`WOLFENTRY_ACTION_RES_INSERTED` ,  
`WOLFENTRY_ACTION_RES_ERROR` ,  
`WOLFENTRY_ACTION_RES_FALLTHROUGH` ,  
`WOLFENTRY_ACTION_RES_UPDATE` ,  
`WOLFENTRY_ACTION_RES_PORT_RESET` ,  
`WOLFENTRY_ACTION_RES_SENDING` ,  
`WOLFENTRY_ACTION_RES_RECEIVED` ,  
`WOLFENTRY_ACTION_RES_BINDING` ,  
`WOLFENTRY_ACTION_RES_LISTENING` ,  
`WOLFENTRY_ACTION_RES_STOPPED_LISTENING` ,  
`WOLFENTRY_ACTION_RES_CONNECTING_OUT` ,  
`WOLFENTRY_ACTION_RES_CLOSED` ,  
`WOLFENTRY_ACTION_RES_UNREACHABLE` ,  
`WOLFENTRY_ACTION_RES_SOCK_ERROR` ,  
`WOLFENTRY_ACTION_RES_USER_BASE` }  
*bit field used to communicate states and attributes through the evaluation pipeline.*

## Functions

- `WOLFENTRY_API const char * wolfentry_action_res_assoc_by_flag` (`wolfentry_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.*
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_action_res_assoc_by_name` (const char \*bit↔ name, size\_t bit\_name\_len, `wolfentry_action_res_t` \*res)  
*Given a `bit_name`, set \*res to the corresponding bit number if known, failing which, return `ITEM_NOT_FOUND`.*
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_action_insert` (`WOLFENTRY_CONTEXT_ARGS_IN`, const char \*label, int label\_len, `wolfentry_action_flags_t` flags, `wolfentry_action_callback_t` handler, void \*handler\_arg, `wolfentry_ent_id_t` \*id)  
*Insert a new action into wolfentry.*
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_action_delete` (`WOLFENTRY_CONTEXT_ARGS_IN`, const char \*label, int label\_len, `wolfentry_action_res_t` \*action\_results)

*Delete an action from wolfentry.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_flush\\_all](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN)

*Flush all actions from wolfentry.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_get\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, struct wolfentry\_action \*\*action)

*Get a reference to an action.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_drop\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_action \*action, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

*Drop a reference to an action.*

- WOLFSENTRY\_API const char \* [wolfentry\\_action\\_get\\_label](#) (const struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_get\\_flags](#) (struct wolfentry\_action \*action, [wolfentry\\_action\\_flags\\_t](#) \*flags)

*Get the flags for an action.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_update\\_flags](#) (struct wolfentry\_action \*action, [wolfentry\\_action\\_flags\\_t](#) flags\_to\_set, [wolfentry\\_action\\_flags\\_t](#) flags\_to\_clear, [wolfentry\\_action\\_flags\\_t](#) \*flags\_before, [wolfentry\\_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 wolfentry\_action\_callback\_t

```
typedef wolfentry\_errcode\_t(* wolfentry_action_callback_t) (WOLFSENTRY\_CONTEXT\_ARGS\_IN,
const struct wolfentry_action *action, void *handler_arg, void *caller_arg, const struct wolfentry↵
_event *trigger_event, wolfentry\_action\_type\_t action_type, const struct wolfentry_route
*trigger_route, struct wolfentry_route_table *route_table, struct wolfentry_route *rule↵
route, wolfentry\_action\_res\_t *action_results)
```

A callback that is triggered when an action is taken.

#### Parameters

|                       |   |
|-----------------------|---|
| <i>action</i>         | a pointer to action details   |
| <i>handler_arg</i>    | an opaque pointer registered with <a href="#">wolfentry_action_insert()</a> , passed to every invocation of the handler |
| <i>caller_arg</i>     | an opaque pointer supplied by the caller to the dispatching <a href="#">wolfentry_route_*</a> () API                    |
| <i>trigger_event</i>  | the event which triggered the action, if any  |
| <i>action_type</i>    | the action type   |
| <i>trigger_route</i>  | a pointer to the subject route, reflecting instantaneous traffic attributes and contents                                |
| <i>route_table</i>    | a pointer to the implicated route table   |
| <i>rule_route</i>     | a pointer to the matched route, reflecting rule logic   |
| <i>action_results</i> | 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 wolfentry\_action\_flags\_t

enum `wolfentry_action_flags_t`

enum for communicating attributes of an action object

Enumerator

|  |  |
|--|--|
| <code>WOLFSENTRY_ACTION_FLAG_NONE</code>     | Default attributes.  |
| <code>WOLFSENTRY_ACTION_FLAG_DISABLED</code> | Disable this action – while this bit is set, dispatches will not call this action. |

#### 8.5.3.2 wolfentry\_action\_res\_t

enum `wolfentry_action_res_t`

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

Enumerator

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

## Enumerator

|  |   |
|--|---|
| WOLFSENTRY_ACTION_RES_SENDING                | caller-preinitiated bit signaling outbound traffic.   |
| WOLFSENTRY_ACTION_RES_RECEIVED               | caller-preinitiated bit signaling inbound traffic.  |
| WOLFSENTRY_ACTION_RES_BINDING                | caller-preinitiated bit signaling that a socket will be bound.  |
| WOLFSENTRY_ACTION_RES_LISTENING              | caller-preinitiated bit signaling that a socket will be listened.   |
| WOLFSENTRY_ACTION_RES_STOPPED_↔<br>LISTENING | caller-preinitiated bit signaling that a socket will stop being listened.   |
| WOLFSENTRY_ACTION_RES_CONNECTING_OUT         | caller-preinitiated bit signaling that an outbound connection will be attempted.  |
| WOLFSENTRY_ACTION_RES_CLOSED                 | caller-preinitiated bit signaling that an association has closed/ended that wasn't created with <code>_CONNECT</code> . |
| WOLFSENTRY_ACTION_RES_UNREACHABLE            | caller-preinitiated bit signaling that traffic destination was unreachable (unbound/unlistened).                        |
| WOLFSENTRY_ACTION_RES SOCK_ERROR             | caller-preinitiated bit signaling that a transport error occurred.  |
| WOLFSENTRY_ACTION_RES_USER_BASE              | start of user-defined results, with user-defined scheme (bit field, sequential, or other). 8 bits are available.        |

## 8.5.3.3 wolfentry\_action\_type\_t

```
enum wolfentry_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 <code>wolfentry_route_dispatch()</code> for a route match.  |
| WOLFSENTRY_ACTION_TYPE_UPDATE   | called by <code>wolfentry_route_dispatch()</code> 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 <code>wolfentry_route_event_dispatch*()</code> .                             |

## 8.5.4 Function Documentation

## 8.5.4.1 wolfentry\_action\_delete()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_delete (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * label,
```

```
int label_len,  
wolfentry_action_res_t * action_results )
```

Delete an action from wolfentry.

#### Parameters

|                       |  |
|-----------------------|--|
| <i>label</i>          | the label of the action to delete  |
| <i>label_len</i>      | the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string |
| <i>action_results</i> | 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 wolfentry\_action\_drop\_reference()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_drop_reference (  
    WOLFSENTRY_CONTEXT_ARGS_IN ,  
    struct wolfentry_action * action,  
    wolfentry_action_res_t * action_results )
```

Drop a reference to an action.

#### Parameters

|                       |   |
|-----------------------|---|
| <i>action</i>         | the action to drop the reference for    |
| <i>action_results</i> | 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 wolfentry\_action\_flush\_all()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_flush_all (  
    WOLFSENTRY_CONTEXT_ARGS_IN )
```

Flush all actions from wolfentry.

**Returns**

`WOLFSENTRY_IS_SUCCESS(ret)` is true on success.

**See also**

`WOLFSENTRY_CONTEXT_ARGS_IN`

**8.5.4.4 wolfentry\_action\_get\_flags()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_get_flags (
    struct wolfentry_action * action,
    wolfentry_action_flags_t * flags )
```

Get the flags for an action.

**Parameters**

|               |                                 |
|---------------|---------------------------------|
| <i>action</i> | the action to get the flags for |
| <i>flags</i>  | the flags to be returned        |

**Returns**

`WOLFSENTRY_IS_SUCCESS(ret)` is true on success.

**8.5.4.5 wolfentry\_action\_get\_label()**

```
WOLFSENTRY_API const char * wolfentry_action_get_label (
    const struct wolfentry_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**

|               |                                 |
|---------------|---------------------------------|
| <i>action</i> | the action to get the label for |
|---------------|---------------------------------|

**Returns**

the label for the action

**8.5.4.6 wolfentry\_action\_get\_reference()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_get_reference (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * label,
    int label_len,
    struct wolfentry_action ** action )
```

Get a reference to an action.



## Parameters

|                  |  |
|------------------|--|
| <i>label</i>     | the label of the action to get the reference for   |
| <i>label_len</i> | the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string |
| <i>action</i>    | 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 wolfentry\_action\_insert()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_insert (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * label,
    int label_len,
    wolfentry_action_flags_t flags,
    wolfentry_action_callback_t handler,
    void * handler_arg,
    wolfentry_ent_id_t * id )
```

Insert a new action into wolfentry.

## Parameters

|                    |  |
|--------------------|--|
| <i>label</i>       | the label for the action   |
| <i>label_len</i>   | the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string |
| <i>flags</i>       | set flags for the action   |
| <i>handler</i>     | a callback handler when the action commences   |
| <i>handler_arg</i> | an arbitrary pointer for the handler callback  |
| <i>id</i>          | 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

|                       |                             |
|-----------------------|-----------------------------|
| <i>action</i>         | the action to update        |
| <i>flags_to_set</i>   | new flags to set            |
| <i>flags_to_clear</i> | old flags to clear          |
| <i>flags_before</i>   | the flags before the change |
| <i>flags_after</i>    | 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\\_CLEAR\\_DEROGATORY](#) ,  
[WOLFSENTRY\\_EVENTCONFIG\\_FLAG\\_INHIBIT\\_ACTIONS](#) }  
*bit field with config flags for events*

## Functions

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_eventconfig\\_init](#) (struct [wolfentry\\_context](#) \*wolfentry, struct [wolfentry\\_eventconfig](#) \*config)  
*Initializes a [wolfentry\\_eventconfig](#) struct with the defaults from the wolfentry context. If no wolfentry context is provided this will initialize to zero.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_eventconfig\\_check](#) (const struct [wolfentry\\_eventconfig](#) \*config)  
*Checks the config for self-consistency and validity.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_insert](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, [wolfentry\\_priority\\_t](#) priority, const struct [wolfentry\\_eventconfig](#) \*config, [wolfentry\\_event\\_flags\\_t](#) flags, [wolfentry\\_ent\\_id\\_t](#) \*id)  
*Insert an event into wolfentry.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_delete](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*Delete an event from wolfentry.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_flush\\_all](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN)  
*Flush all events from wolfentry.*
- WOLFSENTRY\_API const char \* [wolfentry\\_event\\_get\\_label](#) (const struct [wolfentry\\_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 [wolfentry\\_event\\_flags\\_t](#) [wolfentry\\_event\\_get\\_flags](#) (const struct [wolfentry\\_event](#) \*event)  
*Get the flags for an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_get\\_config](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, struct [wolfentry\\_eventconfig](#) \*config)  
*Get the configuration for an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_update\\_config](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, const struct [wolfentry\\_eventconfig](#) \*config)  
*Update the configuration for an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_get\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, struct [wolfentry\\_event](#) \*\*event)  
*Get a reference to an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_drop\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct [wolfentry\\_event](#) \*event, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*Drop a reference to an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_action\\_prepend](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*event\_label, int event\_label\_len, [wolfentry\\_action\\_type\\_t](#) which\_action\_list, const char \*action\_label, int action\_label\_len)  
*Prepend an action into an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_action\\_append](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*event\_label, int event\_label\_len, [wolfentry\\_action\\_type\\_t](#) which\_action\_list, const char \*action\_label, int action\_label\_len)  
*Append an action into an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_action\\_insert\\_after](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*event\_label, int event\_label\_len, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_action\\_delete](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*event\_label, int event\_label\_len, [wolfentry\\_action\\_type\\_t](#) which\_action\_list, const char \*action\_label, int action\_label\_len)  
*Delete an action from an event.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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_↵<br>DEROGATORY_THRESHOLD_IGNORE_↵<br>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_↵<br>COMMENDABLE_CLEARS_DEROGATORY                | If set, then each count from WOLFSENTRY_ACTION_RES_COMMENDABLE zeroes the derogatory count.   |
| WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_↵<br>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

|                          |  |
|--------------------------|--|
| <i>event_label</i>       | the label of the event to append the action into |
| <i>event_label_len</i>   | the length of the event_label                    |
| <i>which_action_list</i> | the action list of the event to update           |
| <i>action_label</i>      | the label of the action to insert                |
| <i>action_label_len</i>  | 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

|                          |  |
|--------------------------|--|
| <i>event_label</i>       | the label of the event to delete the action from |
| <i>event_label_len</i>   | the length of the event_label                    |
| <i>which_action_list</i> | the action list of the event to update           |
| <i>action_label</i>      | the label of the action to delete                |
| <i>action_label_len</i>  | 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 wolfentry\_event\_action\_insert\_after()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_event_action_insert_after (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * event_label,
    int event_label_len,
    wolfentry_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**

|                               |  |
|-------------------------------|--|
| <i>event_label</i>            | the label of the event to insert the action into |
| <i>event_label_len</i>        | the length of the event_label                    |
| <i>which_action_list</i>      | the action list of the event to update           |
| <i>action_label</i>           | the label of the action to insert                |
| <i>action_label_len</i>       | the length of the action_label                   |
| <i>point_action_label</i>     | the label of the action to insert after          |
| <i>point_action_label_len</i> | 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 wolfentry\_event\_action\_list\_done()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_event_action_list_done (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_action_list_ent ** cursor )
```

End iteration started with [wolfentry\\_event\\_action\\_list\\_start\(\)](#). Caller must have a lock on the context at entry.

## Parameters

|               |                                       |
|---------------|---------------------------------------|
| <i>cursor</i> | a pointer to a pointer for the cursor |
|---------------|---------------------------------------|

## Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFENTRY\\_SHARED\\_OR\\_RETURN\(\)](#)  
[WOLFENTRY\\_UNLOCK\\_AND\\_RETURN\(\)](#)  
[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

8.6.3.5 `wolfentry_event_action_list_next()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_list_next (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    struct wolfentry_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

|                         |   |
|-------------------------|---|
| <i>cursor</i>           | a pointer to a pointer for the cursor               |
| <i>action_label</i>     | a pointer to a pointer to the returned action_label |
| <i>action_label_len</i> | the length of action_label                          |

## Returns

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFENTRY\\_SHARED\\_OR\\_RETURN\(\)](#)  
[WOLFENTRY\\_UNLOCK\\_AND\\_RETURN\(\)](#)  
[WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#)

8.6.3.6 `wolfentry_event_action_list_start()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_list_start (
    WOLFENTRY_CONTEXT_ARGS_IN ,
    const char * event_label,
    int event_label_len,
    wolfentry_action_type_t which_action_list,
    struct wolfentry_action_list_ent ** cursor )
```

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

## Parameters

|                          |   |
|--------------------------|---|
| <i>event_label</i>       | the event label to open the iterator for      |
| <i>event_label_len</i>   | the length of the event_label                 |
| <i>which_action_list</i> | the action list of the event to list          |
| <i>cursor</i>            | 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 wolfentry\_event\_action\_prepend()

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_event_action_prepend (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * event_label,
    int event_label_len,
    wolfentry_action_type_t which_action_list,
    const char * action_label,
    int action_label_len )
```

Prepend an action into an event.

## Parameters

|                          |   |
|--------------------------|---|
| <i>event_label</i>       | the label of the event to prepend the action into |
| <i>event_label_len</i>   | the length of the event_label                     |
| <i>which_action_list</i> | the action list of the event to update            |
| <i>action_label</i>      | the label of the action to insert                 |
| <i>action_label_len</i>  | 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

|                       |                                 |
|-----------------------|---------------------------------|
| <i>label</i>          | the label of the even to delete |
| <i>label_len</i>      | the length of the label         |
| <i>action_results</i> | 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

|                       |   |
|-----------------------|---|
| <i>event</i>          | the event to drop the reference for     |
| <i>action_results</i> | 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 wolfentry\_event\_get\_config()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_event_get_config (
    WOLFSENTRY_CONTEXT_ARGS_IN ,
    const char * label,
    int label_len,
    struct wolfentry_eventconfig * config )
```

Get the configuration for an event.

**Parameters**

|                  |   |
|------------------|---|
| <i>label</i>     | the label for the event to get the config for |
| <i>label_len</i> | the length of the label                       |
| <i>config</i>    | the configuration returned                    |

**Returns**

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

**8.6.3.12 wolfentry\_event\_get\_flags()**

```
WOLFSENTRY_API wolfentry_event_flags_t wolfentry_event_get_flags (
    const struct wolfentry_event * event )
```

Get the flags for an event.

**Parameters**

|              |                                |
|--------------|--------------------------------|
| <i>event</i> | 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**

|              |                                |
|--------------|--------------------------------|
| <i>event</i> | 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**

|                  |  |
|------------------|--|
| <i>label</i>     | the label of the event to get the reference for  |
| <i>label_len</i> | the length of the label, use WOLFSENTRY_LENGTH_NULL_TERMINATED for a NUL terminated string |
| <i>event</i>     | 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

|                  |                               |
|------------------|-------------------------------|
| <i>label</i>     | the label for the event       |
| <i>label_len</i> | the length of the label       |
| <i>priority</i>  | the priority of the event     |
| <i>config</i>    | event configuration details   |
| <i>flags</i>     | the flags for the event       |
| <i>id</i>        | 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 `wolfentry_event_set_aux_event()`

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_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

|                            |                                   |
|----------------------------|-----------------------------------|
| <i>event_label</i>         | the parent event label            |
| <i>event_label_len</i>     | the length of the event_label     |
| <i>aux_event_label</i>     | the aux event label               |
| <i>aux_event_label_len</i> | 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 `wolfentry_event_update_config()`

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_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

|                  |   |
|------------------|---|
| <i>label</i>     | the label for the event to get the config for |
| <i>label_len</i> | the length of the label                       |
| <i>config</i>    | the updated configuration                     |

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

## See also

[WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#)

**8.6.3.18 wolfentry\_eventconfig\_check()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_eventconfig_check (
    const struct wolfentry_eventconfig * config )
```

Checks the config for self-consistency and validity.

## Parameters

|               |                                    |
|---------------|------------------------------------|
| <i>config</i> | the pointer to the config to check |
|---------------|------------------------------------|

## Returns

[WOLFSENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**8.6.3.19 wolfentry\_eventconfig\_init()**

```
WOLFSENTRY_API wolfentry_errcode_t wolfentry_eventconfig_init (
    struct wolfentry_context * wolfentry,
    struct wolfentry_eventconfig * config )
```

Initializes a [wolfentry\\_eventconfig](#) struct with the defaults from the wolfentry context. If no wolfentry context is provided this will initialize to zero.

## Parameters

|                  |   |
|------------------|---|
| <i>wolfentry</i> | the wolfentry context                   |
| <i>config</i>    | 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*
  - **#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\_LINK** ([WOLFSENTRY\\_AF\\_BSD\\_OFFSET](#) + 18)  
*Link layer interface.*
- #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 [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_addr\\_family\\_parser\\_t](#)) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const char \*addr\_text, int addr\_text\_len, [byte](#) \*addr\_internal, [wolfentry\\_addr\\_bits\\_t](#) \*addr\_internal\_bits)  
*Function type for parsing handler, to pass to [wolfentry\\_addr\\_family\\_handler\\_install\(\)](#)*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_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 [wolfentry\\_addr\\_family\\_handler\\_install\(\)](#)*

## Functions

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_handler\\_install](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family\_bynumber, const char \*family\_byname, int family\_byname\_len, [wolfentry\\_addr\\_family\\_parser\\_t](#) parser, [wolfentry\\_addr\\_family\\_formatter\\_t](#) formatter, int max\_addr\_bits)  
*Install handlers for an address family.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_get\\_parser](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family, [wolfentry\\_addr\\_family\\_parser\\_t](#) \*parser)  
*Retrieve the parsing handler for an address family.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_get\\_formatter](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family, [wolfentry\\_addr\\_family\\_formatter\\_t](#) \*formatter)  
*Retrieve the formatting handler for an address family.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_handler\\_remove\\_bynumber](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family\_bynumber, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*Remove the handlers for an address family.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_drop\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct [wolfentry\\_addr\\_family\\_bynumber](#) \*family\_bynumber, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*Release an address family record previously returned by [wolfentry\\_addr\\_family\\_ntop\(\)](#)*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_handler\\_remove\\_byname](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*family\_byname, int family\_byname\_len, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*Remove the handlers for an address family.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_pton](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*family\_name, int family\_name\_len, [wolfentry\\_addr\\_family\\_t](#) \*family\_number)  
*Look up an address family by name, returning its number.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_ntop](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family, struct [wolfentry\\_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 [wolfentry\\_addr\\_family\\_drop\\_reference\(\)](#), when done accessing `family_name`.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_addr\\_family\\_max\\_addr\\_bits](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_addr\\_family\\_t](#) family, [wolfentry\\_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 [wolfentry\\_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 [wolfentry\\_kv\\_type\\_t](#).*
- #define **WOLFSENTRY\_KV\_KEY\_LEN**(kv)  
*Evaluates to the length of the key of a [wolfentry\\_kv\\_pair](#).*
- #define **WOLFSENTRY\_KV\_KEY**(kv)  
*Evaluates to the key of a [wolfentry\\_kv\\_pair](#).*

- `#define WOLFSENTRY_KV_TYPE(kv)`  
*Evaluates to the type of a `wolfentry_kv_pair`, with flag bits masked out.*
- `#define WOLFSENTRY_KV_V_UINT(kv)`  
*Evaluates to the `uint64_t` value of a `wolfentry_kv_pair` of type `WOLFSENTRY_KV_UINT`.*
- `#define WOLFSENTRY_KV_V_SINT(kv)`  
*Evaluates to the `int64_t` value of a `wolfentry_kv_pair` of type `WOLFSENTRY_KV_INT`.*
- `#define WOLFSENTRY_KV_V_FLOAT(kv)`  
*Evaluates to the `double` value of a `wolfentry_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 `wolfentry_kv_pair` of type `WOLFSENTRY_KV_STRING`.*
- `#define WOLFSENTRY_KV_V_STRING(kv)`  
*Evaluates to the `char *` value of a `wolfentry_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 `wolfentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.*
- `#define WOLFSENTRY_KV_V_BYTES(kv)`  
*Evaluates to the `byte *` value of a `wolfentry_kv_pair` of type `WOLFSENTRY_KV_BYTES`.*
- `#define WOLFSENTRY_KV_V_JSON(kv)`  
*Evaluates to the `JSON_VALUE *` value of a `wolfentry_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 wolfentry_errcode_t(* wolfentry_kv_validator_t) (WOLFSENTRY_CONTEXT_ARGS_IN, struct wolfentry_kv_pair *kv)`

## Enumerations

- `enum wolfentry_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 wolfentry_errcode_t wolfentry_user_value_set_validator (WOLFSENTRY_CONTEXT_ARGS_IN, wolfentry_kv_validator_t validator, wolfentry_action_res_t *action_results)`  
*Install a supplied `wolfentry_kv_validator_t` to validate all user values before inserting them into the value table.*
- `WOLFSENTRY_API wolfentry_errcode_t wolfentry_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_get\\_type](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, [wolfentry\\_kv\\_type\\_t](#) \*type)

Returns the type of the value with the designated key, using [WOLFSENTRY\\_KV\\_TYPE\(\)](#).

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_delete](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len)

Deletes the value with the designated key.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_store\\_bool](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_get\\_bool](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, [wolfentry\\_kv\\_type\\_t](#) \*value)

Gets a [WOLFSENTRY\\_KV\\_TRUE](#) or [WOLFSENTRY\\_KV\\_FALSE](#) value with the designated key.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_get\\_string](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, const char \*\*value, int \*value\_len, struct wolfentry\_kv\_pair\_internal \*\*user↔\_value\_record)

Gets a [WOLFSENTRY\\_KV\\_STRING](#) value with the designated key.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_store\\_bytes\\_base64](#) (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\\_BYTES](#) value with the designated key and a base64-encoded value.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_get\\_bytes](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, const [byte](#) \*\*value, int \*value\_len, struct wolfentry\_kv\_pair\_internal \*\*user\_value\_record)  
*Gets a WOLFSENTRY\_KV\_BYTES value with the designated key.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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_document_parse()` (or built up programmatically with the [centijson\\_value.h](#) API).*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_get\\_json](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*key, int key\_len, [JSON\\_VALUE](#) \*\*value, struct wolfentry\_kv\_pair\_internal \*\*user\_value\_record)  
*Gets a WOLFSENTRY\_KV\_JSON value with the designated key.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_value\\_release\\_record](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_kv\_pair\_internal \*\*user\_value\_record)  
*Release a `user_value_record` from `wolfentry_user_value_get_string()`, `wolfentry_user_value_get_bytes` or `wolfentry_user_value_get_json()`.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_kv\\_pair\\_export](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_kv\_pair\_internal \*kv, const struct [wolfentry\\_kv\\_pair](#) \*\*kv\_exports)  
*Extract the struct `wolfentry_kv_pair` from a struct `wolfentry_kv_pair_internal`. Caller must have a shared or exclusive lock on the context.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_kv\\_type\\_to\\_string](#) ([wolfentry\\_kv\\_type\\_t](#) type, const char \*\*out)  
*Return a human-readable rendering of a `wolfentry_kv_type_t`.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_kv\\_render\\_value](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_kv\\_pair](#) \*kv, char \*out, int \*out\_len)  
*Render `kv` in human-readable form to caller-preallocated buffer `out`.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_start](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_seek\\_to\\_head](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_seek\\_to\\_tail](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_current](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_cursor \*cursor, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_prev](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_cursor \*cursor, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_next](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_cursor \*cursor, struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_user\\_values\\_iterate\\_end](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_cursor \*\*cursor)  
*End an iteration loop started with `wolfentry_user_values_iterate_start()`. Caller must have a lock on the context at entry.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 `wolfentry_errcode_t`(\* `wolfentry_make_id_cb_t`) (void \*context, `wolfentry_ent_id_t` \*id)

### Enumerations

- enum `wolfentry_object_type_t` {  
`WOLFENTRY_OBJECT_TYPE_UNINITED` ,  
`WOLFENTRY_OBJECT_TYPE_TABLE` ,  
`WOLFENTRY_OBJECT_TYPE_ACTION` ,  
`WOLFENTRY_OBJECT_TYPE_EVENT` ,  
`WOLFENTRY_OBJECT_TYPE_ROUTE` ,  
`WOLFENTRY_OBJECT_TYPE_KV` ,  
`WOLFENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER` ,  
`WOLFENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME` }

*enum for communicating the type of an object.*

### Functions

- WOLFENTRY\_API `wolfentry_object_type_t` `wolfentry_get_object_type` (const void \*object)  
*Get the object type from a wolfentry object pointer.*
- WOLFENTRY\_API `wolfentry_ent_id_t` `wolfentry_get_object_id` (const void \*object)  
*Get the ID from a wolfentry object pointer.*
- WOLFENTRY\_API `wolfentry_errcode_t` `wolfentry_table_ent_get_by_id` (`WOLFENTRY_CONTEXT_ARGS_IN`, `wolfentry_ent_id_t` id, struct `wolfentry_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 `wolfentry_object_checkout()` is called for the object.*
- WOLFENTRY\_API `wolfentry_errcode_t` `wolfentry_object_checkout` (`WOLFENTRY_CONTEXT_ARGS_IN`, void \*object)  
*Increment the refcount for an object, making it safe from deallocation until `wolfentry_object_release()`. Caller must have a context lock on entry.*
- WOLFENTRY\_API `wolfentry_errcode_t` `wolfentry_object_release` (`WOLFENTRY_CONTEXT_ARGS_IN`, void \*object, `wolfentry_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.*
- WOLFENTRY\_API `wolfentry_hitcount_t` `wolfentry_table_n_inserts` (struct `wolfentry_table_header` \*table)  
*Get the number of inserts into a table.*
- WOLFENTRY\_API `wolfentry_hitcount_t` `wolfentry_table_n_deletes` (struct `wolfentry_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 `wolfentry_object_type_t`

enum `wolfentry_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 <code>wolfentry_action</code> .               |
| WOLFSENTRY_OBJECT_TYPE_EVENT                     | Object is a struct <code>wolfentry_event</code> .                |
| WOLFSENTRY_OBJECT_TYPE_ROUTE                     | Object is a struct <code>wolfentry_route</code> .                |
| WOLFSENTRY_OBJECT_TYPE_KV                        | Object is a struct <code>wolfentry_kv_pair_internal</code> .     |
| WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_↔<br>BYNUMBER | Object is a struct <code>wolfentry_addr_family_bynumber</code> . |
| WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_↔<br>BYNAME   | Object is a struct <code>wolfentry_addr_family_byname</code> .   |

## 8.9.3 Function Documentation

### 8.9.3.1 `wolfentry_get_object_id()`

```
WOLFSENTRY_API wolfentry_ent_id_t wolfentry_get_object_id (
    const void * object )
```

Get the ID from a wolfentry object pointer.

## Parameters

|               |                         |
|---------------|-------------------------|
| <i>object</i> | a pointer to the object |
|---------------|-------------------------|

## Returns

the object ID, or WOLFSENTRY\_OBJECT\_TYPE\_UNINITED on error.

### 8.9.3.2 `wolfentry_get_object_type()`

```
WOLFSENTRY_API wolfentry_object_type_t wolfentry_get_object_type (
    const void * object )
```

Get the object type from a wolfentry object pointer.

## Parameters

|               |                         |
|---------------|-------------------------|
| <i>object</i> | 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

|              |                                  |
|--------------|----------------------------------|
| <i>table</i> | 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

|              |                                  |
|--------------|----------------------------------|
| <i>table</i> | 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 structs.
- **#define WOLFSENTRY\_CONTEXT\_SET\_ELEMENTS(s)**  
Counterpart to `WOLFSENTRY_CONTEXT_ELEMENTS` to access the `wolfentry` 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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_get_thread_deadline()` when thread has no deadline set. Not allowed as explicit values passed to `wolfentry_set_deadline_abs()` – use `wolfentry_clear_deadline()` to clear any deadline. Can be overridden with user settings.

- `#define WOLFENTRY_DEADLINE_NOW (-2)`

Value returned in `deadline->tv_sec` and `deadline->tv_nsec` by `wolfentry_get_thread_deadline()` when thread is in non-blocking mode. Not allowed as explicit values passed to `wolfentry_set_deadline_abs()` – use `wolfentry_set_deadline_rel_usecs(WOLFENTRY_CONTEXT_ARGS_OUT, 0)` to put thread in non-blocking mode. Can be overridden with user settings.

- `#define WOLFENTRY_THREAD_NO_ID 0`
- `#define WOLFENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER {0}`

## Enumerations

- enum `wolfentry_thread_flags_t` {  
`WOLFENTRY_THREAD_FLAG_NONE` ,  
`WOLFENTRY_THREAD_FLAG_DEADLINE` ,  
`WOLFENTRY_THREAD_FLAG_READONLY` }  
*wolfentry\_thread\_flags\_t flags are to be ORed together.*
- enum `wolfentry_lock_flags_t` {  
`WOLFENTRY_LOCK_FLAG_NONE` ,  
`WOLFENTRY_LOCK_FLAG_PSHARED` ,  
`WOLFENTRY_LOCK_FLAG_SHARED_ERROR_CHECKING` ,  
`WOLFENTRY_LOCK_FLAG_NONRECURSIVE_MUTEX` ,  
`WOLFENTRY_LOCK_FLAG_NONRECURSIVE_SHARED` ,  
`WOLFENTRY_LOCK_FLAG_GET_RESERVATION_TOO` ,  
`WOLFENTRY_LOCK_FLAG_TRY_RESERVATION_TOO` ,  
`WOLFENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO` ,  
`WOLFENTRY_LOCK_FLAG_AUTO_DOWNGRADE` ,  
`WOLFENTRY_LOCK_FLAG_RETAIN_SEMAPHORE` }  
*flags to pass to `wolfentry_lock_*` () functions, to be ORd together*

## Functions

- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_init_thread_context` (struct `wolfentry_thread_↵`  
context \*thread\_context, `wolfentry_thread_flags_t` init\_thread\_flags, void \*user\_context)  
*Initialize thread\_context according to init\_thread\_flags, storing user\_context for later retrieval with `wolfentry_get_thread_user_context()`.*
- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_alloc_thread_context` (struct `wolfentry_host_platform_interface`  
\*hpi, struct `wolfentry_thread_context` \*\*thread\_context, `wolfentry_thread_flags_t` init\_thread\_flags, void  
\*user\_context)  
*Allocate space for thread\_context using the allocator in hpi, then call `wolfentry_init_thread_context()`.*
- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_get_thread_id` (struct `wolfentry_thread_context`  
\*thread, `wolfentry_thread_id_t` \*id)  
*Write the `wolfentry_thread_id_t` of thread to id.*
- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_get_thread_user_context` (struct `wolfentry_↵`  
thread\_context \*thread, void \*\*user\_context)  
*Store to user\_context the pointer previously passed to `wolfentry_init_thread_context()`.*
- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_get_thread_deadline` (struct `wolfentry_thread_↵`  
context \*thread, struct `timespec` \*deadline)  
*Store the deadline for thread to deadline, or if the thread has no deadline set, store `WOLFENTRY_DEADLINE_NEVER`  
to `deadline->tv_sec` and `deadline->tv_nsec`.*
- WOLFENTRY\_API `wolfentry_errcode_t wolfentry_get_thread_flags` (struct `wolfentry_thread_context`  
\*thread, `wolfentry_thread_flags_t` \*thread\_flags)  
*Store the flags of thread to thread\_flags.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_destroy\_thread\_context** (struct [wolfentry\\_thread\\_context](#) \*thread\_context, [wolfentry\\_thread\\_flags\\_t](#) thread\_flags)  
*Perform final integrity checking on the thread state, and deallocate its ID.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_free\_thread\_context** (struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi, struct [wolfentry\\_thread\\_context](#) \*\*thread\_context, [wolfentry\\_thread\\_flags\\_t](#) thread\_flags)  
*Call [wolfentry\\_destroy\\_thread\\_context\(\)](#) on \*thread\_context, and if that succeeds, deallocate the thread object previously allocated by [wolfentry\\_alloc\\_thread\\_context\(\)](#).*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_set\_deadline\_rel\_usec** (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 [wolfentry\\_errcode\\_t](#) **wolfentry\_set\_deadline\_abs** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, time\_t epoch\_secs, long epoch\_nsec)  
*Set the thread deadline to the time identified by epoch\_secs and epoch\_nsec. The thread will not wait for a lock beyond that deadline.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_clear\_deadline** (WOLFSENTRY\_CONTEXT\_ARGS\_IN)  
*Clear any thread deadline previously set. On time-unbounded calls such as [wolfentry\\_lock\\_shared\(\)](#) and [wolfentry\\_lock\\_mutex\(\)](#), the thread will sleep until the lock is available.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_set\_thread\_readonly** (struct [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_set\_thread\_readwrite** (struct [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_init** (struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi, struct [wolfentry\\_thread\\_context](#) \*thread, struct [wolfentry\\_rwlock](#) \*lock, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*This initializes a semaphore lock structure created by the user.*
- WOLFSENTRY\_API size\_t **wolfentry\_lock\_size** (void)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_alloc** (struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi, struct [wolfentry\\_thread\\_context](#) \*thread, struct [wolfentry\\_rwlock](#) \*\*lock, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Allocates and initializes a semaphore lock structure for use with wolfSentry.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests a shared lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared\_abstimed** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests a shared lock with an absolute timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared\_timed** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests a shared lock with a relative timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests an exclusive lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex\_abstimed** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests an exclusive lock with an absolute timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex\_timed** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Requests an exclusive lock with a relative timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex2shared** (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Downgrade an exclusive lock to a shared lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Upgrade a shared lock to an exclusive lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_abstimed](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, const struct [timespec](#) \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_timed](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_reserve](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Attempt to reserve a upgrade of a shared lock to an exclusive lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_redeem](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Redeem a reservation of a lock upgrade from shared to exclusive.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_redeem\\_abstimed](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, const struct [timespec](#) \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Redeem a reservation of a lock upgrade from shared to exclusive with an absolute timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_redeem\\_timed](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Redeem a reservation of a lock upgrade from shared to exclusive with a relative timeout.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_shared2mutex\\_abandon](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Abandon a reservation of a lock upgrade from shared to exclusive.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_have\\_shared](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Check if the lock is held in shared state.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_have\\_mutex](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Check if the lock is held in exclusive state.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_have\\_either](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Check if the lock is held in either shared or exclusive state.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_have\\_shared2mutex\\_reservation](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Check if an upgrade reservation is held on the lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_get\\_flags](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) \*flags)  
*Extract the current flags from the lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_unlock](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Unlock a lock.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_destroy](#) (struct [wolfentry\\_rwlock](#) \*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Destroy a lock that was created with [wolfentry\\_lock\\_init\(\)](#)*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_lock\\_free](#) (struct [wolfentry\\_rwlock](#) \*\*lock, struct [wolfentry\\_thread\\_context](#) \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)  
*Destroy and free a lock that was created with [wolfentry\\_lock\\_alloc\(\)](#). The lock's pointer will also be set to NULL.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_mutex](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
*Calls [wolfentry\\_lock\\_mutex\(\)](#) on the context.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_mutex\\_abstimed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct timespec \*abs\_timeout)  
Calls [wolfentry\\_lock\\_mutex\\_abstimed\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_mutex\\_abstimed\\_ex](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)  
variant of [wolfentry\\_context\\_lock\\_mutex\\_abstimed\(\)](#) with a *flags* arg.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_mutex\\_timed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_time\\_t](#) max\_wait)  
Calls [wolfentry\\_lock\\_mutex\\_timed\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_mutex\\_timed\\_ex](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)  
variant of [wolfentry\\_context\\_lock\\_mutex\\_timed\(\)](#) with a *flags* arg.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_shared](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
Calls [wolfentry\\_lock\\_shared\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_shared\\_abstimed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct timespec \*abs\_timeout)  
Calls [wolfentry\\_lock\\_shared\\_abstimed\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_shared\\_with\\_reservation\\_abstimed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct timespec \*abs\_timeout)  
Calls [wolfentry\\_lock\\_shared\\_abstimed\(\)](#) on the context, with the [WOLFSENTRY\\_LOCK\\_FLAG\\_GET\\_RESERVATION\\_TOO](#) flag.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_shared\\_timed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_time\\_t](#) max\_wait)  
Calls [wolfentry\\_lock\\_shared\\_timed\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_lock\\_shared\\_with\\_reservation\\_timed](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [wolfentry\\_time\\_t](#) max\_wait)  
Calls [wolfentry\\_lock\\_shared\\_timed\(\)](#) on the context, with the [WOLFSENTRY\\_LOCK\\_FLAG\\_GET\\_RESERVATION\\_TOO](#) flag.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_unlock](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
Calls [wolfentry\\_lock\\_unlock\(\)](#) on the context.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_context\\_unlock\\_and\\_abandon\\_reservation](#) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#))  
Calls [wolfentry\\_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 [wolfentry\\_lock\\_flags\\_t](#)

enum [wolfentry\\_lock\\_flags\\_t](#)

flags to pass to [wolfentry\\_lock\\_\\*\(\)](#) functions, to be OR'd together

Enumerator

|  |   |
|--|---|
| <a href="#">WOLFSENTRY_LOCK_FLAG_NONE</a>    | Default lock behavior.  |
| <a href="#">WOLFSENTRY_LOCK_FLAG_PSHARED</a> | Initialize lock to be shared between processes (currently not used, only allowed by <a href="#">wolfentry_lock_init()</a> , and only functional on POSIX targets) |



## Enumerator

|  |   |
|--|---|
| WOLFSENTRY_LOCK_FLAG_SHARED_ERROR↔<br>_CHECKING  | Enables supplementary error checking on shared lock usage (not currently implemented)   |
| WOLFSENTRY_LOCK_FLAG_NONRECURSIVE↔<br>MUTEX      | Don't allow recursive mutex locking in this call.   |
| WOLFSENTRY_LOCK_FLAG_NONRECURSIVE↔<br>SHARED     | Don't allow recursive shared locking in this call.  |
| WOLFSENTRY_LOCK_FLAG_GET↔<br>RESERVATION_TOO     | If a shared lock is gotten in this call, require that a mutex upgrade reservation also be gotten.   |
| WOLFSENTRY_LOCK_FLAG_TRY↔<br>RESERVATION_TOO     | If a shared lock is gotten in this call, try to get a mutex upgrade reservation too.  |
| WOLFSENTRY_LOCK_FLAG_ABANDON↔<br>RESERVATION_TOO | In a call to <a href="#">wolfsentry_lock_unlock()</a> , 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 <a href="#">wolfsentry_lock_unlock()</a> , 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↔<br>SEMAPHORE        | For use in an interrupt handler: get an async-signal-safe mutex on the lock. Implicitly has <code>try</code> dynamics (immediate return).   |

## 8.10.2.2 wolfsentry\_thread\_flags\_t

```
enum wolfsentry_thread_flags_t
```

wolfsentry\_thread\_flags\_t flags are to be ORed 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**

|               |  |
|---------------|--|
| <i>hpi</i>    | the <a href="#">wolfsentry_host_platform_interface</a>                     |
| <i>thread</i> | pointer to the <a href="#">wolfsentry_thread_context</a>                   |
| <i>lock</i>   | a pointer to a pointer to a lock structure to be allocated and initialized |
| <i>flags</i>  | the initial <a href="#">wolfsentry_lock_flags_t</a>                        |

**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**

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                    |
| <i>thread</i> | pointer to the <a href="#">wolfsentry_thread_context</a> |
| <i>flags</i>  | optional <a href="#">wolfsentry_lock_flags_t</a>         |

**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

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to a pointer to the lock                   |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[wolfentry\\_lock\\_alloc](#)

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

#### 8.10.3.4 `wolfentry_lock_get_flags()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_get_flags (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    wolfentry_lock_flags_t * flags )
```

Extract the current flags from the lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

#### 8.10.3.5 `wolfentry_lock_have_either()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_have_either (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    wolfentry_lock_flags_t flags )
```

Check if the lock is held in either shared or exclusive state.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

When decoded using `WOLFENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFENTRY_SUCCESS_ID_HAVE_MUTEX` if it is a held mutex lock, `WOLFENTRY_SUCCESS_ID_HAVE_READ_LOCK` if it is a held shared lock, `WOLFENTRY_ERROR_ID_LACKING_READ_LOCK` if the lock is valid but not held by the designated thread, or `WOLFENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.6 `wolfentry_lock_have_mutex()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_have_mutex (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_lock_flags_t flags )
```

Check if the lock is held in exclusive state.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

When decoded using `WOLFENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFENTRY_SUCCESS_ID_HAVE_MUTEX` if it is a held mutex lock, `WOLFENTRY_ERROR_ID_LACKING_MUTEX` if the lock is not in mutex state, `WOLFENTRY_ERROR_ID_NOT_PERMITTED` if the mutex is held by another thread, or `WOLFENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.7 `wolfentry_lock_have_shared()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_have_shared (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_lock_flags_t flags )
```

Check if the lock is held in shared state.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

When decoded using `WOLFENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFENTRY_SUCCESS_ID` if it is a held shared lock, `WOLFENTRY_ERROR_ID_LACKING_READ_LOCK` if the lock is valid but not held by the designated thread, or `WOLFENTRY_ERROR_ID_INVALID_ARG` if the lock is not properly initialized.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.8 `wolfentry_lock_have_shared2mutex_reservation()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_have_shared2mutex_reservation (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_lock_flags_t flags )
```

Check if an upgrade reservation is held on the lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

When decoded using `WOLFENTRY_ERROR_DECODE_ERROR_CODE()`, `WOLFENTRY_ERROR_ID_OK` if it is shared lock. Or `WOLFENTRY_ERROR_ID_NOT_OK` if it is not a shared lock.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.9 `wolfentry_lock_init()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_init (
    struct wolfentry_host_platform_interface * hpi,
    struct wolfentry_thread_context * thread,
    struct wolfentry_rwlock * lock,
    wolfentry_lock_flags_t flags )
```

This initializes a semaphore lock structure created by the user.

**Parameters**

|               |   |
|---------------|---|
| <i>hpi</i>    | the <a href="#">wolfentry_host_platform_interface</a>   |
| <i>thread</i> | pointer to the <a href="#">wolfentry_thread_context</a> |
| <i>lock</i>   | a pointer to a lock structure to be initialized         |
| <i>flags</i>  | the initial <a href="#">wolfentry_lock_flags_t</a>      |

**Returns**

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[wolfentry\\_lock\\_alloc](#)

[wolfentry\\_lock\\_destroy](#)

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.10 wolfentry\_lock\_mutex()**

```
WOLFENTRY_API wolfentry\_errcode\_t wolfentry_lock_mutex (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry\_lock\_flags\_t flags )
```

Requests an exclusive lock.

**Parameters**

|               |   |
|---------------|---|
| <i>lock</i>   | a pointer to the lock                                   |
| <i>thread</i> | pointer to the <a href="#">wolfentry_thread_context</a> |
| <i>flags</i>  | optional <a href="#">wolfentry_lock_flags_t</a>         |

**Returns**

[WOLFENTRY\\_IS\\_SUCCESS\(ret\)](#) is true on success.

**See also**

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.11 wolfentry\_lock\_mutex2shared()**

```
WOLFENTRY_API wolfentry\_errcode\_t wolfentry_lock_mutex2shared (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry\_lock\_flags\_t flags )
```

Downgrade an exclusive lock to a shared lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.12 wolfentry\_lock\_mutex\_abstimed()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_mutex_abstimed (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    const struct timespec * abs_timeout,
    wolfentry_lock_flags_t flags )
```

Requests an exclusive lock with an absolute timeout.

## Parameters

|                    |  |
|--------------------|--|
| <i>lock</i>        | a pointer to the lock                                |
| <i>thread</i>      | pointer to the <code>wolfentry_thread_context</code> |
| <i>abs_timeout</i> | the absolute timeout for the lock                    |
| <i>flags</i>       | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.13 wolfentry\_lock\_mutex\_timed()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_mutex_timed (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_time_t max_wait,
    wolfentry_lock_flags_t flags )
```

Requests an exclusive lock with a relative timeout.

## Parameters

|                 |  |
|-----------------|--|
| <i>lock</i>     | a pointer to the lock                                |
| <i>thread</i>   | pointer to the <code>wolfentry_thread_context</code> |
| <i>max_wait</i> | how long to wait for the timeout                     |
| <i>flags</i>    | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

**8.10.3.14 wolfentry\_lock\_shared()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    wolfentry_lock_flags_t flags )
```

Requests a shared lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

**8.10.3.15 wolfentry\_lock\_shared2mutex()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    wolfentry_lock_flags_t flags )
```

Upgrade a shared lock to an exclusive lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.16 wolfentry\_lock\_shared2mutex\_abandon()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_abandon (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    wolfentry_lock_flags_t flags )
```

Abandon a reservation of a lock upgrade from shared to exclusive.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

**8.10.3.17 wolfentry\_lock\_shared2mutex\_abstimed()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_abstimed (  
    struct wolfentry_rwlock * lock,  
    struct wolfentry_thread_context * thread,  
    const struct timespec * abs_timeout,  
    wolfentry_lock_flags_t flags )
```

Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.

## Parameters

|                    |  |
|--------------------|--|
| <i>lock</i>        | a pointer to the lock                                |
| <i>thread</i>      | pointer to the <code>wolfentry_thread_context</code> |
| <i>abs_timeout</i> | the absolute timeout for the lock                    |
| <i>flags</i>       | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

### 8.10.3.18 `wolfentry_lock_shared2mutex_redeem()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_redeem (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

### 8.10.3.19 `wolfentry_lock_shared2mutex_redeem_abstimed()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_redeem_abstimed (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    const struct timespec * abs_timeout,
    wolfentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive with an absolute timeout.



## Parameters

|                    |  |
|--------------------|--|
| <i>lock</i>        | a pointer to the lock                                |
| <i>thread</i>      | pointer to the <code>wolfentry_thread_context</code> |
| <i>abs_timeout</i> | the absolute timeout for the lock                    |
| <i>flags</i>       | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

**8.10.3.20 wolfentry\_lock\_shared2mutex\_redeem\_timed()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_redeem_timed (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_time_t max_wait,
    wolfentry_lock_flags_t flags )
```

Redeem a reservation of a lock upgrade from shared to exclusive with a relative timeout.

## Parameters

|                 |  |
|-----------------|--|
| <i>lock</i>     | a pointer to the lock                                |
| <i>thread</i>   | pointer to the <code>wolfentry_thread_context</code> |
| <i>max_wait</i> | how long to wait for the timeout                     |
| <i>flags</i>    | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

`WOLFENTRY_ERROR_DECODE_ERROR_CODE`

**8.10.3.21 wolfentry\_lock\_shared2mutex\_reserve()**

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_reserve (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_lock_flags_t flags )
```

Attempt to reserve a upgrade of a shared lock to an exclusive lock.

## Parameters

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                                |
| <i>thread</i> | pointer to the <code>wolfentry_thread_context</code> |
| <i>flags</i>  | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[wolfentry\\_lock\\_shared2mutex\\_redeem](#)  
[wolfentry\\_lock\\_shared2mutex\\_redeem\\_abstimed](#)  
[wolfentry\\_lock\\_shared2mutex\\_redeem\\_timed](#)  
[wolfentry\\_lock\\_shared2mutex\\_abandon](#)  
[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.22 `wolfentry_lock_shared2mutex_timed()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared2mutex_timed (
    struct wolfentry_rwlock * lock,
    struct wolfentry_thread_context * thread,
    wolfentry_time_t max_wait,
    wolfentry_lock_flags_t flags )
```

Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.

## Parameters

|                 |  |
|-----------------|--|
| <i>lock</i>     | a pointer to the lock                                |
| <i>thread</i>   | pointer to the <code>wolfentry_thread_context</code> |
| <i>max_wait</i> | how long to wait for the timeout                     |
| <i>flags</i>    | optional <code>wolfentry_lock_flags_t</code>         |

## Returns

`WOLFENTRY_IS_SUCCESS(ret)` is true on success.

## See also

[WOLFENTRY\\_ERROR\\_DECODE\\_ERROR\\_CODE](#)

8.10.3.23 `wolfentry_lock_shared_abstimed()`

```
WOLFENTRY_API wolfentry_errcode_t wolfentry_lock_shared_abstimed (
    struct wolfentry_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

|                    |  |
|--------------------|--|
| <i>lock</i>        | a pointer to the lock                    |
| <i>thread</i>      | pointer to the wolfsentry_thread_context |
| <i>abs_timeout</i> | the absolute timeout for the lock        |
| <i>flags</i>       | 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

|                 |  |
|-----------------|--|
| <i>lock</i>     | a pointer to the lock                    |
| <i>thread</i>   | pointer to the wolfsentry_thread_context |
| <i>max_wait</i> | how long to wait for the timeout         |
| <i>flags</i>    | 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

|               |  |
|---------------|--|
| <i>lock</i>   | a pointer to the lock                    |
| <i>thread</i> | pointer to the wolfsentry_thread_context |
| <i>flags</i>  | 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 \* **wolfentry\_malloc** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, size\_t size)  
*Allocate size bytes using the malloc configured in the wolfSentry context.*
- WOLFSENTRY\_API\_VOID **wolfentry\_free** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, void \*ptr)  
*Free ptr using the free configured in the wolfSentry context.*
- WOLFSENTRY\_API void \* **wolfentry\_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 \* **wolfentry\_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 **wolfentry\_free\_aligned** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, void \*ptr)  
*Free ptr, previously allocated with wolfentry\_memalign(), using the free\_aligned configured in the wolfSentry context.*
- WOLFSENTRY\_API int **\_wolfentry\_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 **wolfentry\_allocator** \* **wolfentry\_get\_allocator** (struct wolfentry\_context \*wolfentry)  
*Return a pointer to the wolfentry\_allocator associated with the supplied wolfentry\_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 **wolfentry\_timecbs**  
*Struct for passing shims that abstract the native implementation of time functions.*

### Typedefs

- typedef **wolfentry\_errcode\_t**(\* **wolfentry\_get\_time\_cb\_t**) (void \*context, **wolfentry\_time\_t** \*ts)  
*Pointer to function that returns time denominated in wolfentry\_time\_t. Takes an initial context arg, which can be ignored.*
- typedef **wolfentry\_time\_t**(\* **wolfentry\_diff\_time\_cb\_t**) (**wolfentry\_time\_t** earlier, **wolfentry\_time\_t** later)  
*Pointer to function that subtracts earlier from later, returning the result.*
- typedef **wolfentry\_time\_t**(\* **wolfentry\_add\_time\_cb\_t**) (**wolfentry\_time\_t** start\_time, **wolfentry\_time\_t** time\_interval)  
*Pointer to function that adds two wolfentry\_time\_t times, returning the result.*
- typedef **wolfentry\_errcode\_t**(\* **wolfentry\_to\_epoch\_time\_cb\_t**) (**wolfentry\_time\_t** when, time\_t \*epoch\_secs, long \*epoch\_nsecs)  
*Pointer to function that converts a wolfentry\_time\_t to seconds and nanoseconds since midnight UTC, 1970-Jan-1.*
- typedef **wolfentry\_errcode\_t**(\* **wolfentry\_from\_epoch\_time\_cb\_t**) (time\_t epoch\_secs, long epoch\_nsecs, **wolfentry\_time\_t** \*when)  
*Pointer to function that converts seconds and nanoseconds since midnight UTC, 1970-Jan-1, to a wolfentry\_time\_t.*

- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_interval\\_to\\_seconds\\_cb\\_t](#)) ([wolfentry\\_time\\_t](#) howlong, time\_t \*howlong\_secs, long \*howlong\_nsecs)  
*Pointer to function that converts a [wolfentry\\_time\\_t](#) expressing an interval to the corresponding seconds and nanoseconds.*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_interval\\_from\\_seconds\\_cb\\_t](#)) (time\_t howlong\_secs, long howlong\_nsecs, [wolfentry\\_time\\_t](#) \*howlong)  
*Pointer to function that converts seconds and nanoseconds expressing an interval to the corresponding [wolfentry\\_time\\_t](#).*

## Functions

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_time\\_now\\_plus\\_delta](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) td, [wolfentry\\_time\\_t](#) \*res)  
*Generate a [wolfentry\\_time\\_t](#) at a given offset from current time.*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_time\\_to\\_timespec](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) t, struct [timespec](#) \*ts)  
*Convert a [wolfentry\\_time\\_t](#) to a struct [timespec](#).*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_time\\_now\\_plus\\_delta\\_timespec](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) td, struct [timespec](#) \*ts)  
*Generate a struct [timespec](#) at a given offset, supplied as [wolfentry\\_time\\_t](#), from current time.*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_get\\_time](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) \*time\_p)  
*Get current time as [wolfentry\\_time\\_t](#).*
- WOLFENTRY\_API [wolfentry\\_time\\_t](#) [wolfentry\\_diff\\_time](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) later, [wolfentry\\_time\\_t](#) earlier)  
*Compute the interval between *later* and *earlier*, using [wolfentry\\_time\\_t](#).*
- WOLFENTRY\_API [wolfentry\\_time\\_t](#) [wolfentry\\_add\\_time](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) start\_time, [wolfentry\\_time\\_t](#) time\_interval)  
*Compute the time *time\_interval* after *start\_time*, using [wolfentry\\_time\\_t](#).*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_to\\_epoch\\_time](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) when, time\_t \*epoch\_secs, long \*epoch\_nsecs)  
*Convert a [wolfentry\\_time\\_t](#) to seconds and nanoseconds since 1970-Jan-1 0:00 UTC.*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_from\\_epoch\\_time](#) (struct [wolfentry\\_context](#) \*wolfentry, time\_t epoch\_secs, long epoch\_nsecs, [wolfentry\\_time\\_t](#) \*when)  
*Convert seconds and nanoseconds since 1970-Jan-1 0:00 UTC to a [wolfentry\\_time\\_t](#).*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_interval\\_to\\_seconds](#) (struct [wolfentry\\_context](#) \*wolfentry, [wolfentry\\_time\\_t](#) howlong, time\_t \*howlong\_secs, long \*howlong\_nsecs)  
*Convert an interval in [wolfentry\\_time\\_t](#) to seconds and nanoseconds.*
- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_interval\\_from\\_seconds](#) (struct [wolfentry\\_context](#) \*wolfentry, time\_t howlong\_secs, long howlong\_nsecs, [wolfentry\\_time\\_t](#) \*howlong)  
*Convert an interval in seconds and nanoseconds to [wolfentry\\_time\\_t](#).*
- WOLFENTRY\_API struct [wolfentry\\_timecbs](#) \* [wolfentry\\_get\\_timecbs](#) (struct [wolfentry\\_context](#) \*wolfentry)  
*Return the active time handlers from the supplied context.*

## 8.12.1 Detailed Description

## 8.13 Semaphore Function Callbacks

### Data Structures

- struct [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_install\_lwip\_filter\_ethernet\_callback** ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [packet\\_filter\\_event\\_mask\\_t](#) ethernet\_mask)  
*Install wolfSentry callbacks into lwIP for ethernet (layer 2) filtering.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_install\_lwip\_filter\_ip\_callbacks** ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [packet\\_filter\\_event\\_mask\\_t](#) ip\_mask)  
*Install wolfSentry callbacks into lwIP for IPv4/IPv6 (layer 3) filtering.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_install\_lwip\_filter\_icmp\_callbacks** ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [packet\\_filter\\_event\\_mask\\_t](#) icmp\_mask)  
*Install wolfSentry callbacks into lwIP for ICMP filtering.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_install\_lwip\_filter\_tcp\_callback** ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [packet\\_filter\\_event\\_mask\\_t](#) tcp\_mask)  
*Install wolfSentry callbacks into lwIP for TCP (layer 4) filtering.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_install\_lwip\_filter\_udp\_callback** ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), [packet\\_filter\\_event\\_mask\\_t](#) udp\_mask)  
*Install wolfSentry callbacks into lwIP for UDP (layer 4) filtering.*
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_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](#) **wolfentry\_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 {  
**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:: { ... } **automaton**
- unsigned **state**
- unsigned **substate**
- `uint32_t` **codepoint** [2]
- unsigned char \* **buf**
- `size_t` **buf\_used**
- `size_t` **buf\_allocated**
- `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 wolfentry\_allocator Struct Reference

Struct for passing shims that abstract the native implementation of the heap allocator.

```
#include <wolfentry.h>
```

### Data Fields

- void \* **context**  
*A user-supplied opaque handle to be passed as the first arg to all callbacks. Can be null.*
- [wolfentry\\_malloc\\_cb\\_t](#) **malloc**  
*Required pointer.*
- [wolfentry\\_free\\_cb\\_t](#) **free**  
*Required pointer.*
- [wolfentry\\_realloc\\_cb\\_t](#) **realloc**  
*Required pointer.*
- [wolfentry\\_memalign\\_cb\\_t](#) **memalign**  
*Optional pointer. Required only if a struct [wolfentry\\_eventconfig](#) is passed in (e.g. to [wolfentry\\_init\(\)](#)) with a nonzeroroute\_private\_data\_alignment`.*
- [wolfentry\\_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 wolfentry\_build\_settings Struct Reference

struct for passing the build version and configuration

```
#include <wolfentry_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 wolfentry_build_settings::config
```

Must be initialized to `__wolfentry_config`.

### 9.8.2.2 version

```
uint32_t wolfsentry_build_settings::version
```

Must be initialized to WOLFSENTRY\_VERSION.

## 9.9 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. Useful primarily in **config** clauses of events (see **events** below). 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.*

### 9.9.1 Detailed Description

struct for representing event configuration

## 9.10 wolfentry\_host\_platform\_interface Struct Reference

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

```
#include <wolfentry.h>
```

### Data Fields

- struct [wolfentry\\_build\\_settings](#) caller\_build\_settings
- struct [wolfentry\\_allocator](#) allocator
- struct [wolfentry\\_timecbs](#) timecbs
- struct [wolfentry\\_semcbs](#) semcbs

### 9.10.1 Detailed Description

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

### 9.10.2 Field Documentation

#### 9.10.2.1 allocator

```
struct wolfentry\_allocator wolfentry_host_platform_interface::allocator
```

Either all-null, or initialized as described for [wolfentry\\_allocator](#).

#### 9.10.2.2 caller\_build\_settings

```
struct wolfentry\_build\_settings wolfentry_host_platform_interface::caller_build_settings
```

Must be initialized as described for [wolfentry\\_build\\_settings](#).

#### 9.10.2.3 semcbs

```
struct wolfentry\_semcbs wolfentry_host_platform_interface::semcbs
```

Either all-null, or initialized as described for [wolfentry\\_semcbs](#).

#### 9.10.2.4 timecbs

```
struct wolfentry\_timecbs wolfentry_host_platform_interface::timecbs
```

Either all-null, or initialized as described for [wolfentry\\_timecbs](#).

## 9.11 wolfentry\_kv\_pair Struct Reference

public structure for passing user-defined values in/out of wolfSentry

```
#include <wolfentry.h>
```

### Data Fields

- int **key\_len**  
*the length of the key, not including the terminating null*
- [wolfentry\\_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*
- [byte](#) **b** []  
*A flexible-length buffer to hold the key, and for strings and bytes, the data.*

### 9.11.1 Detailed Description

public structure for passing user-defined values in/out of wolfSentry

### 9.11.2 Field Documentation

#### 9.11.2.1 b

```
byte wolfentry_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.

## 9.12 wolfentry\_route\_endpoint Struct Reference

struct for exporting socket addresses, with fixed-length fields

```
#include <wolfentry.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\\_NUMBER](#) 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.*

**9.12.1 Detailed Description**

struct for exporting socket addresses, with fixed-length fields

**9.13 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.*

### 9.13.1 Detailed Description

struct for exporting a route for access by applications

## 9.14 wolfentry\_route\_metadata\_exports Struct Reference

struct for exporting route metadata for access by applications

```
#include <wolfentry.h>
```

### Data Fields

- [wolfentry\\_time\\_t insert\\_time](#)  
*The time the route was inserted.*
- [wolfentry\\_time\\_t last\\_hit\\_time](#)  
*The most recent time the route was matched.*
- [wolfentry\\_time\\_t last\\_penaltybox\\_time](#)  
*The most recent time the route had its [WOLFENTRY\\_ROUTE\\_FLAG\\_PENALTYBOXED](#) flag set.*
- [wolfentry\\_time\\_t purge\\_after](#)  
*The expiration time of the route, if any (persistent routes have 0 here)*
- [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)*
- [wolfentry\\_hitcount\\_t hit\\_count](#)  
*The lifetime match count (informational/approximate, and only maintained if the [WOLFENTRY\\_ROUTE\\_FLAG\\_DONT\\_COUNT\\_HITS](#) flag is clear)*

### 9.14.1 Detailed Description

struct for exporting route metadata for access by applications

## 9.15 wolfentry\_semcbs Struct Reference

Struct for passing shims that abstract the native implementation of counting semaphores.

```
#include <wolfentry.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.*

#### 9.15.1 Detailed Description

Struct for passing shims that abstract the native implementation of counting semaphores.

## 9.16 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.*

#### 9.16.1 Detailed Description

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

## 9.17 wolfentry\_thread\_context\_public Struct Reference

Right-sized, right-aligned opaque container for thread state.

```
#include <wolfentry_settings.h>
```

### Data Fields

- uint64\_t **opaque** [8]

### 9.17.1 Detailed Description

Right-sized, right-aligned opaque container for thread state.

## 9.18 wolfentry\_timecbs Struct Reference

Struct for passing shims that abstract the native implementation of time functions.

```
#include <wolfentry.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.*
- [wolfentry\\_get\\_time\\_cb\\_t](#) **get\_time**  
*Required pointer.*
- [wolfentry\\_diff\\_time\\_cb\\_t](#) **diff\_time**  
*Required pointer.*
- [wolfentry\\_add\\_time\\_cb\\_t](#) **add\_time**  
*Required pointer.*
- [wolfentry\\_to\\_epoch\\_time\\_cb\\_t](#) **to\_epoch\_time**  
*Required pointer.*
- [wolfentry\\_from\\_epoch\\_time\\_cb\\_t](#) **from\_epoch\_time**  
*Required pointer.*
- [wolfentry\\_interval\\_to\\_seconds\\_cb\\_t](#) **interval\_to\_seconds**  
*Required pointer.*
- [wolfentry\\_interval\\_from\\_seconds\\_cb\\_t](#) **interval\_from\_seconds**  
*Required pointer.*

### 9.18.1 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 parsing 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_MAINTAINDICTIONORDER 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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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  */
00089 #define JSON_ERR_OUTOFMEMORY (-2)
00090 #define JSON_ERR_SYNTAX (-4) /* Generic syntax error. (More specific error codes
00091  are preferred.) */
00092 #define JSON_ERR_BADCLOSER (-5) /* Mismatch in brackets (e.g. "{" or "[ ]") */
00093 #define JSON_ERR_BADROOTTYPE (-6) /* Root type not allowed by CONFIG::flags. */
00094 #define JSON_ERR_EXPECTEDVALUE (-7) /* Something unexpected where value has to be. */
00095 #define JSON_ERR_EXPECTEDKEY (-8) /* Something unexpected where key has to be. */
00096 #define JSON_ERR_EXPECTEDVALUEORCLOSER (-9) /* Something unexpected where value or array/object
00097  closer has to be. */
00098 #define JSON_ERR_EXPECTEDKEYORCLOSER (-10) /* Something unexpected where key or array/object
00099  closer has to be. */
00100 #define JSON_ERR_EXPECTEDCOLON (-11) /* Something unexpected where colon has to be. */
00101 #define JSON_ERR_EXPECTEDCOMMAORCLOSER (-12) /* Something unexpected where comma or array/object
00102  has to be. */
00103 #define JSON_ERR_EXPECTEDEOF (-13) /* Something unexpected where end-of-file has to be.
00104  */
00105 #define JSON_ERR_MAXTOTALLEN (-14) /* Reached JSON_CONFIG::max_total_len */
00106 #define JSON_ERR_MAXTOTALVALUES (-15) /* Reached JSON_CONFIG::max_total_values */
00107 #define JSON_ERR_MAXNESTINGLEVEL (-16) /* Reached JSON_CONFIG::max_nesting_level */
00108 #define JSON_ERR_MAXNUMBERLEN (-17) /* Reached JSON_CONFIG::max_number_len */
00109 #define JSON_ERR_MAXSTRINGLEN (-18) /* Reached JSON_CONFIG::max_string_len */
00110 #define JSON_ERR_MAXKEYLEN (-19) /* Reached JSON_CONFIG::max_key_len */
00111 #define JSON_ERR_UNCLOSEDSTRING (-20) /* Unclosed string */
00112 #define JSON_ERR_UNESCAPEDCONTROL (-21) /* Unescaped control character (in a string) */
00113 #define JSON_ERR_INVALIDESCAPE (-22) /* Invalid/unknown escape sequence (in a string) */
00114 #define JSON_ERR_INVALIDUTF8 (-23) /* Invalid UTF-8 (in a string) */
00115 #define JSON_ERR_NOT_INITED (-24) /* Attempt to access an uninit JSON_PARSER or
00116  JSON_DOM_PARSER. */
00117
00118
00119 /* Bits for JSON_CONFIG::flags.
00120  */
00121 #define JSON_NONULLASROOT 0x0001U /* Disallow null to be root value */
00122 #define JSON_NOBOOLASROOT 0x0002U /* Disallow false or true to be root value */
00123 #define JSON_NONUMBERASROOT 0x0004U /* Disallow number to be root value */
00124 #define JSON_NOSTRINGASROOT 0x0008U /* Disallow string to be root value */
00125 #define JSON_NOARRAYASROOT 0x0010U /* Disallow array to be root value */
00126 #define JSON_NOOBJECTASROOT 0x0020U /* Disallow object to be root value */
00127
00128 #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
00127 (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
00129 (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 wolfentry_allocator *allocator;
00186 #ifdef WOLFSENTRY_THREADSAFE
00187     struct wolfentry_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 {

```

```

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_allocated;
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 wolfentry_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 wolfentry_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
00355   write_func, void* user_data);
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 "wolfentry.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 wolfentry_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 loses 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 } }
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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 if 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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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 wolfentry_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,
int (*visit_func)(const JSON_VALUE*, JSON_VALUE*, void*), void* ctx);
00524 WOLFSENTRY_API int json_value_dict_walk_sorted(const JSON_VALUE* v,
int (*visit_func)(const JSON_VALUE*, JSON_VALUE*, void*), void* ctx);
00525
00526 /* Remove and destroy all members (recursively).
00527  */
00528 WOLFSENTRY_API int json_value_dict_clean(

```

```

00531 #ifdef WOLFSENTRY
00532     WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct wolfentry_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 wolfentry_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 wolfentry/wolfentry.h File Reference

The main include file for wolfSentry applications.

```

#include <wolfentry/wolfentry_settings.h>
#include <wolfentry/wolfentry_af.h>
#include <wolfentry/wolfentry_errcodes.h>
#include <wolfentry/centijson_dom.h>
#include <wolfentry/wolfentry_util.h>

```

### Data Structures

- struct [wolfentry\\_allocator](#)  
*Struct for passing shims that abstract the native implementation of the heap allocator.*
- struct [wolfentry\\_timecbs](#)  
*Struct for passing shims that abstract the native implementation of time functions.*
- struct [wolfentry\\_semcbs](#)  
*Struct for passing shims that abstract the native implementation of counting semaphores.*
- struct [wolfentry\\_host\\_platform\\_interface](#)  
*struct for passing shims that abstract native implementations of the heap allocator, time functions, and semaphores*
- struct [wolfentry\\_route\\_endpoint](#)  
*struct for exporting socket addresses, with fixed-length fields*
- struct [wolfentry\\_route\\_metadata\\_exports](#)  
*struct for exporting route metadata for access by applications*
- struct [wolfentry\\_route\\_exports](#)  
*struct for exporting a route for access by applications*
- struct [wolfentry\\_sockaddr](#)  
*struct for passing socket addresses into wolfentry\_route\_\*() API routines*
- struct [wolfentry\\_eventconfig](#)  
*struct for representing event configuration*
- struct [wolfentry\\_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 structs.*
- **#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 ( $\leftrightarrow$  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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_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 bits in `wolfentry_action_res_t`.

- **#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 `wolfentry_route_table_default_policy_set()`

- **#define WOLFSENTRY\_ROUTE\_WILDCARD\_FLAGS**

Bit mask for the wildcard bits in a `wolfentry_route_flags_t`.

- **#define WOLFSENTRY\_ROUTE\_IMMUTABLE\_FLAGS**

Bit mask for the bits in a `wolfentry_route_flags_t` that can't change after the implicated route has been inserted in the route table.

- **#define WOLFSENTRY\_SOCKADDR(n)**

Macro to instantiate a `wolfentry_sockaddr` with an `addr` field sized to hold `n` bits of address data. Cast to `struct wolfentry_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 `wolfentry_kv_type_t`.

- **#define WOLFSENTRY\_KV\_KEY\_LEN(kv)**

Evaluates to the length of the key of a `wolfentry_kv_pair`.

- **#define WOLFSENTRY\_KV\_KEY(kv)**

Evaluates to the key of a `wolfentry_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 [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_to\\_epoch\\_time\\_cb\\_t](#)) ([wolfentry\\_time\\_t](#) when, time\_t \*epoch\_secs, long \*epoch\_nsecs)  
*Pointer to function that converts a [wolfentry\\_time\\_t](#) to seconds and nanoseconds since midnight UTC, 1970-Jan-1.*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_from\\_epoch\\_time\\_cb\\_t](#)) (time\_t epoch\_secs, long epoch\_nsecs, [wolfentry\\_time\\_t](#) \*when)  
*Pointer to function that converts seconds and nanoseconds since midnight UTC, 1970-Jan-1, to a [wolfentry\\_time\\_t](#).*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_interval\\_to\\_seconds\\_cb\\_t](#)) ([wolfentry\\_time\\_t](#) howlong, time\_t \*howlong\_secs, long \*howlong\_nsecs)  
*Pointer to function that converts a [wolfentry\\_time\\_t](#) expressing an interval to the corresponding seconds and nanoseconds.*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_interval\\_from\\_seconds\\_cb\\_t](#)) (time\_t howlong\_secs, long howlong\_nsecs, [wolfentry\\_time\\_t](#) \*howlong)  
*Pointer to function that converts seconds and nanoseconds expressing an interval to the corresponding [wolfentry\\_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 [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_action\\_callback\\_t](#)) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct [wolfentry\\_action](#) \*action, void \*handler\_arg, void \*caller\_arg, const struct [wolfentry\\_event](#) \*trigger\_event, [wolfentry\\_action\\_type\\_t](#) action\_type, const struct [wolfentry\\_route](#) \*trigger\_route, struct [wolfentry\\_route\\_table](#) \*route\_table, struct [wolfentry\\_route](#) \*rule\_route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)  
*A callback that is triggered when an action is taken.*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_make\\_id\\_cb\\_t](#)) (void \*context, [wolfentry\\_ent\\_id\\_t](#) \*id)
- typedef void(\* [wolfentry\\_cleanup\\_callback\\_t](#)) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*cleanup\_arg)  
*Function type to pass to [wolfentry\\_cleanup\\_push\(\)](#)*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_addr\\_family\\_parser\\_t](#)) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), const char \*addr\_text, int addr\_text\_len, [byte](#) \*addr\_internal, [wolfentry\\_addr\\_bits\\_t](#) \*addr\_internal\_bits)  
*Function type for parsing handler, to pass to [wolfentry\\_addr\\_family\\_handler\\_install\(\)](#)*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_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 [wolfentry\\_addr\\_family\\_handler\\_install\(\)](#)*
- typedef [wolfentry\\_errcode\\_t](#)(\* [wolfentry\\_kv\\_validator\\_t](#)) ([WOLFSENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_kv\\_pair](#) \*kv)

## Enumerations

- enum [wolfentry\\_init\\_flags\\_t](#) {  
[WOLFSENTRY\\_INIT\\_FLAG\\_NONE](#) ,  
[WOLFSENTRY\\_INIT\\_FLAG\\_LOCK\\_SHARED\\_ERROR\\_CHECKING](#) }  
*flags to pass to [wolfentry\\_init\\_ex\(\)](#), to be ORed together.*
- enum [wolfentry\\_thread\\_flags\\_t](#) {  
[WOLFSENTRY\\_THREAD\\_FLAG\\_NONE](#) ,  
[WOLFSENTRY\\_THREAD\\_FLAG\\_DEADLINE](#) ,  
[WOLFSENTRY\\_THREAD\\_FLAG\\_READONLY](#) }  
*[wolfentry\\_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 OR'd 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_USER_BASE }
```

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

- enum `wolfentry_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_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 `wolfentry_format_flags_t` {  
`WOLFSENTRY_FORMAT_FLAG_NONE` ,  
`WOLFSENTRY_FORMAT_FLAG_ALWAYS_NUMERIC` }

*bit field with options for rendering*

- enum `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_context_clone()` and other cloning functions.*

- enum `wolfentry_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\_usec** ([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 [wolfentry\\_errcode\\_t](#) **wolfentry\_set\_thread\_readwrite** (struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_init** (struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi, struct wolfentry\_thread\_context \*thread, struct wolfentry\_rwlock \*lock, [wolfentry\\_lock\\_flags\\_t](#) flags)

*This initializes a semaphore lock structure created by the user.*

- WOLFSENTRY\_API [size\\_t](#) **wolfentry\_lock\_size** (void)
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_alloc** (struct [wolfentry\\_host\\_platform\\_interface](#) \*hpi, struct wolfentry\_thread\_context \*thread, struct wolfentry\_rwlock \*\*lock, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Allocates and initializes a semaphore lock structure for use with wolfSentry.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests a shared lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared\_abstimed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests a shared lock with an absolute timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared\_timed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests a shared lock with a relative timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests an exclusive lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex\_abstimed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests an exclusive lock with an absolute timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex\_timed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Requests an exclusive lock with a relative timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_mutex2shared** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Downgrade an exclusive lock to a shared lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Upgrade a shared lock to an exclusive lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex\_abstimed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Attempt to upgrade a shared lock to an exclusive lock with an absolute timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex\_timed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_time\\_t](#) max\_wait, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Attempt to upgrade a shared lock to an exclusive lock with a relative timeout.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex\_reserve** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Attempt to reserve a upgrade of a shared lock to an exclusive lock.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex\_redeem** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, [wolfentry\\_lock\\_flags\\_t](#) flags)

*Redeem a reservation of a lock upgrade from shared to exclusive.*

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_lock\_shared2mutex\_redeem\_abstimed** (struct wolfentry\_rwlock \*lock, struct wolfentry\_thread\_context \*thread, const struct timespec \*abs\_timeout, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_from\_epoch\_time** (struct wolfentry\_context \*wolfentry, time\_t epoch\_secs, long epoch\_nsecs, [wolfentry\\_time\\_t](#) \*when)

Convert seconds and nanoseconds since 1970-Jan-1 0:00 UTC to a [wolfentry\\_time\\_t](#).

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_interval\_to\_seconds** (struct wolfentry\_context \*wolfentry, [wolfentry\\_time\\_t](#) howlong, time\_t \*howlong\_secs, long \*howlong\_nsecs)

Convert an interval in [wolfentry\\_time\\_t](#) to seconds and nanoseconds.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_interval\_from\_seconds** (struct wolfentry\_context \*wolfentry, time\_t howlong\_secs, long howlong\_nsecs, [wolfentry\\_time\\_t](#) \*howlong)

Convert an interval in seconds and nanoseconds to [wolfentry\\_time\\_t](#).

- WOLFSENTRY\_API struct [wolfentry\\_timecbs](#) \* **wolfentry\_get\_timecbs** (struct wolfentry\_context \*wolfentry)

Return the active time handlers from the supplied context.

- WOLFSENTRY\_API void \* **wolfentry\_malloc** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, size\_t size)

Allocate size bytes using the malloc configured in the wolfSentry context.

- WOLFSENTRY\_API\_VOID **wolfentry\_free** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, void \*ptr)

Free ptr using the free configured in the wolfSentry context.

- WOLFSENTRY\_API void \* **wolfentry\_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 \* **wolfentry\_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 **wolfentry\_free\_aligned** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, void \*ptr)

Free ptr, previously allocated with [wolfentry\\_memalign\(\)](#), using the free\_aligned configured in the wolfSentry context.

- WOLFSENTRY\_API int **\_wolfentry\_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 [wolfentry\\_allocator](#) \* **wolfentry\_get\_allocator** (struct wolfentry\_context \*wolfentry)

Return a pointer to the [wolfentry\\_allocator](#) associated with the supplied wolfentry\_context, mainly for passing to [json\\_init\(\)](#), [json\\_parse\(\)](#), [json\\_value\\_\\*](#)(), and [json\\_dom\\_\\*](#)().

- WOLFSENTRY\_API const char \* **wolfentry\_action\_res\_assoc\_by\_flag** ([wolfentry\\_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 [wolfentry\\_errcode\\_t](#) **wolfentry\_action\_res\_assoc\_by\_name** (const char \*bit\_name, size\_t bit\_name\_len, [wolfentry\\_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 [wolfentry\\_host\\_platform\\_interface](#) \* **wolfentry\_get\_hpi** (struct wolfentry\_context \*wolfentry)

Return a pointer to the [wolfentry\\_host\\_platform\\_interface](#) associated with the supplied wolfentry\_context, mainly for passing to [wolfentry\\_alloc\\_thread\\_context\(\)](#), [wolfentry\\_free\\_thread\\_context\(\)](#), [wolfentry\\_lock\\_init\(\)](#), and [wolfentry\\_lock\\_alloc\(\)](#).

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_cleanup\_push** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, [wolfentry\\_cleanup\\_callback\\_t](#) handler, void \*arg)

Register handler to be called at shutdown with arg arg.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_cleanup\_pop** (WOLFSENTRY\_CONTEXT\_ARGS\_IN, int execute\_p)

Remove the most recently registered and unpoped handler from the cleanup stack, and if execute\_p is nonzero, call it with the arg with which it was registered.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_cleanup\_all** (WOLFSENTRY\_CONTEXT\_ARGS\_IN)

Iteratively call [wolfentry\\_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\\_t](#) 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\\_CONTEXT\\_ARGS\\_IN](#), [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\\_ARGS\\_IN](#), 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\_IN, 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\_ARGS\_IN, 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\_IN, [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\_IN, 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\_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\\_ARGS\\_IN](#), 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\\_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, 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 [wolfentry\\_route\\_insert\(\)](#) that accepts the new route as [wolfentry\\_route\\_exports](#), takes an explicit `route_table`, and returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_and\\_check\\_out](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label\_len, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

Variant of [wolfentry\\_route\\_insert\(\)](#) that returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_insert\\_by\\_exports\\_and\\_check\\_out](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_route\\_exports](#) \*route↔\_exports, struct [wolfentry\\_route](#) \*\*route, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

Variant of [wolfentry\\_route\\_insert\(\)](#) that accepts the new route as [wolfentry\\_route\\_exports](#) and returns the inserted route, which the caller must eventually drop using [wolfentry\\_route\\_drop\\_reference\(\)](#) or [wolfentry\\_object\\_release\(\)](#)

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete\\_from\\_table](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*route\_table, void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*event\_label, int event\_label↔\_len, [wolfentry\\_action\\_res\\_t](#) \*action\_results, int \*n\_deleted)

Variant of [wolfentry\\_route\\_delete\(\)](#) that takes an explicit `route_table`.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, const struct [wolfentry\\_sockaddr](#) \*remote, const struct [wolfentry\\_sockaddr](#) \*local, [wolfentry\\_route\\_flags\\_t](#) flags, const char \*trigger\_label, int trigger\_label\_len, [wolfentry\\_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 [wolfentry\\_route\\_delete\\_by\\_id\(\)](#). The supplied trigger event, if any, is passed to action handlers, and has no bearing on route matching.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_delete\\_by\\_id](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), void \*caller\_arg, [wolfentry\\_ent\\_id\\_t](#) id, const char \*trigger\_label, int trigger\_label\_len, [wolfentry\\_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.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_get\\_main\\_table](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), struct [wolfentry\\_route\\_table](#) \*\*table)

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

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_start](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*\*cursor)

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

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_seek\\_to\\_head](#) (const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor)

Reset the cursor to the beginning of a table.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_seek\\_to\\_tail](#) (const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_cursor](#) \*cursor)

Move the cursor to the end of a table.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_current](#) (const struct [wolfentry](#)↔\_route\_table \*table, struct [wolfentry\\_cursor](#) \*cursor, struct [wolfentry\\_route](#) \*\*route)

Get the current position for the table cursor.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_prev](#) (const struct [wolfentry](#)↔\_route\_table \*table, struct [wolfentry\\_cursor](#) \*cursor, struct [wolfentry\\_route](#) \*\*route)

Get the previous position for the table cursor.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_next](#) (const struct [wolfentry](#)↔\_route\_table \*table, struct [wolfentry\\_cursor](#) \*cursor, struct [wolfentry\\_route](#) \*\*route)

Get the next position for the table cursor.

- WOLFENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_iterate\\_end](#) ([WOLFENTRY\\_CONTEXT\\_ARGS\\_IN](#), const struct [wolfentry\\_route\\_table](#) \*table, struct [wolfentry\\_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\_IN, 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\_IN, 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\_IN, 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\_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\(\)](#).
- 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\_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](#).
- 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 `wolfentry` and pass it through the filters. The `action_results` are cleared on entry, and can be checked to see what actions `wolfentry` took, and what actions the caller should take (most saliently, `WOLFENTRY_ACTION_RES_ACCEPT` or `WOLFENTRY_ACTION_RES_REJECT`). `action_results` can be filtered with constructs like `WOLFENTRY_MASKIN_BITS(action_results, WOLFENTRY_ACTION_RES_REJECT)`

- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_with_table_with_inited_result (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *route_table, const struct wolfentry_sockaddr *remote, const struct wolfentry_sockaddr *local, wolfentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfentry_ent_id_t *id, wolfentry_route_flags_t *inexact_matches, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_event_dispatch()` that accepts an explicit `route_table`, and doesn't clear `action_results` on entry.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_with_inited_result (WOLFENTRY_CONTEXT_ARGS_IN, const struct wolfentry_sockaddr *remote, const struct wolfentry_sockaddr *local, wolfentry_route_flags_t flags, const char *event_label, int event_label_len, void *caller_arg, wolfentry_ent_id_t *id, wolfentry_route_flags_t *inexact_matches, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_event_dispatch()` that doesn't clear `action_results` on entry.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_by_id (WOLFENTRY_CONTEXT_ARGS_IN, wolfentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_event_dispatch()` that preselects the matched route by ID, mainly for use by application code that tracks ID/session relationships.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_by_id_with_inited_result (WOLFENTRY_CONTEXT_ARGS_IN, wolfentry_ent_id_t id, const char *event_label, int event_label_len, void *caller_arg, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_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.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_by_route (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_event_dispatch()` that preselects the matched route by ID, mainly for use by application code that tracks route/session relationships.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_event_dispatch_by_route_with_inited_result (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route *route, const char *event_label, int event_label_len, void *caller_arg, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_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.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_table_max_purgeable_routes_get (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_hitcount_t *max_purgeable_routes)`  
 Retrieve the current limit for ephemeral routes in `table`. Caller must have a lock on the context at entry.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_table_max_purgeable_routes_set (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_hitcount_t max_purgeable_routes)`  
 Set the limit for ephemeral routes in `table`. Caller must have a mutex on the context at entry.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_stale_purge (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_action_res_t *action_results)`  
 Purges stale (expired) routes from `table`.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_stale_purge_one (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_stale_purge()` that purges at most one stale route, to limit time spent working.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_stale_purge_one_opportunisticly (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_action_res_t *action_results)`  
 Variant of `wolfentry_route_stale_purge()` that purges at most one stale route, and only if the context lock is uncontended.
- `WOLFENTRY_API wolfentry_errcode_t wolfentry_route_flush_table (WOLFENTRY_CONTEXT_ARGS_IN, struct wolfentry_route_table *table, wolfentry_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\_ARGS\_IN, [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\_ARGS\_IN, 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\_ARGS\_IN, 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\_IN, 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\_ARGS\_IN, 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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_start](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*\*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)

Start a rendering loop to export the route table contents as a JSON document that is valid input for [wolfentry\\_config\\_json\\_feed\(\)](#) or [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_next](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)

Render a route within a loop started with [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_table\\_dump\\_json\\_end](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route\_table \*table, struct wolfentry\_cursor \*\*cursor, unsigned char \*\*json\_out, size\_t \*json\_out\_len, [wolfentry\\_format\\_flags\\_t](#) flags)

Finish a rendering loop started with [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_render\\_flags](#) ([wolfentry\\_route\\_flags\\_t](#) flags, FILE \*f)

Render route flags in human-readable form to a stream.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_render](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct wolfentry\_route \*r, FILE \*f)

Renders route information to a file pointer.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_route\\_exports\\_render](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const struct [wolfentry\\_route\\_exports](#) \*r, FILE \*f)

Renders route exports information to a file pointer.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_insert](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, [wolfentry\\_action\\_flags\\_t](#) flags, [wolfentry\\_action\\_callback\\_t](#) handler, void \*handler\_arg, [wolfentry\\_ent\\_id\\_t](#) \*id)

Insert a new action into wolfentry.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_delete](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

Delete an action from wolfentry.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_flush\\_all](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN)

Flush all actions from wolfentry.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_get\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, struct wolfentry\_action \*\*action)

Get a reference to an action.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_drop\\_reference](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, struct wolfentry\_action \*action, [wolfentry\\_action\\_res\\_t](#) \*action\_results)

Drop a reference to an action.

- WOLFSENTRY\_API const char \* [wolfentry\\_action\\_get\\_label](#) (const struct wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_get\\_flags](#) (struct wolfentry\_action \*action, [wolfentry\\_action\\_flags\\_t](#) \*flags)

Get the flags for an action.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_action\\_update\\_flags](#) (struct wolfentry\_action \*action, [wolfentry\\_action\\_flags\\_t](#) flags\_to\_set, [wolfentry\\_action\\_flags\\_t](#) flags\_to\_clear, [wolfentry\\_action\\_flags\\_t](#) \*flags\_before, [wolfentry\\_action\\_flags\\_t](#) \*flags\_after)

Update the flags for an action.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_event\\_insert](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const char \*label, int label\_len, [wolfentry\\_priority\\_t](#) priority, const struct [wolfentry\\_eventconfig](#) \*config, [wolfentry\\_event\\_flags\\_t](#) flags, [wolfentry\\_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 `wolfentry_event_action_list_start()`. Caller must have a lock on the context at entry.

- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_user_value_set_validator` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `wolfentry_kv_validator_t` validator, `wolfentry_action_res_t` \*action\_results)  
*Install a supplied `wolfentry_kv_validator_t` to validate all user values before inserting them into the value table.*
- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_user_value_get_type` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `const char` \*key, `int` key\_len, `wolfentry_kv_type_t` \*type)  
*Returns the type of the value with the designated key, using `WOLFSENTRY_KV_TYPE()`.*
- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_user_value_delete` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `const char` \*key, `int` key\_len)  
*Deletes the value with the designated key.*
- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_user_value_store_bool` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `const char` \*key, `int` key\_len, `wolfentry_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 `wolfentry_errcode_t wolfentry_user_value_get_bool` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `const char` \*key, `int` key\_len, `wolfentry_kv_type_t` \*value)  
*Gets a `WOLFSENTRY_KV_TRUE` or `WOLFSENTRY_KV_FALSE` value with the designated key.*
- WOLFSENTRY\_API `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_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 `wolfentry_errcode_t wolfentry_user_value_get_string` (WOLFSENTRY\_CONTEXT\_ARGS\_IN, `const char` \*key, `int` key\_len, `const char` \*\*value, `int` \*value\_len, `struct wolfentry_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\_IN, 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\_document\_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\_bytes() 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\_ARGS\_IN, 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\_IN, 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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_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 wolfentry.h

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

```
00001 /*
00002  * wolfentry.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
00078 /*
00079  ## Building and testing
00080
00081  Build and test libwolfentry.a on Linux:
00082
00083  `make -j test`
00084
00085  Build verbosely:
00086
00087  `make V=1 -j test`
00088
00089  Build with artifacts in an alternate location (outside or in a subdirectory of the source tree):
00090
00091  `make BUILD_TOP=./build -j test`
00092
00093  Install from an alternate build location to a non-standard destination:
00094
00095  `make BUILD_TOP=./build INSTALL_DIR=/usr INSTALL_LIBDIR=/usr/lib64 install`
00096
00097  Build libwolfentry.a and test it under various analyzers (memory and thread
00098  testing under full battery of valgrind and sanitizer tests):
00099
00100  `make -j check`
00101
00102  Build and test libwolfentry.a without support for multithreading:
00103
00104  `make -j SINGLETHREADED=1 test`
00105
00106  Other available make flags are `STATIC=1`, `STRIPPED=1`, `NO_JSON=1`, and
00107  `NO_JSON_DOM=1`, and the defaults values for `DEBUG`, `OPTIM`, and `C_WARNFLAGS`
00108  can also be usefully overridden.
00109
00110  Build with a user-supplied makefile preamble to override defaults:
00111
00112  `make -j USER_MAKE_CONF=Makefile.settings`
00113
00114  (`Makefile.settings` can contain simple settings like `OPTIM := -Os`, or
00115  elaborate makefile code including additional rules and dependency mechanisms.)
```

```

00116
00117 Build the smallest simplest possible library:
00118
00119 `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'`
00120
00121 Build and test with user settings:
00122
00123 `make -j USER_SETTINGS_FILE=user_settings.h test`
00124
00125 Build for FreeRTOS on ARM32, assuming FreeRTOS and lwIP source trees are located as shown:
00126
00127 `make -j HOST=arm-none-eabi RUNTIME=FreeRTOS-lwIP FREERTOS_TOP=../third/FreeRTOSv202212.00
LWIP_TOP=../third/lwip EXTRA_CFLAGS='-mcpu=cortex-m7'`
00128
00129
00130 ## Examples
00131
00132 In [the wolfSSL repository](https://github.com/wolfSSL/wolfssl), see code in
00133 `wolfSentry/test.h` gated on `WOLFSSL_WOLFSENTRY_HOOKS`, including
00134 `wolfSentry_store_endpoints()`, `wolfSentry_NetworkFilterCallback()`,
00135 `wolfSentry_setup()`, and `tcp_connect_with_wolfSentry()`. See also code in
00136 `examples/server/server.c` and `examples/client/client.c` gated on
00137 `WOLFSSL_WOLFSENTRY_HOOKS`. Use `configure --enable-wolfSentry` to build with
00138 wolfSentry integration, and use `--with-wolfSentry=/the/install/path` if
00139 wolfSentry is installed in a nonstandard location. The wolfSSL test
00140 client/server can be loaded with user-supplied wolfSentry JSON configurations
00141 from the command line, using `--wolfSentry-config <file>`.
00142
00143 More comprehensive examples of API usage are in the wolfSentry repo in
00144 tests/unittests.c, particularly `test_static_routes()`, `test_dynamic_rules()`,
00145 and `test_json()`.
00146
00147 Example JSON configuration files are at `tests/test-config.json` and
00148 `tests/test-config-numeric.json`. The latter differs only by the use of raw
00149 numbers rather than names for address families and protocols.
00150
00151 In the wolfSentry/examples/ directory are a set of example ports and
00152 applications, including a demo pop-up notification system integrating with the
00153 Linux D-Bus facility.
00154
00155
00156 ## Change Log
00157
00158 The latest changes and additions are noted in ChangeLog.md at the top of the repository.
00159
00160 */
00161
00162 #ifndef WOLFSENTRY_H
00163 #define WOLFSENTRY_H
00164
00165 #define WOLFSENTRY_VERSION_MAJOR 1
00166 #define WOLFSENTRY_VERSION_MINOR 5
00167 #define WOLFSENTRY_VERSION_TINY 0
00168 #define WOLFSENTRY_VERSION_ENCODE(major, minor, tiny) (((major) < 16U) | ((minor) < 8U) | (tiny))
00169 #define WOLFSENTRY_VERSION_WOLFSENTRY_VERSION_ENCODE(WOLFSENTRY_VERSION_MAJOR,
WOLFSENTRY_VERSION_MINOR, WOLFSENTRY_VERSION_TINY)
00170 #define WOLFSENTRY_VERSION_GT(major, minor, tiny) (WOLFSENTRY_VERSION >
WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00171 #define WOLFSENTRY_VERSION_GE(major, minor, tiny) (WOLFSENTRY_VERSION >=
WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00172 #define WOLFSENTRY_VERSION_EQ(major, minor, tiny) (WOLFSENTRY_VERSION ==
WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00173 #define WOLFSENTRY_VERSION_LT(major, minor, tiny) (WOLFSENTRY_VERSION <
WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00174 #define WOLFSENTRY_VERSION_LE(major, minor, tiny) (WOLFSENTRY_VERSION <=
WOLFSENTRY_VERSION_ENCODE(major, minor, tiny))
00175
00176 typedef enum {
00177     WOLFSENTRY_INIT_FLAG_NONE = 0,
00178     WOLFSENTRY_INIT_FLAG_LOCK_SHARED_ERROR_CHECKING = 1<0
00179 } wolfSentry_init_flags_t;
00180
00181 #ifndef WOLFSENTRY
00182 #define WOLFSENTRY /* activate wolfSentry codepaths in CentiJSON headers */
00183 #endif
00184
00185 #include <wolfSentry/wolfSentry_settings.h>
00186 #include <wolfSentry/wolfSentry_af.h>
00187 #include <wolfSentry/wolfSentry_errcodes.h>
00188
00189 struct wolfSentry_allocator;
00190 struct wolfSentry_context;
00191 struct wolfSentry_thread_context;
00192
00193 #ifdef WOLFSENTRY_THREADSAFE
00194
00195 typedef void *(*wolfSentry_malloc_cb_t)(void *context, struct wolfSentry_thread_context *thread,

```

```

    size_t size);
00237 typedef void (*wolfentry_free_cb_t)(void *context, struct wolfentry_thread_context *thread, void
    *ptr);
00241 typedef void (*wolfentry_realloc_cb_t)(void *context, struct wolfentry_thread_context *thread, void
    *ptr, size_t size);
00245 typedef void (*wolfentry_memalign_cb_t)(void *context, struct wolfentry_thread_context *thread,
    size_t alignment, size_t size);
00249 typedef void (*wolfentry_free_aligned_cb_t)(void *context, struct wolfentry_thread_context *thread,
    void *ptr);
00255 #else /* !WOLFENTRY_THREADSAFE */
00256
00257 typedef void (*wolfentry_malloc_cb_t)(void *context, size_t size);
00258 typedef void (*wolfentry_free_cb_t)(void *context, void *ptr);
00259 typedef void (*wolfentry_realloc_cb_t)(void *context, void *ptr, size_t size);
00260 typedef void (*wolfentry_memalign_cb_t)(void *context, size_t alignment, size_t size);
00261 typedef void (*wolfentry_free_aligned_cb_t)(void *context, void *ptr);
00262
00263 #endif /* WOLFENTRY_THREADSAFE */
00264
00266 struct wolfentry_allocator {
00267     void *context;
00269     wolfentry_malloc_cb_t malloc;
00271     wolfentry_free_cb_t free;
00273     wolfentry_realloc_cb_t realloc;
00275     wolfentry_memalign_cb_t memalign;
00279     wolfentry_free_aligned_cb_t free_aligned;
00281 };
00282
00289 typedef wolfentry_errcode_t (*wolfentry_get_time_cb_t)(void *context, wolfentry_time_t *ts);
00292 typedef wolfentry_time_t (*wolfentry_diff_time_cb_t)(wolfentry_time_t earlier, wolfentry_time_t
    later);
00294 typedef wolfentry_time_t (*wolfentry_add_time_cb_t)(wolfentry_time_t start_time, wolfentry_time_t
    time_interval);
00296 typedef wolfentry_errcode_t (*wolfentry_to_epoch_time_cb_t)(wolfentry_time_t when, time_t
    *epoch_secs, long *epoch_nsecs);
00298 typedef wolfentry_errcode_t (*wolfentry_from_epoch_time_cb_t)(time_t epoch_secs, long epoch_nsecs,
    wolfentry_time_t *when);
00300 typedef wolfentry_errcode_t (*wolfentry_interval_to_seconds_cb_t)(wolfentry_time_t howlong, time_t
    *howlong_secs, long *howlong_nsecs);
00302 typedef wolfentry_errcode_t (*wolfentry_interval_from_seconds_cb_t)(time_t howlong_secs, long
    howlong_nsecs, wolfentry_time_t *howlong);
00306 struct wolfentry_timecbs {
00307     void *context;
00309     wolfentry_get_time_cb_t get_time;
00311     wolfentry_diff_time_cb_t diff_time;
00313     wolfentry_add_time_cb_t add_time;
00315     wolfentry_to_epoch_time_cb_t to_epoch_time;
00317     wolfentry_from_epoch_time_cb_t from_epoch_time;
00319     wolfentry_interval_to_seconds_cb_t interval_to_seconds;
00321     wolfentry_interval_from_seconds_cb_t interval_from_seconds;
00323 };
00324
00327 #ifdef WOLFENTRY_THREADSAFE
00328
00333 typedef int (*sem_init_cb_t)(sem_t *sem, int pshared, unsigned int value);
00335 typedef int (*sem_post_cb_t)(sem_t *sem);
00337 typedef int (*sem_wait_cb_t)(sem_t *sem);
00339 typedef int (*sem_timedwait_cb_t)(sem_t *sem, const struct timespec *abs_timeout);
00341 typedef int (*sem_trywait_cb_t)(sem_t *sem);
00343 typedef int (*sem_destroy_cb_t)(sem_t *sem);
00347 struct wolfentry_semcbcs {
00348     sem_init_cb_t sem_init;
00350     sem_post_cb_t sem_post;
00352     sem_wait_cb_t sem_wait;
00354     sem_timedwait_cb_t sem_timedwait;
00356     sem_trywait_cb_t sem_trywait;
00358     sem_destroy_cb_t sem_destroy;
00360 };
00361
00364 #endif /* WOLFENTRY_THREADSAFE */
00365
00371 struct wolfentry_host_platform_interface {
00372     struct wolfentry_build_settings caller_build_settings; /* must be first */
00374     struct wolfentry_allocator allocator;
00376     struct wolfentry_timecbs timecbs;
00378 #ifdef WOLFENTRY_THREADSAFE
00379     struct wolfentry_semcbcs semcbcs;
00381 #endif
00382 };
00383
00384 WOLFENTRY_API struct wolfentry_build_settings wolfentry_get_build_settings(void);
00386 WOLFENTRY_API wolfentry_errcode_t wolfentry_build_settings_compatible(struct
    wolfentry_build_settings caller_build_settings);
00391 #ifdef WOLFENTRY_THREADSAFE
00392
00398 typedef enum {
00399     WOLFENTRY_THREAD_FLAG_NONE = 0,

```



```

00401     WOLFSENTRY_THREAD_FLAG_DEADLINE = 1<<0,
00403     WOLFSENTRY_THREAD_FLAG_READONLY = 1<<1
00405 } wolfsentry_thread_flags_t;
00406
00407 #define WOLFSENTRY_CONTEXT_ARGS_IN struct wolfsentry_context *wolfsentry, struct
wolfsentry_thread_context *thread
00409 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX(ctx) ctx, struct wolfsentry_thread_context *thread
00414 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX4(ctx, thr) struct wolfsentry_context *ctx, struct
wolfsentry_thread_context *thr
00416 #define WOLFSENTRY_CONTEXT_ELEMENTS struct wolfsentry_context *wolfsentry; struct
wolfsentry_thread_context *thread
00418 #define WOLFSENTRY_CONTEXT_SET_ELEMENTS(s) (s).wolfsentry = wolfsentry; (s).thread = thread
00420 #define WOLFSENTRY_CONTEXT_GET_ELEMENTS(s) (s).wolfsentry, (s).thread
00422 #define WOLFSENTRY_CONTEXT_ARGS_OUT wolfsentry, thread
00424 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX(ctx) ctx, thread
00426 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX2(x) (x)->wolfsentry, (x)->thread
00428 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX3(x, y) (x)->y, (x)->thread
00430 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(x, y) x, y
00432 #define WOLFSENTRY_CONTEXT_ARGS_NOT_USED (void)wolfsentry; (void)thread
00434 #define WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED (void)thread
00437 /* note WOLFSENTRY_THREAD_HEADER_DECLS includes final semicolon. */
00438 #define WOLFSENTRY_THREAD_HEADER_DECLS
00439     struct wolfsentry_thread_context_public thread_buffer =
00440     WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER;
00441     struct wolfsentry_thread_context *thread =
00442     (struct wolfsentry_thread_context *)&thread_buffer;
00443     wolfsentry_errcode_t _thread_context_ret;
00446 #define WOLFSENTRY_THREAD_HEADER_INIT(flags)
00447     (_thread_context_ret =
00448     wolfsentry_init_thread_context(thread, flags, NULL /* user_context */))
00451 #define WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(flags)
00452     do {
00453         _thread_context_ret =
00454         wolfsentry_init_thread_context(thread, flags, NULL /* user_context */); \
00455         if (_thread_context_ret < 0)
00456             return _thread_context_ret;
00457     } while (0)
00460 #define WOLFSENTRY_THREAD_HEADER(flags)
00461     struct wolfsentry_thread_context_public thread_buffer =
00462     WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER;
00463     struct wolfsentry_thread_context *thread =
00464     (struct wolfsentry_thread_context *)&thread_buffer;
00465     wolfsentry_errcode_t _thread_context_ret =
00466     wolfsentry_init_thread_context(thread, flags, NULL /* user_context */))
00469 #define WOLFSENTRY_THREAD_HEADER_CHECK()
00470     do {
00471         if (_thread_context_ret < 0)
00472             return _thread_context_ret;
00473     } while (0)
00476 #define WOLFSENTRY_THREAD_HEADER_CHECKED(flags)
00477     WOLFSENTRY_THREAD_HEADER(flags);
00478     WOLFSENTRY_THREAD_HEADER_CHECK()
00481 #define WOLFSENTRY_THREAD_TAILER(flags) (_thread_context_ret =
wolfsentry_destroy_thread_context(thread, flags))
00483 #define WOLFSENTRY_THREAD_TAILER_CHECKED(flags) do { WOLFSENTRY_THREAD_TAILER(flags); if
(_thread_context_ret < 0) return _thread_context_ret; } while (0)
00485 #define WOLFSENTRY_THREAD_GET_ERROR _thread_context_ret
00489 typedef enum {
00490     WOLFSENTRY_LOCK_FLAG_NONE = 0,
00492     WOLFSENTRY_LOCK_FLAG_PSHARED = 1<<0,
00494     WOLFSENTRY_LOCK_FLAG_SHARED_ERROR_CHECKING = 1<<1,
00496     WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_MUTEX = 1<<2,
00498     WOLFSENTRY_LOCK_FLAG_NONRECURSIVE_SHARED = 1<<3,
00500     WOLFSENTRY_LOCK_FLAG_GET_RESERVATION_TOO = 1<<4,
00502     WOLFSENTRY_LOCK_FLAG_TRY_RESERVATION_TOO = 1<<5,
00504     WOLFSENTRY_LOCK_FLAG_ABANDON_RESERVATION_TOO = 1<<6,
00506     WOLFSENTRY_LOCK_FLAG_AUTO_DOWNGRADE = 1<<7,
00508     WOLFSENTRY_LOCK_FLAG_RETAIN_SEMAPHORE = 1<<8
00510 } wolfsentry_lock_flags_t;
00511
00512 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);
00514 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);
00516 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_id(struct wolfsentry_thread_context *thread,
wolfsentry_thread_id_t *id);
00518 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_user_context(struct
wolfsentry_thread_context *thread, void **user_context);
00520 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_deadline(struct wolfsentry_thread_context
*thread, struct timespec *deadline);
00522 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_thread_flags(struct wolfsentry_thread_context
*thread, wolfsentry_thread_flags_t *thread_flags);
00524 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_destroy_thread_context(struct wolfsentry_thread_context
*thread_context, wolfsentry_thread_flags_t thread_flags);
00526 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);
00528 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_deadline_rel_usecs(WOLFSENTRY_CONTEXT_ARGS_IN, int
        usecs);
00530 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_deadline_abs(WOLFSENTRY_CONTEXT_ARGS_IN, time_t
        epoch_secs, long epoch_nsecs);
00532 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_clear_deadline(WOLFSENTRY_CONTEXT_ARGS_IN);
00534 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_thread_readonly(struct wolfsentry_thread_context
        *thread_context);
00536 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_set_thread_readwrite(struct wolfsentry_thread_context
        *thread_context);
00539 struct wolfsentry_rwlock;
00540
00555 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);
00556 WOLFSENTRY_API size_t wolfsentry_lock_size(void);
00571 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);
00583 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00596 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);
00609 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);
00621 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00634 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);
00647 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);
00659 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_mutex2shared(struct wolfsentry_rwlock *lock,
        struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00671 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex(struct wolfsentry_rwlock *lock,
        struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00684 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);
00697 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);
00713 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_reserve(struct wolfsentry_rwlock
        *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00725 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_redeem(struct wolfsentry_rwlock
        *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00738 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);
00751 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);
00763 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_shared2mutex_abandon(struct wolfsentry_rwlock
        *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00777 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00791 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_mutex(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00806 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_either(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00819 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_have_shared2mutex_reservation(struct
        wolfsentry_rwlock *lock, struct wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00831 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_get_flags(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t *flags);
00843 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_unlock(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00856 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_destroy(struct wolfsentry_rwlock *lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00870 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_lock_free(struct wolfsentry_rwlock **lock, struct
        wolfsentry_thread_context *thread, wolfsentry_lock_flags_t flags);
00871
00872 #else /* !WOLFSENTRY_THREADSafe */
00873
00874 #define WOLFSENTRY_CONTEXT_ARGS_IN struct wolfsentry_context *wolfsentry
00875 #define WOLFSENTRY_CONTEXT_ARGS_IN_EX(ctx) ctx
00876 #define WOLFSENTRY_CONTEXT_ELEMENTS struct wolfsentry_context *wolfsentry
00877 #define WOLFSENTRY_CONTEXT_SET_ELEMENTS(s) (s).wolfsentry = wolfsentry
00878 #define WOLFSENTRY_CONTEXT_GET_ELEMENTS(s) (s).wolfsentry
00879 #define WOLFSENTRY_CONTEXT_ARGS_OUT wolfsentry
00880 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX(ctx) ctx
00881 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX2(x) (x)->wolfsentry
00882 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX3(x, y) (x)->y
00883 #define WOLFSENTRY_CONTEXT_ARGS_OUT_EX4(x, y) x
00884 #define WOLFSENTRY_CONTEXT_ARGS_NOT_USED (void)wolfsentry
00885 #define WOLFSENTRY_CONTEXT_ARGS_THREAD_NOT_USED DO_NOTHING
00886
00887 #define WOLFSENTRY_THREAD_HEADER_DECLS

```

```

00888 #define WOLFSENTRY_THREAD_HEADER(flags) DO_NOTHING
00889 #define WOLFSENTRY_THREAD_HEADER_INIT(flags) 0
00890 #define WOLFSENTRY_THREAD_HEADER_INIT_CHECKED(flags) DO_NOTHING
00891 #define WOLFSENTRY_THREAD_HEADER_CHECKED(flags) DO_NOTHING
00892 #define WOLFSENTRY_THREAD_HEADER_CHECK() DO_NOTHING
00893 #define WOLFSENTRY_THREAD_GET_ERROR 0
00894 #define WOLFSENTRY_THREAD_TAILER(flags) 0
00895 #define WOLFSENTRY_THREAD_TAILER_CHECKED(flags) DO_NOTHING
00896
00897 #define wolfsentry_lock_init(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00898 #define wolfsentry_lock_alloc(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00899 #define wolfsentry_lock_shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00900 #define wolfsentry_lock_shared_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00901 #define wolfsentry_lock_mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00902 #define wolfsentry_lock_mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00903 #define wolfsentry_lock_mutex_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00904 #define wolfsentry_lock_mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00905 #define wolfsentry_lock_mutex2shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00906 #define wolfsentry_lock_shared2mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00907 #define wolfsentry_lock_shared2mutex_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00908 #define wolfsentry_lock_shared2mutex_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00909 #define wolfsentry_lock_shared2mutex_reserve(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00910 #define wolfsentry_lock_shared2mutex_redeem(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00911 #define wolfsentry_lock_shared2mutex_redeem_abstimed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00912 #define wolfsentry_lock_shared2mutex_redeem_timed(x, y, z, w) WOLFSENTRY_ERROR_ENCODE(OK)
00913 #define wolfsentry_lock_shared2mutex_abandon(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00914 #define wolfsentry_lock_have_shared(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00915 #define wolfsentry_lock_have_mutex(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00916 #define wolfsentry_lock_have_either(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00917 #define wolfsentry_lock_have_shared2mutex_reservation(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00918 #define wolfsentry_lock_unlock(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00919 #define wolfsentry_lock_destroy(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00920 #define wolfsentry_lock_free(x, y, z) WOLFSENTRY_ERROR_ENCODE(OK)
00921
00922 #endif /* WOLFSENTRY_THREADSafe */
00923
00931 typedef enum {
00932     WOLFSENTRY_OBJECT_TYPE_UNINITED = 0,
00933     WOLFSENTRY_OBJECT_TYPE_TABLE,
00934     WOLFSENTRY_OBJECT_TYPE_ACTION,
00935     WOLFSENTRY_OBJECT_TYPE_EVENT,
00936     WOLFSENTRY_OBJECT_TYPE_ROUTE,
00937     WOLFSENTRY_OBJECT_TYPE_KV,
00938     WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNUMBER,
00939     WOLFSENTRY_OBJECT_TYPE_ADDR_FAMILY_BYNAME
00940 } wolfsentry_object_type_t;
00941
00957 typedef enum {
00958     WOLFSENTRY_ACTION_FLAG_NONE = 0U,
00959     WOLFSENTRY_ACTION_FLAG_DISABLED = 1U << 0U
00960 } wolfsentry_action_flags_t;
00961
00965 typedef enum {
00966     WOLFSENTRY_ACTION_TYPE_NONE = 0,
00967     WOLFSENTRY_ACTION_TYPE_POST = 1,
00968     WOLFSENTRY_ACTION_TYPE_INSERT = 2,
00969     WOLFSENTRY_ACTION_TYPE_MATCH = 3,
00970     WOLFSENTRY_ACTION_TYPE_UPDATE = 4,
00971     WOLFSENTRY_ACTION_TYPE_DELETE = 5,
00972     WOLFSENTRY_ACTION_TYPE_DECISION = 6
00973 } wolfsentry_action_type_t;
00974
00981
00982 #define WOLFSENTRY_ACTION_RES_USER_SHIFT 24U
00983
00986 typedef enum {
00987     WOLFSENTRY_ACTION_RES_NONE = 0U,
00988     WOLFSENTRY_ACTION_RES_ACCEPT = 1U << 0U,
00989     WOLFSENTRY_ACTION_RES_REJECT = 1U << 1U,
00990     WOLFSENTRY_ACTION_RES_CONNECT = 1U << 2U,
00991     WOLFSENTRY_ACTION_RES_DISCONNECT = 1U << 3U,
00992     WOLFSENTRY_ACTION_RES_DEROGATORY = 1U << 4U,
00993     WOLFSENTRY_ACTION_RES_COMMENDABLE = 1U << 5U,
00994     WOLFSENTRY_ACTION_RES_EXCLUDE_REJECT_ROUTES = WOLFSENTRY_ACTION_RES_DEROGATORY |
00995     WOLFSENTRY_ACTION_RES_COMMENDABLE, /* internal use -- overload used by wolfsentry_route_lookup_0() */
01004     WOLFSENTRY_ACTION_RES_STOP = 1U << 6U,
01006     WOLFSENTRY_ACTION_RES_DEALLOCATED = 1U << 7U,
01008     WOLFSENTRY_ACTION_RES_INSERTED = 1U << 8U,
01010     WOLFSENTRY_ACTION_RES_ERROR = 1U << 9U,
01012     WOLFSENTRY_ACTION_RES_FALLTHROUGH = 1U << 10U,
01014     WOLFSENTRY_ACTION_RES_UPDATE = 1U << 11U,
01016     WOLFSENTRY_ACTION_RES_PORT_RESET = 1U << 12U,
01018     WOLFSENTRY_ACTION_RES_SENDING = 1U << 13U,
01020     WOLFSENTRY_ACTION_RES_RECEIVED = 1U << 14U,
01022     WOLFSENTRY_ACTION_RES_BINDING = 1U << 15U,
01024     WOLFSENTRY_ACTION_RES_LISTENING = 1U << 16U,
01026     WOLFSENTRY_ACTION_RES_STOPPED_LISTENING = 1U << 17U,
01028     WOLFSENTRY_ACTION_RES_CONNECTING_OUT = 1U << 18U,
01030     WOLFSENTRY_ACTION_RES_CLOSED = 1U << 19U,

```

```

01032     WOLFSENTRY_ACTION_RES_UNREACHABLE = 1U << 20U,
01034     WOLFSENTRY_ACTION_RES_SOCK_ERROR   = 1U << 21U,
01037     WOLFSENTRY_ACTION_RES_RESERVED22  = 1U << 22U,
01038     WOLFSENTRY_ACTION_RES_RESERVED23  = 1U << 23U,
01040     WOLFSENTRY_ACTION_RES_USER_BASE   = 1U << WOLFSENTRY_ACTION_RES_USER_SHIFT
01042 } wolfsentry_action_res_t;
01043
01046 struct wolfsentry_table_header;
01047 struct wolfsentry_table_ent_header;
01048 struct wolfsentry_route;
01049 struct wolfsentry_route_table;
01050 struct wolfsentry_event;
01051 struct wolfsentry_event_table;
01052 struct wolfsentry_action;
01053 struct wolfsentry_action_table;
01054 struct wolfsentry_action_list;
01055 struct wolfsentry_action_list_ent;
01056 struct wolfsentry_cursor;
01057
01079 typedef wolfsentry_errcode_t (*wolfsentry_action_callback_t) (
01080     WOLFSENTRY_CONTEXT_ARGS_IN,
01081     const struct wolfsentry_action *action,
01082     void *handler_arg,
01083     void *caller_arg,
01084     const struct wolfsentry_event *trigger_event,
01085     wolfsentry_action_type_t action_type,
01086     const struct wolfsentry_route *trigger_route,
01087     struct wolfsentry_route_table *route_table,
01088     struct wolfsentry_route *rule_route,
01089     wolfsentry_action_res_t *action_results);
01090
01097 #define WOLFSENTRY_ROUTE_DEFAULT_POLICY_MASK (WOLFSENTRY_ACTION_RES_ACCEPT |
WOLFSENTRY_ACTION_RES_REJECT | WOLFSENTRY_ACTION_RES_STOP | WOLFSENTRY_ACTION_RES_ERROR)
01101 typedef enum {
01102     WOLFSENTRY_ROUTE_FLAG_NONE = 0U,
01104     /* note the wildcard bits need to be at the start, in order of field
01105      * comparison by wolfsentry_route_key_cmp_1(), due to math in
01106      * wolfsentry_route_lookup_0().
01107      */
01108     WOLFSENTRY_ROUTE_FLAG_SA_FAMILY_WILDCARD = 1U<<0U,
01110     WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_ADDR_WILDCARD = 1U<<1U,
01112     WOLFSENTRY_ROUTE_FLAG_SA_PROTO_WILDCARD = 1U<<2U,
01114     WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_PORT_WILDCARD = 1U<<3U,
01116     WOLFSENTRY_ROUTE_FLAG_SA_LOCAL_ADDR_WILDCARD = 1U<<4U,
01118     WOLFSENTRY_ROUTE_FLAG_SA_REMOTE_PORT_WILDCARD = 1U<<5U,
01120     WOLFSENTRY_ROUTE_FLAG_REMOTE_INTERFACE_WILDCARD = 1U<<6U,
01122     WOLFSENTRY_ROUTE_FLAG_LOCAL_INTERFACE_WILDCARD = 1U<<7U,
01124     WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD = 1U<<8U,
01126     WOLFSENTRY_ROUTE_FLAG_TCPLIKE_PORT_NUMBERS = 1U<<9U,
01128     WOLFSENTRY_ROUTE_FLAG_DIRECTION_IN = 1U<<10U,
01130     WOLFSENTRY_ROUTE_FLAG_DIRECTION_OUT = 1U<<11U,
01133     /* immutable above here. */
01134
01135     /* internal use from here... */
01136     WOLFSENTRY_ROUTE_FLAG_IN_TABLE = 1U<<12U,
01138     WOLFSENTRY_ROUTE_FLAG_PENDING_DELETE = 1U<<13U,
01140     WOLFSENTRY_ROUTE_FLAG_INSERT_ACTIONS_CALLED = 1U<<14U,
01142     WOLFSENTRY_ROUTE_FLAG_DELETE_ACTIONS_CALLED = 1U<<15U,
01145     /* ...to here. */
01146
01147     /* mutable below here. */
01148
01149     WOLFSENTRY_ROUTE_FLAG_PENALTYBOXED = 1U<<16U,
01151     WOLFSENTRY_ROUTE_FLAG_GREENLISTED = 1U<<17U,
01153     WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_HITS = 1U<<18U,
01155     WOLFSENTRY_ROUTE_FLAG_DONT_COUNT_CURRENT_CONNECTIONS = 1U<<19U,
01157     WOLFSENTRY_ROUTE_FLAG_PORT_RESET = 1U<<20U
01159 } wolfsentry_route_flags_t;
01160
01161 /* note, _PARENT_EVENT_WILDCARD is excluded because it isn't an intrinsic attribute of network/bus
traffic. */
01162 #define WOLFSENTRY_ROUTE_WILDCARD_FLAGS
((wolfsentry_route_flags_t)WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD - 1U)
01165 #define WOLFSENTRY_ROUTE_IMMUTABLE_FLAGS ((wolfsentry_route_flags_t)WOLFSENTRY_ROUTE_FLAG_IN_TABLE -
1U)
01169 #define WOLFSENTRY_ROUTE_FLAG_TRIGGER_WILDCARD WOLFSENTRY_ROUTE_FLAG_PARENT_EVENT_WILDCARD /* xxx
backward compatibility */
01173 struct wolfsentry_route_endpoint {
01174     wolfsentry_port_t sa_port;
01176     wolfsentry_addr_bits_t addr_len;
01178     byte extra_port_count;
01180     byte interface;
01182 };
01183
01185 struct wolfsentry_route_metadata_exports {
01186     wolfsentry_time_t insert_time;
01188     wolfsentry_time_t last_hit_time;

```

```

01190     wolfsentry_time_t last_penaltybox_time;
01192     wolfsentry_time_t purge_after;
01194     uint16_t connection_count;
01196     uint16_t derogatory_count;
01198     uint16_t commendable_count;
01200     wolfsentry_hitcount_t hit_count;
01202 };
01203
01205 struct wolfsentry_route_exports {
01206     const char *parent_event_label;
01208     int parent_event_label_len;
01210     wolfsentry_route_flags_t flags;
01212     wolfsentry_addr_family_t sa_family;
01214     wolfsentry_proto_t sa_proto;
01216     struct wolfsentry_route_endpoint remote;
01218     struct wolfsentry_route_endpoint local;
01220     const byte *remote_address;
01222     const byte *local_address;
01224     const wolfsentry_port_t *remote_extra_ports;
01226     const wolfsentry_port_t *local_extra_ports;
01228     struct wolfsentry_route_metadata_exports meta;
01230     void *private_data;
01232     size_t private_data_size;
01234 };
01235
01237 struct wolfsentry_sockaddr {
01238     wolfsentry_addr_family_t sa_family;
01240     wolfsentry_proto_t sa_proto;
01242     wolfsentry_port_t sa_port;
01244     wolfsentry_addr_bits_t addr_len;
01246     byte interface;
01248     attr_align_to(4) byte addr[WOLFSENTRY_FLEXIBLE_ARRAY_SIZE];
01250 };
01251
01252 #define WOLFSENTRY_SOCKADDR(n) struct {
01253     wolfsentry_addr_family_t sa_family;
01254     wolfsentry_proto_t sa_proto;
01255     wolfsentry_port_t sa_port;
01256     wolfsentry_addr_bits_t addr_len;
01257     byte interface;
01258     attr_align_to(4) byte addr[WOLFSENTRY_BITS_TO_BYTES(n)];
01259 }
01263 typedef enum {
01264     WOLFSENTRY_FORMAT_FLAG_NONE = 0,
01266     WOLFSENTRY_FORMAT_FLAG_ALWAYS_NUMERIC = 1U << 0U
01268 } wolfsentry_format_flags_t;
01269
01277 typedef enum {
01278     WOLFSENTRY_EVENT_FLAG_NONE = 0,
01280     WOLFSENTRY_EVENT_FLAG_IS_PARENT_EVENT = 1U << 0U,
01282     WOLFSENTRY_EVENT_FLAG_IS_SUBEVENT = 1U << 1U
01284 } wolfsentry_event_flags_t;
01285
01287 typedef enum {
01288     WOLFSENTRY_EVENTCONFIG_FLAG_NONE = 0U,
01290     WOLFSENTRY_EVENTCONFIG_FLAG_DEROGATORY_THRESHOLD_IGNORE_COMMENDABLE = 1U << 0U,
01292     WOLFSENTRY_EVENTCONFIG_FLAG_COMMENDABLE_CLEARS_DEROGATORY = 1U << 1U,
01294     WOLFSENTRY_EVENTCONFIG_FLAG_INHIBIT_ACTIONS = 1U << 2U
01296 } wolfsentry_eventconfig_flags_t;
01297
01299 struct wolfsentry_eventconfig {
01300     size_t route_private_data_size;
01302     size_t route_private_data_alignment;
01304     uint32_t max_connection_count;
01306     wolfsentry_hitcount_t derogatory_threshold_for_penaltybox;
01308     wolfsentry_time_t penaltybox_duration;
01310     wolfsentry_time_t route_idle_time_for_purge;
01312     wolfsentry_eventconfig_flags_t flags;
01314     wolfsentry_route_flags_t route_flags_to_add_on_insert;
01316     wolfsentry_route_flags_t route_flags_to_clear_on_insert;
01318     wolfsentry_action_res_t action_res_filter_bits_set;
01320     wolfsentry_action_res_t action_res_filter_bits_unset;
01322     wolfsentry_action_res_t action_res_bits_to_add;
01324     wolfsentry_action_res_t action_res_bits_to_clear;
01326 };
01327
01334 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_now_plus_delta(struct wolfsentry_context
    *wolfsentry, wolfsentry_time_t td, wolfsentry_time_t *res);
01337 #ifndef WOLFSENTRY_THREADSAFE
01338 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_to_timespec(struct wolfsentry_context *wolfsentry,
    wolfsentry_time_t t, struct timespec *ts);
01340 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_time_now_plus_delta_timespec(struct wolfsentry_context
    *wolfsentry, wolfsentry_time_t td, struct timespec *ts);
01342 #endif
01343
01344 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_get_time(struct wolfsentry_context *wolfsentry,
    wolfsentry_time_t *time_p);

```

```

01346 WOLFSENTRY_API wolfentry_time_t wolfentry_diff_time(struct wolfentry_context *wolfentry,
wolfentry_time_t later, wolfentry_time_t earlier);
01348 WOLFSENTRY_API wolfentry_time_t wolfentry_add_time(struct wolfentry_context *wolfentry,
wolfentry_time_t start_time, wolfentry_time_t time_interval);
01350 WOLFSENTRY_API wolfentry_errcode_t wolfentry_to_epoch_time(struct wolfentry_context *wolfentry,
wolfentry_time_t when, time_t *epoch_secs, long *epoch_nsecs);
01352 WOLFSENTRY_API wolfentry_errcode_t wolfentry_from_epoch_time(struct wolfentry_context *wolfentry,
time_t epoch_secs, long epoch_nsecs, wolfentry_time_t *when);
01354 WOLFSENTRY_API wolfentry_errcode_t wolfentry_interval_to_seconds(struct wolfentry_context
*wolfentry, wolfentry_time_t howlong, time_t *howlong_secs, long *howlong_nsecs);
01356 WOLFSENTRY_API wolfentry_errcode_t wolfentry_interval_from_seconds(struct wolfentry_context
*wolfentry, time_t howlong_secs, long howlong_nsecs, wolfentry_time_t *howlong);
01359 WOLFSENTRY_API struct wolfentry_timecbs *wolfentry_get_timecbs(struct wolfentry_context
*wolfentry);
01367 typedef wolfentry_errcode_t (*wolfentry_make_id_cb_t)(void *context, wolfentry_ent_id_t *id);
01373 WOLFSENTRY_API void *wolfentry_malloc(WOLFSENTRY_CONTEXT_ARGS_IN, size_t size);
01375 WOLFSENTRY_API void wolfentry_free(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr);
01377 WOLFSENTRY_API void *wolfentry_realloc(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr, size_t size);
01379 WOLFSENTRY_API void *wolfentry_memalign(WOLFSENTRY_CONTEXT_ARGS_IN, size_t alignment, size_t size);
01381 WOLFSENTRY_API void wolfentry_free_aligned(WOLFSENTRY_CONTEXT_ARGS_IN, void *ptr);
01383 #if (defined(WOLFSENTRY_MALLOC_BUILTINS) && defined(WOLFSENTRY_MALLOC_DEBUG)) ||
defined(WOLFSENTRY_FOR_DOXYGEN)
01384 WOLFSENTRY_API int _wolfentry_get_n_mallocs(void);
01386 #endif
01387
01388 WOLFSENTRY_API struct wolfentry_allocator *wolfentry_get_allocator(struct wolfentry_context
*wolfentry);
01393 #if defined(WOLFSENTRY_PROTOCOL_NAMES) || !defined(WOLFSENTRY_NO_JSON)
01397 WOLFSENTRY_API const char *wolfentry_action_res_assoc_by_flag(wolfentry_action_res_t res, unsigned
int bit);
01399 WOLFSENTRY_API wolfentry_errcode_t wolfentry_action_res_assoc_by_name(const char *bit_name, size_t
bit_name_len, wolfentry_action_res_t *res);
01402 #endif
01403
01408 WOLFSENTRY_API struct wolfentry_host_platform_interface *wolfentry_get_hpi(struct wolfentry_context
*wolfentry);
01411 typedef void (*wolfentry_cleanup_callback_t)(
01412     WOLFSENTRY_CONTEXT_ARGS_IN,
01413     void *cleanup_arg);
01416 WOLFSENTRY_API wolfentry_errcode_t wolfentry_cleanup_push(
01417     WOLFSENTRY_CONTEXT_ARGS_IN,
01418     wolfentry_cleanup_callback_t handler,
01419     void *arg);
01422 WOLFSENTRY_API wolfentry_errcode_t wolfentry_cleanup_pop(
01423     WOLFSENTRY_CONTEXT_ARGS_IN,
01424     int execute_p);
01427 WOLFSENTRY_API wolfentry_errcode_t wolfentry_cleanup_all(
01428     WOLFSENTRY_CONTEXT_ARGS_IN);
01437 /* must return _BUFFER_TOO_SMALL and set *addr_internal_bits to an
01438 * accurate value when supplied with a NULL output buf ptr.
01439 * whenever _BUFFER_TOO_SMALL is returned, *addr_*_bits must be set to an
01440 * accurate value.
01441 */
01442 typedef wolfentry_errcode_t (*wolfentry_addr_family_parser_t)(
01443     WOLFSENTRY_CONTEXT_ARGS_IN,
01444     const char *addr_text,
01445     int addr_text_len,
01446     byte *addr_internal,
01447     wolfentry_addr_bits_t *addr_internal_bits);
01450 typedef wolfentry_errcode_t (*wolfentry_addr_family_formatter_t)(
01451     WOLFSENTRY_CONTEXT_ARGS_IN,
01452     const byte *addr_internal,
01453     unsigned int addr_internal_bits,
01454     char *addr_text,
01455     int *addr_text_len);
01458 WOLFSENTRY_API wolfentry_errcode_t wolfentry_addr_family_handler_install(
01459     WOLFSENTRY_CONTEXT_ARGS_IN,
01460     wolfentry_addr_family_t family_bynumber,
01461     const char *family_byname, /* if defined(WOLFSENTRY_PROTOCOL_NAMES), must not be NULL, else
ignored. */
01462     int family_byname_len,
01463     wolfentry_addr_family_parser_t parser,
01464     wolfentry_addr_family_formatter_t formatter,
01465     int max_addr_bits);
01468 WOLFSENTRY_API wolfentry_errcode_t wolfentry_addr_family_get_parser(
01469     WOLFSENTRY_CONTEXT_ARGS_IN,
01470     wolfentry_addr_family_t family,
01471     wolfentry_addr_family_parser_t *parser);
01474 WOLFSENTRY_API wolfentry_errcode_t wolfentry_addr_family_get_formatter(
01475     WOLFSENTRY_CONTEXT_ARGS_IN,
01476     wolfentry_addr_family_t family,
01477     wolfentry_addr_family_formatter_t *formatter);
01480 WOLFSENTRY_API wolfentry_errcode_t wolfentry_addr_family_handler_remove_bynumber(
01481     WOLFSENTRY_CONTEXT_ARGS_IN,
01482     wolfentry_addr_family_t family_bynumber,
01483     wolfentry_action_res_t *action_results);
01486 struct wolfentry_addr_family_bynumber;

```



```

01487
01488 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_drop_reference(
01489     WOLFSENTRY_CONTEXT_ARGS_IN,
01490     struct wolfsentry_addr_family_bynumber *family_bynumber,
01491     wolfsentry_action_res_t *action_results);
01494 #ifndef WOLFSENTRY_PROTOCOL_NAMES
01495
01496 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_handler_remove_byname(
01497     WOLFSENTRY_CONTEXT_ARGS_IN,
01498     const char *family_byname,
01499     int family_byname_len,
01500     wolfsentry_action_res_t *action_results);
01503 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_pton(
01504     WOLFSENTRY_CONTEXT_ARGS_IN,
01505     const char *family_name,
01506     int family_name_len,
01507     wolfsentry_addr_family_t *family_number);
01510 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_ntop(
01511     WOLFSENTRY_CONTEXT_ARGS_IN,
01512     wolfsentry_addr_family_t family,
01513     struct wolfsentry_addr_family_bynumber **addr_family,
01514     const char **family_name);
01517 #endif /* WOLFSENTRY_PROTOCOL_NAMES */
01518
01519 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_addr_family_max_addr_bits(
01520     WOLFSENTRY_CONTEXT_ARGS_IN,
01521     wolfsentry_addr_family_t family,
01522     wolfsentry_addr_bits_t *bits);
01540 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_init(
01541     struct wolfsentry_context *wolfsentry,
01542     struct wolfsentry_eventconfig *config);
01550 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_eventconfig_check(
01551     const struct wolfsentry_eventconfig *config);
01552
01558 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init_ex(
01559     struct wolfsentry_build_settings caller_build_settings,
01560     WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi),
01561     const struct wolfsentry_eventconfig *config,
01562     struct wolfsentry_context **wolfsentry,
01563     wolfsentry_init_flags_t flags);
01578 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_init(
01579     struct wolfsentry_build_settings caller_build_settings,
01580     WOLFSENTRY_CONTEXT_ARGS_IN_EX(const struct wolfsentry_host_platform_interface *hpi),
01581     const struct wolfsentry_eventconfig *config,
01582     struct wolfsentry_context **wolfsentry);
01590 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_get(
01591     WOLFSENTRY_CONTEXT_ARGS_IN,
01592     struct wolfsentry_eventconfig *config);
01602 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_defaultconfig_update(
01603     WOLFSENTRY_CONTEXT_ARGS_IN,
01604     const struct wolfsentry_eventconfig *config);
01612 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_flush(WOLFSENTRY_CONTEXT_ARGS_IN);
01622 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_free(WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct
    wolfsentry_context **wolfsentry));
01631 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_shutdown(WOLFSENTRY_CONTEXT_ARGS_IN_EX(struct
    wolfsentry_context **wolfsentry));
01639 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_inhibit_actions(WOLFSENTRY_CONTEXT_ARGS_IN);
01647 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_enable_actions(WOLFSENTRY_CONTEXT_ARGS_IN);
01648
01650 typedef enum {
01651     WOLFSENTRY_CLONE_FLAG_NONE = 0U,
01653     WOLFSENTRY_CLONE_FLAG_AS_AT_CREATION = 1U << 0U,
01655     WOLFSENTRY_CLONE_FLAG_NO_ROUTES = 2U << 0U
01657 } wolfsentry_clone_flags_t;
01668 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_clone(WOLFSENTRY_CONTEXT_ARGS_IN, struct
    wolfsentry_context **clone, wolfsentry_clone_flags_t flags);
01678 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_exchange(WOLFSENTRY_CONTEXT_ARGS_IN, struct
    wolfsentry_context *wolfsentry2);
01679
01686 #ifndef WOLFSENTRY_THREADSafe
01687
01688 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex(
01689     WOLFSENTRY_CONTEXT_ARGS_IN);
01691 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_abstimed(
01692     WOLFSENTRY_CONTEXT_ARGS_IN,
01693     const struct timespec *abs_timeout);
01695 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_abstimed_ex(
01696     WOLFSENTRY_CONTEXT_ARGS_IN,
01697     const struct timespec *abs_timeout,
01698     wolfsentry_lock_flags_t flags);
01700 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_timed(
01701     WOLFSENTRY_CONTEXT_ARGS_IN,
01702     wolfsentry_time_t max_wait);
01704 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_mutex_timed_ex(
01705     WOLFSENTRY_CONTEXT_ARGS_IN,
01706     wolfsentry_time_t max_wait,
01707     wolfsentry_lock_flags_t flags);

```

```

01709 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared(
01710     WOLFSENTRY_CONTEXT_ARGS_IN);
01712 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_abstimed(
01713     WOLFSENTRY_CONTEXT_ARGS_IN,
01714     const struct timespec *abs_timeout);
01716 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_with_reservation_abstimed(
01717     WOLFSENTRY_CONTEXT_ARGS_IN,
01718     const struct timespec *abs_timeout);
01720 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_timed(
01721     WOLFSENTRY_CONTEXT_ARGS_IN,
01722     wolfsentry_time_t max_wait);
01724 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_lock_shared_with_reservation_timed(
01725     WOLFSENTRY_CONTEXT_ARGS_IN,
01726     wolfsentry_time_t max_wait);
01728 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_unlock(
01729     WOLFSENTRY_CONTEXT_ARGS_IN);
01731 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_context_unlock_and_abandon_reservation(
01732     WOLFSENTRY_CONTEXT_ARGS_IN);
01735 #else /* !WOLFSENTRY_THREADSafe */
01736
01737 #define wolfsentry_context_lock_mutex(x) WOLFSENTRY_ERROR_ENCODE(OK)
01738 #define wolfsentry_context_lock_mutex_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01739 #define wolfsentry_context_lock_mutex_timed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01740 #define wolfsentry_context_lock_shared(x) WOLFSENTRY_ERROR_ENCODE(OK)
01741 #define wolfsentry_context_lock_shared_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01742 #define wolfsentry_context_lock_shared_with_reservation_abstimed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01743 #define wolfsentry_context_lock_shared_timed(x, y) WOLFSENTRY_ERROR_ENCODE(OK)
01744 #define wolfsentry_context_unlock(x) WOLFSENTRY_ERROR_ENCODE(OK)
01745
01746 #endif /* WOLFSENTRY_THREADSafe */
01747
01750 #define WOLFSENTRY_LENGTH_NULL_TERMINATED (-1)
01764 WOLFSENTRY_API wolfsentry_object_type_t wolfsentry_get_object_type(const void *object);
01765
01773 WOLFSENTRY_API wolfsentry_ent_id_t wolfsentry_get_object_id(const void *object);
01774
01775 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_table_ent_get_by_id(
01776     WOLFSENTRY_CONTEXT_ARGS_IN,
01777     wolfsentry_ent_id_t id,
01778     struct wolfsentry_table_ent_header **ent);
01781 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_object_checkout(WOLFSENTRY_CONTEXT_ARGS_IN, void
01782     *object);
01784 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_object_release(WOLFSENTRY_CONTEXT_ARGS_IN, void
01785     *object, wolfsentry_action_res_t *action_results);
01794 WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_inserts(struct wolfsentry_table_header
01795     *table);
01796
01803 WOLFSENTRY_API wolfsentry_hitcount_t wolfsentry_table_n_deletes(struct wolfsentry_table_header
01804     *table);
01805
01811 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_check_flags_sensical(
01812     wolfsentry_route_flags_t flags);
01815 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_into_table(
01816     WOLFSENTRY_CONTEXT_ARGS_IN,
01817     struct wolfsentry_route_table *route_table,
01818     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01819     const struct wolfsentry_sockaddr *remote,
01820     const struct wolfsentry_sockaddr *local,
01821     wolfsentry_route_flags_t flags,
01822     const char *event_label,
01823     int event_label_len,
01824     wolfsentry_ent_id_t *id,
01825     wolfsentry_action_res_t *action_results);
01828 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_into_table(
01829     WOLFSENTRY_CONTEXT_ARGS_IN,
01830     struct wolfsentry_route_table *route_table,
01831     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01832     const struct wolfsentry_route_exports *route_exports,
01833     wolfsentry_ent_id_t *id,
01834     wolfsentry_action_res_t *action_results);
01853 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert(
01854     WOLFSENTRY_CONTEXT_ARGS_IN,
01855     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01856     const struct wolfsentry_sockaddr *remote,
01857     const struct wolfsentry_sockaddr *local,
01858     wolfsentry_route_flags_t flags,
01859     const char *event_label,
01860     int event_label_len,
01861     wolfsentry_ent_id_t *id,
01862     wolfsentry_action_res_t *action_results);
01863
01864 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports(
01865     WOLFSENTRY_CONTEXT_ARGS_IN,
01866     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01867     const struct wolfsentry_route_exports *route_exports,
01868     wolfsentry_ent_id_t *id,
01869     wolfsentry_action_res_t *action_results);

```



```

01872 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_into_table_and_check_out(
01873     WOLFSENTRY_CONTEXT_ARGS_IN,
01874     struct wolfsentry_route_table *route_table,
01875     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01876     const struct wolfsentry_sockaddr *remote,
01877     const struct wolfsentry_sockaddr *local,
01878     wolfsentry_route_flags_t flags,
01879     const char *event_label,
01880     int event_label_len,
01881     struct wolfsentry_route **route,
01882     wolfsentry_action_res_t *action_results);
01885 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_into_table_and_check_out(
01886     WOLFSENTRY_CONTEXT_ARGS_IN,
01887     struct wolfsentry_route_table *route_table,
01888     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01889     const struct wolfsentry_route_exports *route_exports,
01890     struct wolfsentry_route **route,
01891     wolfsentry_action_res_t *action_results);
01894 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_and_check_out(
01895     WOLFSENTRY_CONTEXT_ARGS_IN,
01896     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01897     const struct wolfsentry_sockaddr *remote,
01898     const struct wolfsentry_sockaddr *local,
01899     wolfsentry_route_flags_t flags,
01900     const char *event_label,
01901     int event_label_len,
01902     struct wolfsentry_route **route,
01903     wolfsentry_action_res_t *action_results);
01906 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_insert_by_exports_and_check_out(
01907     WOLFSENTRY_CONTEXT_ARGS_IN,
01908     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01909     const struct wolfsentry_route_exports *route_exports,
01910     struct wolfsentry_route **route,
01911     wolfsentry_action_res_t *action_results);
01914 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_from_table(
01915     WOLFSENTRY_CONTEXT_ARGS_IN,
01916     struct wolfsentry_route_table *route_table,
01917     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01918     const struct wolfsentry_sockaddr *remote,
01919     const struct wolfsentry_sockaddr *local,
01920     wolfsentry_route_flags_t flags,
01921     const char *event_label,
01922     int event_label_len,
01923     wolfsentry_action_res_t *action_results,
01924     int *n_deleted);
01943 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete(
01944     WOLFSENTRY_CONTEXT_ARGS_IN,
01945     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01946     const struct wolfsentry_sockaddr *remote,
01947     const struct wolfsentry_sockaddr *local,
01948     wolfsentry_route_flags_t flags,
01949     const char *trigger_label,
01950     int trigger_label_len,
01951     wolfsentry_action_res_t *action_results,
01952     int *n_deleted);
01953
01967 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_delete_by_id(
01968     WOLFSENTRY_CONTEXT_ARGS_IN,
01969     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
01970     wolfsentry_ent_id_t id,
01971     const char *trigger_label,
01972     int trigger_label_len,
01973     wolfsentry_action_res_t *action_results);
01974
01986 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_main_table(
01987     WOLFSENTRY_CONTEXT_ARGS_IN,
01988     struct wolfsentry_route_table **table);
01989
02002 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_start(
02003     WOLFSENTRY_CONTEXT_ARGS_IN,
02004     const struct wolfsentry_route_table *table,
02005     struct wolfsentry_cursor **cursor);
02006
02015 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_head(
02016     const struct wolfsentry_route_table *table,
02017     struct wolfsentry_cursor *cursor);
02018
02027 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_seek_to_tail(
02028     const struct wolfsentry_route_table *table,
02029     struct wolfsentry_cursor *cursor);
02030
02040 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_current(
02041     const struct wolfsentry_route_table *table,
02042     struct wolfsentry_cursor *cursor,
02043     struct wolfsentry_route **route);
02044
02054 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_prev(

```

```

02055     const struct wolfsentry_route_table *table,
02056     struct wolfsentry_cursor *cursor,
02057     struct wolfsentry_route **route);
02058
02068 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_next(
02069     const struct wolfsentry_route_table *table,
02070     struct wolfsentry_cursor *cursor,
02071     struct wolfsentry_route **route);
02072
02085 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_iterate_end(
02086     WOLFSENTRY_CONTEXT_ARGS_IN,
02087     const struct wolfsentry_route_table *table,
02088     struct wolfsentry_cursor **cursor);
02089
02100 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_set(
02101     WOLFSENTRY_CONTEXT_ARGS_IN,
02102     struct wolfsentry_route_table *table,
02103     wolfsentry_action_res_t default_policy);
02104
02105 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_default_policy_set(
02106     WOLFSENTRY_CONTEXT_ARGS_IN,
02107     wolfsentry_action_res_t default_policy);
02123 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_default_policy_get(
02124     WOLFSENTRY_CONTEXT_ARGS_IN,
02125     struct wolfsentry_route_table *table,
02126     wolfsentry_action_res_t *default_policy);
02127
02128 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_default_policy_get(
02129     WOLFSENTRY_CONTEXT_ARGS_IN,
02130     wolfsentry_action_res_t *default_policy);
02150 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_reference(
02151     WOLFSENTRY_CONTEXT_ARGS_IN,
02152     const struct wolfsentry_route_table *table,
02153     const struct wolfsentry_sockaddr *remote,
02154     const struct wolfsentry_sockaddr *local,
02155     wolfsentry_route_flags_t flags,
02156     const char *event_label,
02157     int event_label_len,
02158     int exact_p,
02159     wolfsentry_route_flags_t *inexact_matches,
02160     struct wolfsentry_route **route);
02161
02172 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_drop_reference(
02173     WOLFSENTRY_CONTEXT_ARGS_IN,
02174     struct wolfsentry_route *route,
02175     wolfsentry_action_res_t *action_results);
02176
02177 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_clear_default_event(
02178     WOLFSENTRY_CONTEXT_ARGS_IN,
02179     struct wolfsentry_route_table *table);
02182 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_set_default_event(
02183     WOLFSENTRY_CONTEXT_ARGS_IN,
02184     struct wolfsentry_route_table *table,
02185     const char *event_label,
02186     int event_label_len);
02189 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_get_default_event(
02190     WOLFSENTRY_CONTEXT_ARGS_IN,
02191     struct wolfsentry_route_table *table,
02192     char *event_label,
02193     int *event_label_len);
02204 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_fallthrough_route_get(
02205     WOLFSENTRY_CONTEXT_ARGS_IN,
02206     struct wolfsentry_route_table *route_table,
02207     const struct wolfsentry_route **fallthrough_route);
02208
02217 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_addrs(
02218     const struct wolfsentry_route *route,
02219     wolfsentry_addr_family_t *af,
02220     wolfsentry_addr_bits_t *local_addr_len,
02221     const byte **local_addr,
02222     wolfsentry_addr_bits_t *remote_addr_len,
02223     const byte **remote_addr);
02224
02240 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_export(
02241     WOLFSENTRY_CONTEXT_ARGS_IN,
02242     const struct wolfsentry_route *route,
02243     struct wolfsentry_route_exports *route_exports);
02244
02245 /* returned wolfsentry_event remains valid only as long as the wolfsentry lock
02246 * is held (shared or exclusive), unless the route was obtained via
02247 * wolfsentry_route_get_reference(), in which case it's valid until
02248 * wolfsentry_route_drop_reference()..
02249 */
02259 WOLFSENTRY_API const struct wolfsentry_event *wolfsentry_route_parent_event(const struct
    wolfsentry_route *route);
02260
02261 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_table(

```

```

02262     WOLFSENTRY_CONTEXT_ARGS_IN,
02263     struct wolfsentry_route_table *route_table,
02264     const struct wolfsentry_sockaddr *remote,
02265     const struct wolfsentry_sockaddr *local,
02266     wolfsentry_route_flags_t flags,
02267     const char *event_label,
02268     int event_label_len,
02269     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02270     wolfsentry_ent_id_t *id,
02271     wolfsentry_route_flags_t *inexact_matches,
02272     wolfsentry_action_res_t *action_results);
02292 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch(
02293     WOLFSENTRY_CONTEXT_ARGS_IN,
02294     const struct wolfsentry_sockaddr *remote,
02295     const struct wolfsentry_sockaddr *local,
02296     wolfsentry_route_flags_t flags,
02297     const char *event_label,
02298     int event_label_len,
02299     void *caller_arg, /* passed to action callback(s). */
02300     wolfsentry_ent_id_t *id,
02301     wolfsentry_route_flags_t *inexact_matches,
02302     wolfsentry_action_res_t *action_results);
02303
02304 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_table_with_initiated_result(
02305     WOLFSENTRY_CONTEXT_ARGS_IN,
02306     struct wolfsentry_route_table *route_table,
02307     const struct wolfsentry_sockaddr *remote,
02308     const struct wolfsentry_sockaddr *local,
02309     wolfsentry_route_flags_t flags,
02310     const char *event_label,
02311     int event_label_len,
02312     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02313     wolfsentry_ent_id_t *id,
02314     wolfsentry_route_flags_t *inexact_matches,
02315     wolfsentry_action_res_t *action_results);
02318 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_with_initiated_result(
02319     WOLFSENTRY_CONTEXT_ARGS_IN,
02320     const struct wolfsentry_sockaddr *remote,
02321     const struct wolfsentry_sockaddr *local,
02322     wolfsentry_route_flags_t flags,
02323     const char *event_label,
02324     int event_label_len,
02325     void *caller_arg, /* passed to action callback(s). */
02326     wolfsentry_ent_id_t *id,
02327     wolfsentry_route_flags_t *inexact_matches,
02328     wolfsentry_action_res_t *action_results);
02331 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_id(
02332     WOLFSENTRY_CONTEXT_ARGS_IN,
02333     wolfsentry_ent_id_t id,
02334     const char *event_label,
02335     int event_label_len,
02336     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02337     wolfsentry_action_res_t *action_results
02338 );
02341 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_id_with_initiated_result(
02342     WOLFSENTRY_CONTEXT_ARGS_IN,
02343     wolfsentry_ent_id_t id,
02344     const char *event_label,
02345     int event_label_len,
02346     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02347     wolfsentry_action_res_t *action_results
02348 );
02351 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_route(
02352     WOLFSENTRY_CONTEXT_ARGS_IN,
02353     struct wolfsentry_route *route,
02354     const char *event_label,
02355     int event_label_len,
02356     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02357     wolfsentry_action_res_t *action_results
02358 );
02361 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_event_dispatch_by_route_with_initiated_result(
02362     WOLFSENTRY_CONTEXT_ARGS_IN,
02363     struct wolfsentry_route *route,
02364     const char *event_label,
02365     int event_label_len,
02366     void *caller_arg, /* passed to action callback(s) as the caller_arg. */
02367     wolfsentry_action_res_t *action_results
02368 );
02371 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_routes_get(
02372     WOLFSENTRY_CONTEXT_ARGS_IN,
02373     struct wolfsentry_route_table *table,
02374     wolfsentry_hitcount_t *max_purgeable_routes);
02377 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_max_purgeable_routes_set(
02378     WOLFSENTRY_CONTEXT_ARGS_IN,
02379     struct wolfsentry_route_table *table,
02380     wolfsentry_hitcount_t max_purgeable_routes);
02393 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge(

```

```

02394     WOLFSENTRY_CONTEXT_ARGS_IN,
02395     struct wolfsentry_route_table *table,
02396     wolfsentry_action_res_t *action_results);
02397
02398 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge_one(
02399     WOLFSENTRY_CONTEXT_ARGS_IN,
02400     struct wolfsentry_route_table *table,
02401     wolfsentry_action_res_t *action_results);
02402 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_stale_purge_one_opportunisticly(
02403     WOLFSENTRY_CONTEXT_ARGS_IN,
02404     struct wolfsentry_route_table *table,
02405     wolfsentry_action_res_t *action_results);
02406 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flush_table(
02407     WOLFSENTRY_CONTEXT_ARGS_IN,
02408     struct wolfsentry_route_table *table,
02409     wolfsentry_action_res_t *action_results);
02410
02411 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_clear_insert_action_status(
02412     WOLFSENTRY_CONTEXT_ARGS_IN,
02413     wolfsentry_action_res_t *action_results);
02414
02415 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_bulk_insert_actions(
02416     WOLFSENTRY_CONTEXT_ARGS_IN,
02417     wolfsentry_action_res_t *action_results);
02418
02419 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_private_data(
02420     WOLFSENTRY_CONTEXT_ARGS_IN,
02421     struct wolfsentry_route *route,
02422     void **private_data,
02423     size_t *private_data_size);
02424
02425 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_flags(
02426     const struct wolfsentry_route *route,
02427     wolfsentry_route_flags_t *flags);
02428
02429 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_get_metadata(
02430     const struct wolfsentry_route *route,
02431     struct wolfsentry_route_metadata_exports *metadata);
02432
02433 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_metadata_exports(
02434     struct wolfsentry_route_exports *route_exports);
02435 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_update_flags(
02436     WOLFSENTRY_CONTEXT_ARGS_IN,
02437     struct wolfsentry_route *route,
02438     wolfsentry_route_flags_t flags_to_set,
02439     wolfsentry_route_flags_t flags_to_clear,
02440     wolfsentry_route_flags_t *flags_before,
02441     wolfsentry_route_flags_t *flags_after,
02442     wolfsentry_action_res_t *action_results);
02443
02444 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_increment_derogatory_count(
02445     WOLFSENTRY_CONTEXT_ARGS_IN,
02446     struct wolfsentry_route *route,
02447     int count_to_add,
02448     int *new_derogatory_count_ptr);
02449 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_increment_commendable_count(
02450     WOLFSENTRY_CONTEXT_ARGS_IN,
02451     struct wolfsentry_route *route,
02452     int count_to_add,
02453     int *new_commendable_count);
02454 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_derogatory_count(
02455     WOLFSENTRY_CONTEXT_ARGS_IN,
02456     struct wolfsentry_route *route,
02457     int *old_derogatory_count_ptr);
02458 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_reset_commendable_count(
02459     WOLFSENTRY_CONTEXT_ARGS_IN,
02460     struct wolfsentry_route *route,
02461     int *old_commendable_count_ptr);
02462 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_set_wildcard(
02463     struct wolfsentry_route *route,
02464     wolfsentry_route_flags_t wildcards_to_set);
02465
02466 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_format_address(
02467     WOLFSENTRY_CONTEXT_ARGS_IN,
02468     wolfsentry_addr_family_t sa_family,
02469     const byte *addr,
02470     unsigned int addr_bits,
02471     char *buf,
02472     int *buflen);
02473
02474 #if defined(WOLFSENTRY_PROTOCOL_NAMES) || defined(WOLFSENTRY_JSON_DUMP_UTILS) ||
02475     !defined(WOLFSENTRY_NO_JSON)
02476
02477 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flag_assoc_by_flag(
02478     wolfsentry_route_flags_t flag,
02479     const char **name);
02480 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_flag_assoc_by_name(
02481     const char *name,

```

```

02573     int len,
02574     wolfsentry_route_flags_t *flag);
02577 #endif /* WOLFSENTRY_PROTOCOL_NAMES || WOLFSENTRY_JSON_DUMP_UTILS || !WOLFSENTRY_NO_JSON */
02578
02579 #if !defined(WOLFSENTRY_NO_JSON) || defined(WOLFSENTRY_JSON_DUMP_UTILS)
02580
02581 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_format_json(
02582     WOLFSENTRY_CONTEXT_ARGS_IN,
02583     const struct wolfsentry_route *r,
02584     unsigned char **json_out,
02585     size_t *json_out_len,
02586     wolfsentry_format_flags_t flags);
02589 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_start(
02590     WOLFSENTRY_CONTEXT_ARGS_IN,
02591     const struct wolfsentry_route_table *table,
02592     struct wolfsentry_cursor **cursor,
02593     unsigned char **json_out,
02594     size_t *json_out_len,
02595     wolfsentry_format_flags_t flags);
02598 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_next(
02599     WOLFSENTRY_CONTEXT_ARGS_IN,
02600     const struct wolfsentry_route_table *table,
02601     struct wolfsentry_cursor *cursor,
02602     unsigned char **json_out,
02603     size_t *json_out_len,
02604     wolfsentry_format_flags_t flags);
02607 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_table_dump_json_end(
02608     WOLFSENTRY_CONTEXT_ARGS_IN,
02609     const struct wolfsentry_route_table *table,
02610     struct wolfsentry_cursor **cursor,
02611     unsigned char **json_out,
02612     size_t *json_out_len,
02613     wolfsentry_format_flags_t flags);
02616 #endif /* !WOLFSENTRY_NO_JSON || WOLFSENTRY_JSON_DUMP_UTILS */
02617
02618 #ifndef WOLFSENTRY_NO_STDIO
02619 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render_flags(wolfsentry_route_flags_t flags, FILE
02620     *f);
02632 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_render(WOLFSENTRY_CONTEXT_ARGS_IN, const struct
02633     wolfsentry_route *r, FILE *f);
02643 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_route_exports_render(WOLFSENTRY_CONTEXT_ARGS_IN, const
02644     struct wolfsentry_route_exports *r, FILE *f);
02645 #endif
02646
02666 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_insert(
02667     WOLFSENTRY_CONTEXT_ARGS_IN,
02668     const char *label,
02669     int label_len,
02670     wolfsentry_action_flags_t flags,
02671     wolfsentry_action_callback_t handler,
02672     void *handler_arg,
02673     wolfsentry_ent_id_t *id);
02674
02686 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_delete(
02687     WOLFSENTRY_CONTEXT_ARGS_IN,
02688     const char *label,
02689     int label_len,
02690     wolfsentry_action_res_t *action_results);
02691
02699 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_flush_all(WOLFSENTRY_CONTEXT_ARGS_IN);
02700
02712 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_reference(
02713     WOLFSENTRY_CONTEXT_ARGS_IN,
02714     const char *label,
02715     int label_len,
02716     struct wolfsentry_action **action);
02717
02728 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_drop_reference(
02729     WOLFSENTRY_CONTEXT_ARGS_IN,
02730     struct wolfsentry_action *action,
02731     wolfsentry_action_res_t *action_results);
02732
02740 WOLFSENTRY_API const char *wolfsentry_action_get_label(const struct wolfsentry_action *action);
02741
02750 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_get_flags(
02751     struct wolfsentry_action *action,
02752     wolfsentry_action_flags_t *flags);
02753
02765 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_action_update_flags(
02766     struct wolfsentry_action *action,
02767     wolfsentry_action_flags_t flags_to_set,
02768     wolfsentry_action_flags_t flags_to_clear,
02769     wolfsentry_action_flags_t *flags_before,
02770     wolfsentry_action_flags_t *flags_after);
02771
02792 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_insert(
02793     WOLFSENTRY_CONTEXT_ARGS_IN,

```

```
02794     const char *label,
02795     int label_len,
02796     wolfentry_priority_t priority,
02797     const struct wolfentry_eventconfig *config,
02798     wolfentry_event_flags_t flags,
02799     wolfentry_ent_id_t *id);
02800
02810 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_delete(
02811     WOLFENTRY_CONTEXT_ARGS_IN,
02812     const char *label,
02813     int label_len,
02814     wolfentry_action_res_t *action_results);
02815
02823 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_flush_all(WOLFENTRY_CONTEXT_ARGS_IN);
02824
02832 WOLFENTRY_API const char *wolfentry_event_get_label(const struct wolfentry_event *event);
02833
02841 WOLFENTRY_API wolfentry_event_flags_t wolfentry_event_get_flags(const struct wolfentry_event
*event);
02842
02854 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_get_config(
02855     WOLFENTRY_CONTEXT_ARGS_IN,
02856     const char *label,
02857     int label_len,
02858     struct wolfentry_eventconfig *config);
02859
02871 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_update_config(
02872     WOLFENTRY_CONTEXT_ARGS_IN,
02873     const char *label,
02874     int label_len,
02875     const struct wolfentry_eventconfig *config);
02876
02888 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_get_reference(
02889     WOLFENTRY_CONTEXT_ARGS_IN,
02890     const char *label,
02891     int label_len,
02892     struct wolfentry_event **event);
02893
02904 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_drop_reference(
02905     WOLFENTRY_CONTEXT_ARGS_IN,
02906     struct wolfentry_event *event,
02907     wolfentry_action_res_t *action_results);
02908
02922 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_prepend(
02923     WOLFENTRY_CONTEXT_ARGS_IN,
02924     const char *event_label,
02925     int event_label_len,
02926     wolfentry_action_type_t which_action_list,
02927     const char *action_label,
02928     int action_label_len);
02929
02943 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_append(
02944     WOLFENTRY_CONTEXT_ARGS_IN,
02945     const char *event_label,
02946     int event_label_len,
02947     wolfentry_action_type_t which_action_list,
02948     const char *action_label,
02949     int action_label_len);
02950
02966 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_insert_after(
02967     WOLFENTRY_CONTEXT_ARGS_IN,
02968     const char *event_label,
02969     int event_label_len,
02970     wolfentry_action_type_t which_action_list,
02971     const char *action_label,
02972     int action_label_len,
02973     const char *point_action_label,
02974     int point_action_label_len);
02975
02989 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_delete(
02990     WOLFENTRY_CONTEXT_ARGS_IN,
02991     const char *event_label,
02992     int event_label_len,
02993     wolfentry_action_type_t which_action_list,
02994     const char *action_label,
02995     int action_label_len);
02996
03009 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_set_aux_event(
03010     WOLFENTRY_CONTEXT_ARGS_IN,
03011     const char *event_label,
03012     int event_label_len,
03013     const char *aux_event_label,
03014     int aux_event_label_len);
03015
03016 WOLFENTRY_API const struct wolfentry_event *wolfentry_event_get_aux_event(
03017     const struct wolfentry_event *event);
03034 WOLFENTRY_API wolfentry_errcode_t wolfentry_event_action_list_start(
```

```

03035     WOLFSENTRY_CONTEXT_ARGS_IN,
03036     const char *event_label,
03037     int event_label_len,
03038     wolfsentry_action_type_t which_action_list,
03039     struct wolfsentry_action_list_ent **cursor);
03040
03054 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_next(
03055     WOLFSENTRY_CONTEXT_ARGS_IN,
03056     struct wolfsentry_action_list_ent **cursor,
03057     const char **action_label,
03058     int *action_label_len);
03059
03071 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_event_action_list_done(
03072     WOLFSENTRY_CONTEXT_ARGS_IN,
03073     struct wolfsentry_action_list_ent **cursor);
03074
03077 #ifndef WOLFSENTRY_HAVE_JSON_DOM
03078 #include <wolfsentry/centijson_dom.h>
03079 #endif
03080
03086 typedef enum {
03087     WOLFSENTRY_KV_NONE = 0,
03088     WOLFSENTRY_KV_NULL,
03089     WOLFSENTRY_KV_TRUE,
03090     WOLFSENTRY_KV_FALSE,
03091     WOLFSENTRY_KV_UINT,
03092     WOLFSENTRY_KV_SINT,
03093     WOLFSENTRY_KV_FLOAT,
03094     WOLFSENTRY_KV_STRING,
03095     WOLFSENTRY_KV_BYTES,
03096     WOLFSENTRY_KV_JSON,
03097     WOLFSENTRY_KV_FLAG_READONLY = 1<<30
03098 } wolfsentry_kv_type_t;
03099
03100 #define WOLFSENTRY_KV_FLAG_MASK WOLFSENTRY_KV_FLAG_READONLY
03104 struct wolfsentry_kv_pair {
03105     int key_len;
03107     wolfsentry_kv_type_t v_type;
03109     union {
03110         uint64_t v_uint;
03112         int64_t v_sint;
03114         double v_float;
03116         size_t string_len;
03118         size_t bytes_len;
03120 #ifdef WOLFSENTRY_HAVE_JSON_DOM
03121         JSON_VALUE v_json; /* 16 bytes */
03123 #endif
03124     } a;
03125     byte b[WOLFSENTRY_FLEXIBLE_ARRAY_SIZE];
03130 };
03131
03132 #define WOLFSENTRY_KV_KEY_LEN(kv) ((kv)->key_len)
03134 #define WOLFSENTRY_KV_KEY(kv) ((char *)((kv)->b))
03136 #define WOLFSENTRY_KV_TYPE(kv) ((uint32_t)(kv)->v_type & ~(uint32_t)WOLFSENTRY_KV_FLAG_MASK)
03138 #define WOLFSENTRY_KV_V_UINT(kv) ((kv)->a.v_uint)
03140 #define WOLFSENTRY_KV_V_SINT(kv) ((kv)->a.v_sint)
03142 #define WOLFSENTRY_KV_V_FLOAT(kv) ((kv)->a.v_float)
03144 #define WOLFSENTRY_KV_V_STRING_LEN(kv) ((kv)->a.string_len)
03146 #define WOLFSENTRY_KV_V_STRING(kv) ((char *)((kv)->b + (kv)->key_len + 1))
03148 #define WOLFSENTRY_KV_V_BYTES_LEN(kv) ((kv)->a.bytes_len)
03150 #define WOLFSENTRY_KV_V_BYTES(kv) ((kv)->b + (kv)->key_len + 1)
03152 #ifdef WOLFSENTRY_HAVE_JSON_DOM
03153 #define WOLFSENTRY_KV_V_JSON(kv) (&(kv)->a.v_json)
03155 #endif
03156
03157 typedef wolfsentry_errcode_t (*wolfsentry_kv_validator_t)(
03158     WOLFSENTRY_CONTEXT_ARGS_IN,
03159     struct wolfsentry_kv_pair *kv);
03162 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_set_validator(
03163     WOLFSENTRY_CONTEXT_ARGS_IN,
03164     wolfsentry_kv_validator_t validator,
03165     wolfsentry_action_res_t *action_results);
03168 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_set_mutability(
03169     WOLFSENTRY_CONTEXT_ARGS_IN,
03170     const char *key,
03171     int key_len,
03172     int mutable);
03175 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_mutability(
03176     WOLFSENTRY_CONTEXT_ARGS_IN,
03177     const char *key,
03178     int key_len,
03179     int *mutable);
03182 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_type(
03183     WOLFSENTRY_CONTEXT_ARGS_IN,
03184     const char *key,
03185     int key_len,
03186     wolfsentry_kv_type_t *type);

```



```
03189 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_delete(  
03190     WOLFSENTRY_CONTEXT_ARGS_IN,  
03191     const char *key,  
03192     int key_len);  
03195 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_null(  
03196     WOLFSENTRY_CONTEXT_ARGS_IN,  
03197     const char *key,  
03198     int key_len,  
03199     int overwrite_p);  
03202 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bool(  
03203     WOLFSENTRY_CONTEXT_ARGS_IN,  
03204     const char *key,  
03205     int key_len,  
03206     wolfsentry_kv_type_t value,  
03207     int overwrite_p);  
03210 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bool(  
03211     WOLFSENTRY_CONTEXT_ARGS_IN,  
03212     const char *key,  
03213     int key_len,  
03214     wolfsentry_kv_type_t *value);  
03217 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_uint(  
03218     WOLFSENTRY_CONTEXT_ARGS_IN,  
03219     const char *key,  
03220     int key_len,  
03221     uint64_t value,  
03222     int overwrite_p);  
03225 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_uint(  
03226     WOLFSENTRY_CONTEXT_ARGS_IN,  
03227     const char *key,  
03228     int key_len,  
03229     uint64_t *value);  
03232 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_sint(  
03233     WOLFSENTRY_CONTEXT_ARGS_IN,  
03234     const char *key,  
03235     int key_len,  
03236     int64_t value,  
03237     int overwrite_p);  
03240 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_sint(  
03241     WOLFSENTRY_CONTEXT_ARGS_IN,  
03242     const char *key,  
03243     int key_len,  
03244     int64_t *value);  
03247 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_double(  
03248     WOLFSENTRY_CONTEXT_ARGS_IN,  
03249     const char *key,  
03250     int key_len,  
03251     double value,  
03252     int overwrite_p);  
03255 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_float(  
03256     WOLFSENTRY_CONTEXT_ARGS_IN,  
03257     const char *key,  
03258     int key_len,  
03259     double *value);  
03262 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_string(  
03263     WOLFSENTRY_CONTEXT_ARGS_IN,  
03264     const char *key,  
03265     int key_len,  
03266     const char *value,  
03267     int value_len,  
03268     int overwrite_p);  
03271 struct wolfsentry_kv_pair_internal;  
03272  
03279 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_string(  
03280     WOLFSENTRY_CONTEXT_ARGS_IN,  
03281     const char *key,  
03282     int key_len,  
03283     const char **value,  
03284     int *value_len,  
03285     struct wolfsentry_kv_pair_internal **user_value_record);  
03286  
03287 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bytes(  
03288     WOLFSENTRY_CONTEXT_ARGS_IN,  
03289     const char *key,  
03290     int key_len,  
03291     const byte *value,  
03292     int value_len,  
03293     int overwrite_p);  
03296 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_bytes_base64(  
03297     WOLFSENTRY_CONTEXT_ARGS_IN,  
03298     const char *key,  
03299     int key_len,  
03300     const char *value,  
03301     int value_len,  
03302     int overwrite_p);  
03311 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_bytes(  
03312     WOLFSENTRY_CONTEXT_ARGS_IN,  
03313     const char *key,
```



```

03314     int key_len,
03315     const byte **value,
03316     int *value_len,
03317     struct wolfsentry_kv_pair_internal **user_value_record);
03318
03319 #ifndef WOLFSENTRY_HAVE_JSON_DOM
03320 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_store_json(
03321     WOLFSENTRY_CONTEXT_ARGS_IN,
03322     const char *key,
03323     int key_len,
03324     JSON_VALUE *value,
03325     int overwrite_p);
03326 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_get_json(
03327     WOLFSENTRY_CONTEXT_ARGS_IN,
03328     const char *key,
03329     int key_len,
03330     JSON_VALUE **value,
03331     struct wolfsentry_kv_pair_internal **user_value_record);
03332 #endif /* WOLFSENTRY_HAVE_JSON_DOM */
03333
03334 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_value_release_record(
03335     WOLFSENTRY_CONTEXT_ARGS_IN,
03336     struct wolfsentry_kv_pair_internal **user_value_record);
03337 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_pair_export(
03338     WOLFSENTRY_CONTEXT_ARGS_IN,
03339     struct wolfsentry_kv_pair_internal *kv,
03340     const struct wolfsentry_kv_pair **kv_exports);
03341 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_type_to_string(
03342     wolfsentry_kv_type_t type,
03343     const char **out);
03344 #ifndef WOLFSENTRY_NO_STDIO
03345 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_kv_render_value(
03346     WOLFSENTRY_CONTEXT_ARGS_IN,
03347     const struct wolfsentry_kv_pair *kv,
03348     char *out,
03349     int *out_len);
03350 #endif
03351
03352 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_start(
03353     WOLFSENTRY_CONTEXT_ARGS_IN,
03354     struct wolfsentry_cursor **cursor);
03355 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_seek_to_head(
03356     WOLFSENTRY_CONTEXT_ARGS_IN,
03357     struct wolfsentry_cursor *cursor);
03358 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_seek_to_tail(
03359     WOLFSENTRY_CONTEXT_ARGS_IN,
03360     struct wolfsentry_cursor *cursor);
03361 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_current(
03362     WOLFSENTRY_CONTEXT_ARGS_IN,
03363     struct wolfsentry_cursor *cursor,
03364     struct wolfsentry_kv_pair_internal **kv);
03365 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_prev(
03366     WOLFSENTRY_CONTEXT_ARGS_IN,
03367     struct wolfsentry_cursor *cursor,
03368     struct wolfsentry_kv_pair_internal **kv);
03369 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_next(
03370     WOLFSENTRY_CONTEXT_ARGS_IN,
03371     struct wolfsentry_cursor *cursor,
03372     struct wolfsentry_kv_pair_internal **kv);
03373 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_values_iterate_end(
03374     WOLFSENTRY_CONTEXT_ARGS_IN,
03375     struct wolfsentry_cursor **cursor);
03376 #define WOLFSENTRY_BASE64_DECODED_BUFSPC(buf, len) \
03377     (((len)+3)/4)*3 - ((len) > 1 ? \
03378         ((buf)[(len)-1] == '=' ? \
03379             0) \
03379         - ((len) > 2 ? ((buf)[(len)-2] == '=' ? 0) : 0)) \
03380
03381 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_base64_decode(
03382     const char *src,
03383     size_t src_len,
03384     byte *dest,
03385     size_t *dest_spc,
03386     int ignore_junk_p);
03387 #ifdef WOLFSENTRY_LWIP
03388 #include "wolfsentry/wolfsentry_lwip.h"
03389 #endif
03390
03391 /* conditionally include wolfsentry_util.h last -- none of the above rely on it.
03392 */
03393 #ifndef WOLFSENTRY_NO_UTIL_H
03394 #include <wolfsentry/wolfsentry_util.h>
03395 #endif
03396 #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*
- #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\_LINK** ([WOLFSENTRY\\_AF\\_BSD\\_OFFSET](#) + 18)
- Link layer interface.*
- #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 [wolfentry.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_LINK       (WOLFSENTRY_AF_BSD_OFFSET + 18)
00101 #define WOLFSENTRY_AF_COIP       (WOLFSENTRY_AF_BSD_OFFSET + 20)
00102 #define WOLFSENTRY_AF_CNT        (WOLFSENTRY_AF_BSD_OFFSET + 21)
00103 #define WOLFSENTRY_AF_SIP        (WOLFSENTRY_AF_BSD_OFFSET + 24)
00104 #define WOLFSENTRY_AF_SLOW       (WOLFSENTRY_AF_BSD_OFFSET + 33)
00105 #define WOLFSENTRY_AF_SCLUSTER   (WOLFSENTRY_AF_BSD_OFFSET + 34)
00106 #define WOLFSENTRY_AF_ARP        (WOLFSENTRY_AF_BSD_OFFSET + 35)
00107 #define WOLFSENTRY_AF_IEEE80211  (WOLFSENTRY_AF_BSD_OFFSET + 37)
00108 #define WOLFSENTRY_AF_INET_SDP   (WOLFSENTRY_AF_BSD_OFFSET + 40)
00109 #define WOLFSENTRY_AF_INET6_SDP  (WOLFSENTRY_AF_BSD_OFFSET + 42)
00110 #define WOLFSENTRY_AF_HYPERV     (WOLFSENTRY_AF_BSD_OFFSET + 43)
00112 #define WOLFSENTRY_AF_USER_OFFSET 256
00113
00116 #endif /* WOLFSENTRY_AF_H */

```

## 10.8 wolfentry/wolfentry\_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 wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_errcode_t`*

- **#define WOLFSENTRY\_ERROR\_DECODE\_SOURCE\_ID(x)**

*Extract the bare source file ID from an encoded `wolfentry_errcode_t`*

- **#define WOLFSENTRY\_ERROR\_DECODE\_LINE\_NUMBER(x)**

*Extract the bare source line number from an encoded `wolfentry_errcode_t`*

- **#define WOLFSENTRY\_ERROR\_RECODE(x)**

*Take an encoded `wolfentry_errcode_t` and recode it with the current source ID and line number.*

- **#define WOLFSENTRY\_ERROR\_CODE\_IS(x, name)**

*Take an encoded `wolfentry_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 `wolfentry_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 `wolfentry_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 `wolfentry_errcode_t`.*
- **#define WOLFSENTRY\_UNLOCK\_AND\_RETURN(ret)**  
*Unlock the current context, and return the supplied `wolfentry_errcode_t`.*
- **#define WOLFSENTRY\_ERROR\_UNLOCK\_AND\_RETURN(name)**  
*Unlock the current context, and return a `wolfentry_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 `wolfentry_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 `wolfentry_context` `ctx`, and return a `wolfentry_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 `wolfentry_context` `ctx`, then take an encoded `wolfentry_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 `wolfentry_errcode_t`.*
- **#define WOLFSENTRY\_ERROR\_RERETURN\_AND\_UNLOCK(y)**  
*Calculate the `wolfentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfentry_errcode_t`.*
- **#define WOLFSENTRY\_SUCCESS\_UNLOCK\_AND\_RETURN(name)**  
*Unlock the current context, and return a `wolfentry_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 `wolfentry_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 `wolfentry_errcode_t`.*
- **#define WOLFSENTRY\_SUCCESS\_RERETURN\_AND\_UNLOCK(y)**  
*Calculate the `wolfentry_errcode_t` return value for an expression `y`, then unlock the current context, and finally, return the encoded `wolfentry_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 `wolfentry_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 `wolfentry_errcode_t` encoding the current source ID and line number, and the success code `OK`.*
- **#define WOLFSENTRY\_RERETURN\_IF\_ERROR(y)**  
*If `wolfentry_errcode_t` `y` is a failure code, return it.*
- **#define WOLFSENTRY\_UNLOCK\_AND\_RERETURN\_IF\_ERROR(y)**  
*If `wolfentry_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` or `WOLFSENTRY_NO_DIAG_MSGS` is set, `DO_NOTHING`.*
- **#define WOLFSENTRY\_WARN\_ON\_FAILURE(...)**  
*Evaluate the supplied expression, and if the resulting `wolfentry_errcode_t` encodes an error, render the expression and the decoded error using `WOLFSENTRY_PRINTF_ERR()`, but if `WOLFSENTRY_NO_STDIO` 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` 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_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_USER_BASE = 128 }

```

## Functions

- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_source\_string** ([wolfentry\\_errcode\\_t](#) e)  
Return the name of the source code file associated with `wolfentry_errcode_t` e, or "unknown user defined source", or "unknown source".
- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_error\_string** ([wolfentry\\_errcode\\_t](#) e)  
Return a description of the failure or success code associated with `wolfentry_errcode_t` e, or various "unknown" strings if not known.
- WOLFSENTRY\_API const char \* **wolfentry\_errcode\_error\_name** ([wolfentry\\_errcode\\_t](#) e)  
Return the short name of the failure or success code associated with `wolfentry_errcode_t` e, or `wolfentry_errcode_error_string(e)` if not known.
- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) **wolfentry\_user\_source\_string\_set** (enum `wolfentry_errcode_t` source\_id `wolfentry_source_id`, const char \*source\_string)  
Register a source code file so that `wolfentry_errcode_source_string()`, and therefore `WOLFSENTRY_ERROR_FMT_ARGS()` 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 [wolfentry\\_errcode\\_t](#) **wolfentry\_user\_error\_string\_set** (enum `wolfentry_errcode_t` error\_id `wolfentry_error_id`, const char \*message\_string)  
Register an error (negative) or success (positive) code, and corresponding message, so that `wolfentry_errcode_error_string()` 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 [wolfentry.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) ({ wolfentry_errcode_t _xret = (x);
WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(_xret); })
00100 #define WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(x) ({ wolfentry_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(wolfentry_errcode_t x) {
00111     return WOLFSENTRY_ERROR_DECODE_ERROR_CODE_1(x);
00112 }
00113 static inline int WOLFSENTRY_ERROR_DECODE_SOURCE_ID(wolfentry_errcode_t x) {
00114     return WOLFSENTRY_ERROR_DECODE_SOURCE_ID_1(x);
00115 }
00116 static inline int WOLFSENTRY_ERROR_DECODE_LINE_NUMBER(wolfentry_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 #ifndef 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),
wolfentry_errcode_error_string(x), WOLFSENTRY_ERROR_DECODE_SOURCE_ID(x),
wolfentry_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 #ifndef 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)
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 *_wolfentry_call_depth(void);
00170             #define _INDENT_FMT "%s"
00171             #define _INDENT_ARGS _wolfentry_call_depth()
00172             #else
00173             WOLFSENTRY_API unsigned int _wolfentry_call_depth(void);
00174             #define _INDENT_FMT "%*s"
00175             #define _INDENT_ARGS _wolfentry_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, wolfentry_errcode_error_name(x)); return
WOLFSENTRY_ERROR_ENCODE_1(x); } while (0)
00178             #define WOLFSENTRY_ERROR_RETURN_RECODED(x) do { wolfentry_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),
wolfentry_errcode_error_name(_xret)); return
WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); } while (0)
00179             #define WOLFSENTRY_ERROR_RERETURN(x) do { wolfentry_errcode_t _xret = (x); const char *_fn =
strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR(_INDENT_FMT
"%s L%d %s(): rerecord %d (%s)\n", _INDENT_ARGS, _fn, _LINE_, _FUNCTION_,
WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret), wolfentry_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,
wolfentry_errcode_error_name(x)); return WOLFSENTRY_ERROR_ENCODE_1(x); } while (0)
00184 #define WOLFSENTRY_ERROR_RETURN_RECODED(x) do { wolfentry_errcode_t _xret = (x); const char
*_fn = strrchr(__FILE__, '/'); if (_fn) { ++_fn; } else { _fn = __FILE__; } WOLFSENTRY_PRINTF_ERR("%s L%d:
L%d: return-recoded %d (%s)\n", _fn, __LINE__, WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret),
wolfentry_errcode_error_name(_xret)); return
WOLFSENTRY_ERROR_ENCODE_0(WOLFSENTRY_ERROR_DECODE_ERROR_CODE(_xret)); } while (0)
00185 #define WOLFSENTRY_ERROR_RERETURN(x) do { wolfentry_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),
wolfentry_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 { wolfentry_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 { wolfentry_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     wolfentry_errcode_t _lock_ret;
00219     if ((_lock_ret = wolfentry_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(wolfentry)
00228 #define WOLFSENTRY_UNLOCK_AND_UNRESERVE_FOR_RETURN_EX(ctx) do { \
00229     wolfentry_errcode_t _lock_ret;
00230     if ((_lock_ret = wolfentry_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(wolfentry)
00239 #define WOLFSENTRY_MUTEX_EX(ctx) wolfentry_context_lock_mutex_abstimed(ctx, thread, NULL)
00242 #define WOLFSENTRY_MUTEX_OR_RETURN() do { \
00243     wolfentry_errcode_t _lock_ret;
00244     if ((_lock_ret = WOLFSENTRY_MUTEX_EX(wolfentry)) < 0) { \
00245         WOLFSENTRY_ERROR_RERETURN(_lock_ret);
00246     } while (0)
00249 #define WOLFSENTRY_SHARED_EX(ctx) wolfentry_context_lock_shared_abstimed(ctx, thread, NULL)
00252 #define WOLFSENTRY_SHARED_OR_RETURN() do { \
00253     wolfentry_errcode_t _lock_ret;
00254     if (thread == NULL) \
00255         _lock_ret = WOLFSENTRY_MUTEX_EX(wolfentry); \
00256     else \
00257         _lock_ret = WOLFSENTRY_SHARED_EX(wolfentry); \
00258     WOLFSENTRY_RERETURN_IF_ERROR(_lock_ret); \
00259 } while (0)
00262 #define WOLFSENTRY_PROMOTABLE_EX(ctx)
wolfentry_context_lock_shared_with_reservation_abstimed(ctx, thread, NULL)
00265 #define WOLFSENTRY_PROMOTABLE_OR_RETURN() do { \
00266     wolfentry_errcode_t _lock_ret;
00267     if (thread == NULL) \
00268         _lock_ret = WOLFSENTRY_MUTEX_EX(wolfentry); \
00269     else \
00270         _lock_ret = WOLFSENTRY_PROMOTABLE_EX(wolfentry); \

```

```

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)wolfentry
00286     #define WOLFSENTRY_SHARED_EX(ctx) (void)(ctx)
00287     #define WOLFSENTRY_SHARED_OR_RETURN() (void)wolfentry
00288     #define WOLFSENTRY_PROMOTABLE_EX(ctx) (void)(ctx)
00289     #define WOLFSENTRY_PROMOTABLE_OR_RETURN() (void)wolfentry
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 { wolfentry_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 { wolfentry_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 { wolfentry_errcode_t _yret = (y); if (_yret < 0)
WOLFSENTRY_ERROR_RERETURN(_yret); } while (0)
00326 #define WOLFSENTRY_UNLOCK_AND_RERETURN_IF_ERROR(y) do { wolfentry_errcode_t _yret = (y); if (_yret <
0) { WOLFSENTRY_UNLOCK_FOR_RETURN(); WOLFSENTRY_ERROR_RERETURN(_yret); } } while (0)
00329 #ifndef WOLFSENTRY_ERROR_STRINGS
00330 WOLFSENTRY_API const char *wolfentry_errcode_source_string(wolfentry_errcode_t e);
00332 WOLFSENTRY_API const char *wolfentry_errcode_error_string(wolfentry_errcode_t e);
00334 WOLFSENTRY_API const char *wolfentry_errcode_error_name(wolfentry_errcode_t e);
00336 #endif
00337
00338 #if !defined(WOLFSENTRY_NO_STDIO) && !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 { wolfentry_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 && !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 wolfentry_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 #ifndef WOLFSENTRY_ERROR_STRINGS
00387 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_source_string_set(enum wolfsentry_source_id
00388     wolfsentry_source_id, const char *source_string);
00389 #define WOLFSENTRY_REGISTER_SOURCE() wolfsentry_user_source_string_set(WOLFSENTRY_SOURCE_ID, __FILE__)
00390 #endif
00391
00392 enum wolfsentry_error_id {
00393     WOLFSENTRY_ERROR_ID_OK              = 0,
00394     WOLFSENTRY_ERROR_ID_NOT_OK          = -1,
00395     WOLFSENTRY_ERROR_ID_INTERNAL_CHECK_FATAL = -2,
00396     WOLFSENTRY_ERROR_ID_SYS_OP_FATAL    = -3,
00397     WOLFSENTRY_ERROR_ID_SYS_OP_FAILED   = -4,
00398     WOLFSENTRY_ERROR_ID_SYS_RESOURCE_FAILED = -5,
00399     WOLFSENTRY_ERROR_ID_INCOMPATIBLE_STATE = -6,
00400     WOLFSENTRY_ERROR_ID_TIMED_OUT       = -7,
00401     WOLFSENTRY_ERROR_ID_INVALID_ARG     = -8,
00402     WOLFSENTRY_ERROR_ID_BUSY            = -9,
00403     WOLFSENTRY_ERROR_ID_INTERRUPTED     = -10,
00404     WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_BIG = -11,
00405     WOLFSENTRY_ERROR_ID_NUMERIC_ARG_TOO_SMALL = -12,
00406     WOLFSENTRY_ERROR_ID_STRING_ARG_TOO_LONG = -13,
00407     WOLFSENTRY_ERROR_ID_BUFFER_TOO_SMALL = -14,
00408     WOLFSENTRY_ERROR_ID_IMPLEMENTATION_MISSING = -15,
00409     WOLFSENTRY_ERROR_ID_ITEM_NOT_FOUND   = -16,
00410     WOLFSENTRY_ERROR_ID_ITEM_ALREADY_PRESENT = -17,
00411     WOLFSENTRY_ERROR_ID_ALREADY_STOPPED  = -18,
00412     WOLFSENTRY_ERROR_ID_WRONG_OBJECT     = -19,
00413     WOLFSENTRY_ERROR_ID_DATA_MISSING     = -20,
00414     WOLFSENTRY_ERROR_ID_NOT_PERMITTED    = -21,
00415     WOLFSENTRY_ERROR_ID_ALREADY          = -22,
00416     WOLFSENTRY_ERROR_ID_CONFIG_INVALID_KEY = -23,
00417     WOLFSENTRY_ERROR_ID_CONFIG_INVALID_VALUE = -24,
00418     WOLFSENTRY_ERROR_ID_CONFIG_OUT_OF_SEQUENCE = -25,
00419     WOLFSENTRY_ERROR_ID_CONFIG_UNEXPECTED = -26,
00420     WOLFSENTRY_ERROR_ID_CONFIG_MISPLACED_KEY = -27,
00421     WOLFSENTRY_ERROR_ID_CONFIG_PARSER     = -28,
00422     WOLFSENTRY_ERROR_ID_CONFIG_MISSING_HANDLER = -29,
00423     WOLFSENTRY_ERROR_ID_CONFIG_JSON_VALUE_SIZE = -30,
00424     WOLFSENTRY_ERROR_ID_OP_NOT_SUPP_FOR_PROTO = -31,
00425     WOLFSENTRY_ERROR_ID_WRONG_TYPE        = -32,
00426     WOLFSENTRY_ERROR_ID_BAD_VALUE         = -33,
00427     WOLFSENTRY_ERROR_ID_DEADLOCK_AVERTED = -34,
00428     WOLFSENTRY_ERROR_ID_OVERFLOW_AVERTED = -35,
00429     WOLFSENTRY_ERROR_ID_LACKING_MUTEX     = -36,
00430     WOLFSENTRY_ERROR_ID_LACKING_READ_LOCK = -37,
00431     WOLFSENTRY_ERROR_ID_LIB_MISMATCH      = -38,
00432     WOLFSENTRY_ERROR_ID_LIBCONFIG_MISMATCH = -39,
00433     WOLFSENTRY_ERROR_ID_IO_FAILED         = -40,
00434
00435     WOLFSENTRY_ERROR_ID_USER_BASE         = -128,
00436
00437     WOLFSENTRY_SUCCESS_ID_OK              = 0,
00438     WOLFSENTRY_SUCCESS_ID_LOCK_OK_AND_GOT_RESV = 1,
00439     WOLFSENTRY_SUCCESS_ID_HAVE_MUTEX       = 2,
00440     WOLFSENTRY_SUCCESS_ID_HAVE_READ_LOCK   = 3,
00441     WOLFSENTRY_SUCCESS_ID_USED_FALLBACK    = 4,
00442     WOLFSENTRY_SUCCESS_ID_YES              = 5,
00443     WOLFSENTRY_SUCCESS_ID_NO              = 6,
00444     WOLFSENTRY_SUCCESS_ID_ALREADY_OK       = 7,
00445     WOLFSENTRY_SUCCESS_ID_USER_BASE       = 128
00446 };
00447
00448 #ifndef WOLFSENTRY_ERROR_STRINGS
00449 WOLFSENTRY_API wolfsentry_errcode_t wolfsentry_user_error_string_set(enum wolfsentry_error_id
00450     wolfsentry_error_id, const char *message_string);
00451 #define WOLFSENTRY_REGISTER_ERROR(name, msg) wolfsentry_user_error_string_set(WOLFSENTRY_ERROR_ID_##
00452     name, msg)
00453 #endif
00454
00455 #endif /* WOLFSENTRY_ERRCODES_H */
```

## 10.10 wolfentry/wolfentry\_json.h File Reference

Types and prototypes for loading/reloading configuration using JSON.

```
#include "wolfentry.h"
#include "centijson_sax.h"
```

### Macros

- `#define WOLFSENTRY`
- `#define WOLFSENTRY_MAX_JSON_NESTING 16`

*Can be overridden.*

### Typedefs

- `typedef uint32_t wolfentry_config_load_flags_t`  
*Type for holding flag bits from [wolfentry\\_config\\_load\\_flags](#).*

### Enumerations

- `enum wolfentry_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_MAINTAININDICTORDER,`  
`WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES,`  
`WOLFSENTRY_CONFIG_LOAD_FLAG_FINI}`

*Flags to be OR'd together to communicate options to [wolfentry\\_config\\_json\\_init\(\)](#)*

### Functions

- `WOLFSENTRY_API wolfentry_errcode_t wolfentry_centijson_errcode_translate (wolfentry_errcode_t centijson_errcode)`  
*Convert CentiJSON numeric error code to closest-corresponding wolfSentry error code.*
- `WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_init (WOLFSENTRY_CONTEXT_ARGS_IN, wolfentry_config_load_flags_t load_flags, struct wolfentry_json_process_state **jps)`  
*Allocate and initialize a struct wolfentry\_json\_process\_state with the designated load\_flags, to subsequently pass to [wolfentry\\_config\\_json\\_feed\(\)](#).*
- `WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_init_ex (WOLFSENTRY_CONTEXT_ARGS_IN, wolfentry_config_load_flags_t load_flags, const JSON_CONFIG *json_config, struct wolfentry_json_↵ process_state **jps)`  
*Variant of [wolfentry\\_config\\_json\\_init\(\)](#) with an additional JSON\_CONFIG argument, json\_↵ config, for tailoring of JSON parsing dynamics.*
- `WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_feed (struct wolfentry_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_centijson\\_errcode](#) (struct wolfentry\_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 wolfentry\_json↔\_process\_state allocated by [wolfentry\\_config\\_json\\_init\(\)](#).

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_json\\_fini](#) (struct wolfentry\_json↔\_process\_state \*\*jps, char \*err\_buf, size\_t err\_buf\_size)

To be called when done iterating [wolfentry\\_config\\_json\\_feed\(\)](#), completing the configuration load.

- WOLFSENTRY\_API [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_json\\_oneshot](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const unsigned char \*json\_in, size\_t json\_in\_len, [wolfentry\\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_config\\_json\\_oneshot\\_ex](#) (WOLFSENTRY\_CONTEXT\_ARGS\_IN, const unsigned char \*json\_in, size\_t json\_in\_len, [wolfentry\\_config\\_load\\_flags\\_t](#) load\_flags, const [JSON\\_CONFIG](#) \*json\_config, char \*err\_buf, size\_t err\_buf\_size)

Variant of [wolfentry\\_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 wolfentry\_json.h

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

```
00001 /*
00002  * wolfentry_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 "wolfentry.h"
00033
00034 #ifdef WOLFSENTRY_NO_STDIO
00035 #error wolfentry_json requires stdio
00036 #endif
00037
00038 #ifndef WOLFSENTRY
00039 #define WOLFSENTRY
00040 #endif
00041 #include "centijson_sax.h"
00042
```

```

00047 WOLFSENTRY_API wolfentry_errcode_t wolfentry_centijson_errcode_translate(wolfentry_errcode_t
centijson_errcode);
00050 #ifndef WOLFSENTRY_MAX_JSON_NESTING
00051 #define WOLFSENTRY_MAX_JSON_NESTING 16
00053 #endif
00054
00055 typedef uint32_t wolfentry_config_load_flags_t;
00059 enum wolfentry_config_load_flags {
00060     WOLFSENTRY_CONFIG_LOAD_FLAG_NONE = 0U,
00062     WOLFSENTRY_CONFIG_LOAD_FLAG_NO_FLUSH = 1U << 0U,
00064     WOLFSENTRY_CONFIG_LOAD_FLAG_DRY_RUN = 1U << 1U,
00066     WOLFSENTRY_CONFIG_LOAD_FLAG_LOAD_THEN_COMMIT = 1U << 2U,
00068     WOLFSENTRY_CONFIG_LOAD_FLAG_NO_ROUTES_OR_EVENTS = 1U << 3U,
00070     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_ABORT = 1U << 4U,
00072     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USEFIRST = 1U << 5U,
00074     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_DUPKEY_USELAST = 1U << 6U,
00076     WOLFSENTRY_CONFIG_LOAD_FLAG_JSON_DOM_MAINTAININDICTORDER = 1U << 7U,
00078     WOLFSENTRY_CONFIG_LOAD_FLAG_FLUSH_ONLY_ROUTES = 1U << 8U,
00080     WOLFSENTRY_CONFIG_LOAD_FLAG_FINI = 1U << 30U
00082 };
00083
00084 struct wolfentry_json_process_state;
00085
00086 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_init(
00087     WOLFSENTRY_CONTEXT_ARGS_IN,
00088     wolfentry_config_load_flags_t load_flags,
00089     struct wolfentry_json_process_state **jps);
00092 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_init_ex(
00093     WOLFSENTRY_CONTEXT_ARGS_IN,
00094     wolfentry_config_load_flags_t load_flags,
00095     const JSON_CONFIG *json_config,
00096     struct wolfentry_json_process_state **jps);
00099 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_feed(
00100     struct wolfentry_json_process_state *jps,
00101     const unsigned char *json_in,
00102     size_t json_in_len,
00103     char *err_buf,
00104     size_t err_buf_size);
00107 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_centijson_errcode(struct
wolfentry_json_process_state *jps, int *json_errcode, const char **json_errmsg);
00110 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_fini(
00111     struct wolfentry_json_process_state **jps,
00112     char *err_buf,
00113     size_t err_buf_size);
00116 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_oneshot(
00117     WOLFSENTRY_CONTEXT_ARGS_IN,
00118     const unsigned char *json_in,
00119     size_t json_in_len,
00120     wolfentry_config_load_flags_t load_flags,
00121     char *err_buf,
00122     size_t err_buf_size);
00125 WOLFSENTRY_API wolfentry_errcode_t wolfentry_config_json_oneshot_ex(
00126     WOLFSENTRY_CONTEXT_ARGS_IN,
00127     const unsigned char *json_in,
00128     size_t json_in_len,
00129     wolfentry_config_load_flags_t load_flags,
00130     const JSON_CONFIG *json_config,
00131     char *err_buf,
00132     size_t err_buf_size);
00137 #endif /* WOLFSENTRY_JSON_H */

```

## 10.12 wolfentry/wolfentry\_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 [wolfentry\\_errcode\\_t](#) [wolfentry\\_install\\_lwip\\_filter\\_ethernet\\_callback](#)(WOLFSENTRY\_CONTEXT\_ARGS\_IN, 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\_ARGS, 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>
#include <errno.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 [WOLFSENTRY\\_USER\\_SETTINGS\\_FILE](#) 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`.*
- #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 typedefs or include must be supplied with [WOLFSENTRY\\_USER\\_SETTINGS\\_FILE](#)).*
- #define **WOLFSENTRY\_NO\_STDINT\_H**  
*Define to inhibit inclusion of `stdint.h` (alternative typedefs 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 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\_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](#)).*
- #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 struct [wolfSentry\\_timecbs](#).*
- #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 struct [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\_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\_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\_ENT\_ID\_FMT**  
*printf-style format string appropriate for pairing with `wolfentry_ent_id_t`*
- **#define WOLFSENTRY\_ENT\_ID\_NONE**  
*always-invalid object ID*
- **#define WOLFSENTRY\_HITCOUNT\_FMT**  
*printf-style format string appropriate for pairing with `wolfentry_hitcount_t`*
- **#define \_\_wolfentry\_wur**  
*abstracted attribute designating that the return value must be checked to avoid a compiler warning*
- **#define wolfentry\_static\_assert(c)**  
*abstracted static assert – `c` must be true, else `c` is printed*
- **#define wolfentry\_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 `wolfentry_get_thread_deadline()` when thread has no deadline set. Not allowed as explicit values passed to `wolfentry_set_deadline_abs()` – use `wolfentry_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 `wolfentry_get_thread_deadline()` when thread is in non-blocking mode. Not allowed as explicit values passed to `wolfentry_set_deadline_abs()` – use `wolfentry_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}**
- **#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. Incurs 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)*

### 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 #ifndef WOLFSENTRY_FOR_DOXYGEN
00036 #define WOLFSENTRY_USER_SETTINGS_FILE "the_path"
00038 #undef WOLFSENTRY_USER_SETTINGS_FILE
00039 #endif
00040
00041 #ifndef WOLFSENTRY_USER_SETTINGS_FILE
00042 #include WOLFSENTRY_USER_SETTINGS_FILE

```

```

00043 #endif
00044
00045 #ifndef BUILDING_LIBWOLFSENTRY
00046 #include <wolfentry/wolfentry_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))
00078 #else
00079 #define __attribute_maybe_unused__
00080 #endif
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) && !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) && !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_HAVE_NONGNU_ATOMICS
00191 #undef WOLFSENTRY_HAVE_NONGNU_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_HAVE_NONGNU_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_MALLOC_BUILTIN
00233 #undef WOLFSENTRY_NO_MALLOC_BUILTIN
00234
00235     #define WOLFSENTRY_NO_ERROR_STRINGS
00237 #undef WOLFSENTRY_NO_ERROR_STRINGS
00238
00239     #define WOLFSENTRY_NO_PROTOCOL_NAMES
00240 #undef WOLFSENTRY_NO_PROTOCOL_NAMES
00243 #endif /* WOLFSENTRY_FOR_DOXYGEN */
00244
00247 #ifndef WOLFSENTRY_NO_CLOCK_BUILTIN
00248     #define WOLFSENTRY_CLOCK_BUILTINS
00249 #endif
00250
00251 #ifndef WOLFSENTRY_NO_MALLOC_BUILTIN
00252     #define WOLFSENTRY_MALLOC_BUILTINS
00253 #endif
00254
00255 #ifndef WOLFSENTRY_NO_ERROR_STRINGS
00256     #define WOLFSENTRY_ERROR_STRINGS
00257 #endif
00258
00259 #ifndef WOLFSENTRY_NO_PROTOCOL_NAMES
00260     #define WOLFSENTRY_PROTOCOL_NAMES
00261 #endif
00262
00271 #if defined(WOLFSENTRY_USE_NATIVE_POSIX_SEMAPHORES) || defined(WOLFSENTRY_CLOCK_BUILTINS) ||
    defined(WOLFSENTRY_MALLOC_BUILTINS)
00272 #ifndef _XOPEN_SOURCE
```

```

00273 #if __STDC_VERSION__ >= 201112L
00274 #define _XOPEN_SOURCE 700
00275 #elif __STDC_VERSION__ >= 199901L
00276 #define _XOPEN_SOURCE 600
00277 #else
00278 #define _XOPEN_SOURCE 500
00279 #endif /* __STDC_VERSION__ */
00280 #endif
00281 #endif
00282
00283 #if !defined(WOLFSENTRY_NO_POSIX_MEMALIGN) && (!defined(_POSIX_C_SOURCE) || (_POSIX_C_SOURCE <
    200112L))
00284     #define WOLFSENTRY_NO_POSIX_MEMALIGN
00286 #endif
00287
00288 #if defined(__STRICT_ANSI__)
00289 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE 1
00290 #elif defined(__GNUC__) && !defined(__clang__)
00291 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE
00293 #else
00294 #define WOLFSENTRY_FLEXIBLE_ARRAY_SIZE 0
00295 #endif
00296
00299 #ifndef WOLFSENTRY_NO_TIME_H
00300 #ifndef __USE_POSIX199309
00301 /* glibc needs this for struct timespec with -std=c99 */
00302 #define __USE_POSIX199309
00303 #endif
00304 #endif
00305
00308 #ifdef SIZE_T_32
00309     #define SIZET_FMT "%u"
00310 #elif __STDC_VERSION__ >= 199901L
00311     #define SIZET_FMT "%zu"
00312 #else
00313     #define SIZET_FMT "%lu"
00315 #endif
00316
00317 #ifndef WOLFSENTRY_NO_STDDEF_H
00318 #include <stddef.h>
00319 #endif
00320 #ifndef WOLFSENTRY_NO_ASSERT_H
00321 #include <assert.h>
00322 #endif
00323 #ifndef WOLFSENTRY_NO_STDIO
00324 #ifndef __USE_ISOC99
00325 /* kludge to make glibc snprintf() prototype visible even when -std=c89 */
00327 #define __USE_ISOC99
00329 #include <stdio.h>
00330 #undef __USE_ISOC99
00331 #else
00332 #include <stdio.h>
00333 #endif
00334 #endif
00335 #ifndef WOLFSENTRY_NO_STRING_H
00336 #include <string.h>
00337 #endif
00338 #ifndef WOLFSENTRY_NO_STRINGS_H
00339 #include <strings.h>
00340 #endif
00341 #ifndef WOLFSENTRY_NO_TIME_H
00342 #include <time.h>
00343 #endif
00344
00345 #if !defined(WOLFSENTRY_NO_GETPROTOBY) && (!defined(__GLIBC__) || !defined(__USE_MISC) ||
    defined(WOLFSENTRY_C89))
00346     /* get*by*_r() is non-standard. */
00347     #define WOLFSENTRY_NO_GETPROTOBY
00349 #endif
00350
00351 typedef unsigned char byte;
00354 typedef uint16_t wolfentry_addr_family_t;
00357 typedef uint16_t wolfentry_proto_t;
00359 typedef uint16_t wolfentry_port_t;
00361 #ifdef WOLFSENTRY_ENT_ID_TYPE
00362 typedef WOLFSENTRY_ENT_ID_TYPE wolfentry_ent_id_t;
00363 #else
00364 typedef uint32_t wolfentry_ent_id_t;
00366 #define WOLFSENTRY_ENT_ID_FMT "%u"
00368 #endif
00369 #define WOLFSENTRY_ENT_ID_NONE 0
00371 typedef uint16_t wolfentry_addr_bits_t;
00373 #ifdef WOLFSENTRY_HITCOUNT_TYPE
00374 typedef WOLFSENTRY_HITCOUNT_TYPE wolfentry_hitcount_t;
00375 #else
00376 typedef uint32_t wolfentry_hitcount_t;
00378 #define WOLFSENTRY_HITCOUNT_FMT "%u"

```

```

00380 #endif
00381 #ifdef WOLFSENTRY_TIME_TYPE
00382 typedef WOLFSENTRY_TIME_TYPE wolfentry_time_t;
00383 #else
00384 typedef int64_t wolfentry_time_t;
00386 #endif
00387
00388 #ifdef WOLFSENTRY_PRIORITY_TYPE
00389 typedef WOLFSENTRY_PRIORITY_TYPE wolfentry_priority_t;
00390 #else
00391 typedef uint16_t wolfentry_priority_t;
00393 #endif
00394
00395 #ifndef attr_align_to
00396 #ifdef __GNUC__
00397 #define attr_align_to(x) __attribute__((aligned(x)))
00398 #elif defined(_MSC_VER)
00399 /* disable align warning, we want alignment ! */
00400 #pragma warning(disable: 4324)
00401 #define attr_align_to(x) __declspec(align(x))
00402 #else
00403 #error must supply definition for attr_align_to() macro.
00404 #endif
00405 #endif
00406
00407 #ifndef __wolfentry_wur
00408 #ifdef __wur
00409 #define __wolfentry_wur __wur
00410 #elif defined(__must_check)
00411 #define __wolfentry_wur __must_check
00412 #elif defined(__GNUC__) && (__GNUC__ >= 4)
00413 #define __wolfentry_wur __attribute__((warn_unused_result))
00415 #else
00416 #define __wolfentry_wur
00417 #endif
00418 #endif
00419
00420 #ifndef wolfentry_static_assert
00421 #if defined(__GNUC__) && defined(static_assert) && !defined(__STRICT_ANSI__)
00422 /* note semicolon included in expansion, so that assert can completely disappear in ISO C builds. */
00423 #define wolfentry_static_assert(c) static_assert(c, #c);
00424 #define wolfentry_static_assert2(c, m) static_assert(c, m);
00425 #else
00426 #define wolfentry_static_assert(c)
00428 #define wolfentry_static_assert2(c, m)
00430 #endif
00431 #endif /* !wolfentry_static_assert */
00432
00433 #if defined(WOLFSENTRY_THREADSAFE)
00440
00441 #ifndef WOLFSENTRY_DEADLINE_NEVER
00442 #define WOLFSENTRY_DEADLINE_NEVER (-1)
00444 #endif
00445 #ifndef WOLFSENTRY_DEADLINE_NOW
00446 #define WOLFSENTRY_DEADLINE_NOW (-2)
00448 #endif
00449
00450 #ifndef WOLFSENTRY_NO_ERRNO_H
00451 #include <errno.h>
00452 #endif
00453
00454 #ifdef WOLFSENTRY_USE_NATIVE_POSIX_SEMAPHORES
00455
00456 #ifndef __USE_XOPEN2K
00457 /* kludge to force glibc sem_timedwait() prototype visible with -std=c99 */
00458 #define __USE_XOPEN2K
00459 #include <semaphore.h>
00460 #undef __USE_XOPEN2K
00461 #else
00462 #include <semaphore.h>
00463 #endif
00464
00465 #elif defined(__MACH__)
00466
00467 #include <dispatch/dispatch.h>
00468 #include <semaphore.h>
00469 #define sem_t dispatch_semaphore_t
00470
00471 #elif defined(FREERTOS)
00472
00473 #include <semphr.h>
00474 #include <atomic.h>
00475
00476 #define SEM_VALUE_MAX 0x7FFFU
00477
00478 #define sem_t StaticSemaphore_t
00479

```

```

00480 #else
00481
00482 #ifdef WOLFSENTRY_SEMAPHORE_INCLUDE
00483 #include WOLFSENTRY_SEMAPHORE_INCLUDE
00484 #endif
00485
00486 #endif
00487
00488 #ifdef WOLFSENTRY_THREAD_INCLUDE
00489 #include WOLFSENTRY_THREAD_INCLUDE
00490 #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00491 #include <pthread.h>
00492 #endif
00493 #ifdef WOLFSENTRY_THREAD_ID_T
00494 typedef WOLFSENTRY_THREAD_ID_T wolfentry_thread_id_t;
00495 #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00496 typedef pthread_t wolfentry_thread_id_t;
00497 #elif defined(FREERTOS)
00498 typedef TaskHandle_t wolfentry_thread_id_t;
00499 #else
00500 #error Must supply WOLFSENTRY_THREAD_ID_T for WOLFSENTRY_THREADSAFE on non-POSIX targets.
00501 #endif
00502 /* note WOLFSENTRY_THREAD_GET_ID_HANDLER must return WOLFSENTRY_THREAD_NO_ID on failure. */
00503 #ifdef WOLFSENTRY_THREAD_GET_ID_HANDLER
00504 #elif defined(WOLFSENTRY_USE_NATIVE_POSIX_THREADS)
00505 #define WOLFSENTRY_THREAD_GET_ID_HANDLER pthread_self
00506 #elif defined(FREERTOS)
00507 #define WOLFSENTRY_THREAD_GET_ID_HANDLER xTaskGetCurrentTaskHandle
00508 #else
00509 #error Must supply WOLFSENTRY_THREAD_GET_ID_HANDLER for WOLFSENTRY_THREADSAFE on non-POSIX
targets.
00510 #endif
00511
00512 struct wolfentry_thread_context;
00513
00514 /* WOLFSENTRY_THREAD_NO_ID must be zero. */
00515 #define WOLFSENTRY_THREAD_NO_ID 0
00516
00517 struct wolfentry_thread_context_public {
00518     uint64_t opaque[8];
00519 };
00520
00521 #define WOLFSENTRY_THREAD_CONTEXT_PUBLIC_INITIALIZER {0}
00522 #endif
00523
00524 #ifdef BUILDING_LIBWOLFSENTRY
00525 #if defined(_MSC_VER) || defined(__MINGW32__) || defined(__CYGWIN__) || \
00526     defined(_WIN32_WCE)
00527     #if defined(WOLFSENTRY_DLL)
00528         #define WOLFSENTRY_API_BASE __declspec(dllexport)
00529     #else
00530         #define WOLFSENTRY_API_BASE
00531     #endif
00532 #elif defined(HAVE_VISIBILITY) && HAVE_VISIBILITY
00533     #define WOLFSENTRY_API_BASE __attribute__((visibility("default")))
00534     #define WOLFSENTRY_LOCAL_BASE __attribute__((visibility("hidden")))
00535 #elif defined(__SUNPRO_C) && (__SUNPRO_C >= 0x550)
00536     #define WOLFSENTRY_API_BASE __global
00537     #define WOLFSENTRY_LOCAL_BASE __hidden
00538 #else
00539     #define WOLFSENTRY_API_BASE
00540     #define WOLFSENTRY_LOCAL_BASE
00541 #endif /* HAVE_VISIBILITY */
00542 #else /* !BUILDING_LIBWOLFSENTRY */
00543 #if defined(_MSC_VER) || defined(__MINGW32__) || defined(__CYGWIN__) || \
00544     defined(_WIN32_WCE)
00545     #if defined(WOLFSENTRY_DLL)
00546         #define WOLFSENTRY_API_BASE __declspec(dllimport)
00547     #else
00548         #define WOLFSENTRY_API_BASE
00549     #endif
00550 #elif defined(HAVE_VISIBILITY) && HAVE_VISIBILITY
00551     #define WOLFSENTRY_API_BASE __attribute__((visibility("default")))
00552     #define WOLFSENTRY_LOCAL_BASE __attribute__((visibility("hidden")))
00553 #elif defined(__SUNPRO_C) && (__SUNPRO_C >= 0x550)
00554     #define WOLFSENTRY_API_BASE __global
00555     #define WOLFSENTRY_LOCAL_BASE __hidden
00556 #else
00557     #define WOLFSENTRY_API_BASE
00558     #define WOLFSENTRY_LOCAL_BASE
00559 #endif /* HAVE_VISIBILITY */
00560 #endif /* !BUILDING_LIBWOLFSENTRY */
00561
00562 #define WOLFSENTRY_API_VOID WOLFSENTRY_API_BASE void
00563 #define WOLFSENTRY_API WOLFSENTRY_API_BASE __wolfentry_wur
00564 #define WOLFSENTRY_LOCAL_VOID WOLFSENTRY_LOCAL_BASE void
00565 #define WOLFSENTRY_LOCAL WOLFSENTRY_LOCAL_BASE __wolfentry_wur
00566 #ifndef WOLFSENTRY_NO_DESIGNATED_INITIALIZERS
00567 #define WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00568 #endif
00569
00570 #endif
00571
00572 #endif
00573
00574 #endif
00575
00576 #endif
00577
00578 #endif
00579
00580 #endif
00581
00582 #endif
00583
00584 #endif

```

```

00585 #ifndef WOLFSENTRY_NO_LONG_LONG
00586 #define WOLFSENTRY_HAVE_LONG_LONG
00587 #endif
00588
00591 #ifndef WOLFSENTRY_MAX_ADDR_BYTES
00592 #define WOLFSENTRY_MAX_ADDR_BYTES 16
00594 #elif WOLFSENTRY_MAX_ADDR_BYTES * 8 > 0xffff
00595 #error WOLFSENTRY_MAX_ADDR_BYTES * 8 must fit in a uint16_t.
00596 #endif
00597
00598 #ifndef WOLFSENTRY_MAX_ADDR_BITS
00599 #define WOLFSENTRY_MAX_ADDR_BITS (WOLFSENTRY_MAX_ADDR_BYTES*8)
00601 #else
00602 #if WOLFSENTRY_MAX_ADDR_BITS > (WOLFSENTRY_MAX_ADDR_BYTES*8)
00603 #error WOLFSENTRY_MAX_ADDR_BITS is too large for given/default WOLFSENTRY_MAX_ADDR_BYTES
00604 #endif
00605 #endif
00606
00607 #ifndef WOLFSENTRY_MAX_LABEL_BYTES
00608 #define WOLFSENTRY_MAX_LABEL_BYTES 32
00610 #elif WOLFSENTRY_MAX_LABEL_BYTES > 0xff
00611 #error WOLFSENTRY_MAX_LABEL_BYTES must fit in a byte.
00612 #endif
00613
00614 #ifndef WOLFSENTRY_BUILTIN_LABEL_PREFIX
00615 #define WOLFSENTRY_BUILTIN_LABEL_PREFIX "%"
00617 #endif
00618
00619 #ifndef WOLFSENTRY_KV_MAX_VALUE_BYTES
00620 #define WOLFSENTRY_KV_MAX_VALUE_BYTES 16384
00622 #endif
00623
00624 #if defined(WOLFSENTRY_ENT_ID_TYPE) || \
00625     defined(WOLFSENTRY_HITCOUNT_TYPE) || \
00626     defined(WOLFSENTRY_TIME_TYPE) || \
00627     defined(WOLFSENTRY_PRIORITY_TYPE) || \
00628     defined(WOLFSENTRY_THREAD_ID_T) || \
00629     defined(SIZE_T_32) || \
00630     defined(SIZE_T_64)
00631 #define WOLFSENTRY_USER_DEFINED_TYPES
00632 #endif
00633
00642 enum wolfsentry_build_flags {
00643     WOLFSENTRY_CONFIG_FLAG_ENDIANNESSESS_ONE = (1U << 0U),
00644     WOLFSENTRY_CONFIG_FLAG_USER_DEFINED_TYPES = (1U << 1U),
00645     WOLFSENTRY_CONFIG_FLAG_THREADSAFE = (1U << 2U),
00646     WOLFSENTRY_CONFIG_FLAG_CLOCK_BUILTINS = (1U << 3U),
00647     WOLFSENTRY_CONFIG_FLAG_MALLOC_BUILTINS = (1U << 4U),
00648     WOLFSENTRY_CONFIG_FLAG_ERROR_STRINGS = (1U << 5U),
00649     WOLFSENTRY_CONFIG_FLAG_PROTOCOL_NAMES = (1U << 6U),
00650     WOLFSENTRY_CONFIG_FLAG_NO_STDIO = (1U << 7U),
00651     WOLFSENTRY_CONFIG_FLAG_NO_JSON = (1U << 8U),
00652     WOLFSENTRY_CONFIG_FLAG_HAVE_JSON_DOM = (1U << 9U),
00653     WOLFSENTRY_CONFIG_FLAG_DEBUG_CALL_TRACE = (1U << 10U),
00654     WOLFSENTRY_CONFIG_FLAG_LWIP = (1U << 11U),
00655     WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS = (1U << 12U),
00656     WOLFSENTRY_CONFIG_FLAG_MAX = WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS,
00657     WOLFSENTRY_CONFIG_FLAG_ENDIANNESSESS_ZERO = (0U << 31U)
00658 };
00659
00663 struct wolfsentry_build_settings {
00664     uint32_t version;
00666     uint32_t config;
00668 };
00669
00670 #if !defined(BUILDING_LIBWOLFSENTRY) || defined(WOLFSENTRY_DEFINE_BUILD_SETTINGS)
00671
00672 static const __attribute__((maybe_unused)) uint32_t __wolfsentry_config =
    WOLFSENTRY_CONFIG_FLAG_ENDIANNESSESS_ONE
00673 #ifdef WOLFSENTRY_USER_DEFINED_TYPES
00674     | WOLFSENTRY_CONFIG_FLAG_USER_DEFINED_TYPES
00675 #endif
00676 #ifdef WOLFSENTRY_THREADSAFE
00677     | WOLFSENTRY_CONFIG_FLAG_THREADSAFE
00678 #endif
00679 #ifdef WOLFSENTRY_CLOCK_BUILTINS
00680     | WOLFSENTRY_CONFIG_FLAG_CLOCK_BUILTINS
00681 #endif
00682 #ifdef WOLFSENTRY_MALLOC_BUILTINS
00683     | WOLFSENTRY_CONFIG_FLAG_MALLOC_BUILTINS
00684 #endif
00685 #ifdef WOLFSENTRY_ERROR_STRINGS
00686     | WOLFSENTRY_CONFIG_FLAG_ERROR_STRINGS
00687 #endif
00688 #ifdef WOLFSENTRY_PROTOCOL_NAMES
00689     | WOLFSENTRY_CONFIG_FLAG_PROTOCOL_NAMES
00690 #endif

```

```

00691 #ifdef WOLFSENTRY_NO_STDIO
00692     | WOLFSENTRY_CONFIG_FLAG_NO_STDIO
00693 #endif
00694 #ifdef WOLFSENTRY_NO_JSON
00695     | WOLFSENTRY_CONFIG_FLAG_NO_JSON
00696 #endif
00697 #ifdef WOLFSENTRY_HAVE_JSON_DOM
00698     | WOLFSENTRY_CONFIG_FLAG_HAVE_JSON_DOM
00699 #endif
00700 #ifdef WOLFSENTRY_DEBUG_CALL_TRACE
00701     | WOLFSENTRY_CONFIG_FLAG_DEBUG_CALL_TRACE
00702 #endif
00703 #ifdef WOLFSENTRY_LWIP
00704     | WOLFSENTRY_CONFIG_FLAG_LWIP
00705 #endif
00706 /* with compilers that can't evaluate the below expression as a compile-time
00707  * constant, WOLFSENTRY_SHORT_ENUMS can be defined in user settings to 0 or
00708  * 1 to avoid the dependency.
00709  */
00710 #ifdef WOLFSENTRY_SHORT_ENUMS
00711 #if WOLFSENTRY_SHORT_ENUMS == 0
00712     | WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS
00713 #endif
00714 #else
00715     | ((sizeof(wolfentry_init_flags_t) < sizeof(int)) ? WOLFSENTRY_CONFIG_FLAG_SHORT_ENUMS : 0)
00716 #endif
00717 ;
00718
00719 static __attribute__((maybe_unused)) struct wolfentry_build_settings wolfentry_build_settings = {
00720 #ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00721     .version =
00722 #endif
00723     WOLFSENTRY_VERSION,
00724 #ifdef WOLFSENTRY_HAVE_DESIGNATED_INITIALIZERS
00725     .config =
00726 #endif
00727     __wolfentry_config
00728 };
00731 #endif /* !BUILDING_LIBWOLFSENTRY || WOLFSENTRY_DEFINE_BUILD_SETTINGS */
00732
00735 #endif /* WOLFSENTRY_SETTINGS_H */

```

## 10.16 wolfentry/wolfentry\_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 WOLFENTRY\_BYTE\_STREAM\_DECLARE\_STACK(buf, bufsiz)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_DECLARE\_HEAP(buf, bufsiz)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_INIT\_HEAP(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_FREE\_HEAP(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_RESET(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_LEN(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_HEAD(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_BYTE\_STREAM\_PTR(buf)**  
*Byte stream helper macro.*
- **#define WOLFENTRY\_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 WOLFENTRY\_SET\_BITS(enumint, bits)**  
*Sets the designated bits in enumint.*
- **#define WOLFENTRY\_CHECK\_BITS(enumint, bits)**  
*Evaluates to true if bits are all set in enumint.*
- **#define WOLFENTRY\_CLEAR\_BITS(enumint, bits)**  
*Clears the designated bits in enumint.*
- **#define WOLFENTRY\_MASKIN\_BITS(enumint, bits)**  
*Evaluates to the bits that are set in both enumint and bits.*
- **#define WOLFENTRY\_MASKOUT\_BITS(enumint, bits)**  
*Evaluates to the bits that are set enumint but not set in bits.*
- **#define WOLFENTRY\_CLEAR\_ALL\_BITS(enumint)**  
*Clears all bits in enumint.*
- **#define BITS\_PER\_BYTE 8**
- **#define WOLFENTRY\_BITS\_TO\_BYTES(x)**  
*Evaluates to the number of bytes needed to represent x bits.*
- **#define WOLFENTRY\_ATOMIC\_INCREMENT(i, x)**  
*Adds x to i thread-safely, returning the sum.*
- **#define WOLFENTRY\_ATOMIC\_DECREMENT(i, x)**  
*Subtracts x from i thread-safely, returning the difference.*
- **#define WOLFENTRY\_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 [wolfentry.h](#).

## 10.17 wolfentry\_util.h

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

```
00001 /*
00002  * wolfentry_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))
00035 #endif
00036 #ifndef sizeof_field
00037 #define sizeof_field(structure, element) sizeof(((structure *)0)->element)
00038 #endif
00039 #ifndef instance_of_field
00040 #define instance_of_field(structure, element) (((structure *)0)->element)
00041 #endif
00042 #ifndef container_of
00043 #define container_of(ptr, container_type, member_name) ((container_type *) (void *) ((byte *) (ptr)) -
offsetof(container_type, member_name)))
00044 #endif
00045 #ifndef length_of_array
00046 #define length_of_array(x) (sizeof (x) / sizeof (x)[0])
00047 #endif
00048 #ifndef end_ptr_of_array
00049 #define end_ptr_of_array(x) (&(x)[length_of_array(x)])
00050 #endif
00051 #ifndef popcount32
00052 #ifdef __GNUC__
00053 #define popcount32(x) __builtin_popcount(x)
00054 #else
00055 #error Must supply binding for popcount32() on non-__GNUC__ targets.
00056 #endif
00057 #endif
00058 #if defined(__GNUC__) && !defined(WOLFSENTRY_NO_BUILTIN_CLZ)
00059 #ifndef LOG2_32
00060 #define LOG2_32(x) (31 - __builtin_clz((unsigned int)(x)))
00061 #endif
00062 #ifndef LOG2_64
00063 #define LOG2_64(x) ((sizeof(unsigned long long) * 8ULL) - (unsigned long
long)__builtin_clzll((unsigned long long)(x)) - 1ULL)
00064 #endif
00065 #endif
00066 #define streq(vs,fs,vs_len) (((vs_len) == strlen(fs)) && (memcmp(vs,fs,vs_len) == 0))
00067 #define strcaseeq(vs,fs,vs_len) (((vs_len) == strlen(fs)) && (strncasecmp(vs,fs,vs_len) == 0))
00068 #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
00069 #define WOLFSENTRY_BYTE_STREAM_DECLARE_HEAP(buf, bufsiz) static const size_t buf ## siz = (bufsiz);
unsigned char * (buf), *buf ## _p; size_t buf ## spc
00070 #define WOLFSENTRY_BYTE_STREAM_INIT_HEAP(buf) ((buf) = (unsigned char *)WOLFSENTRY_MALLOC(buf ## siz))
00071 #define WOLFSENTRY_BYTE_STREAM_FREE_HEAP(buf) WOLFSENTRY_FREE(buf)
00072 #define WOLFSENTRY_BYTE_STREAM_RESET(buf) do { (buf ## _p) = (buf); (buf ## spc) = (buf ## siz); }
while (0)
00073 #define WOLFSENTRY_BYTE_STREAM_LEN(buf) ((buf ## siz) - (buf ## spc))
00074 #define WOLFSENTRY_BYTE_STREAM_HEAD(buf) (buf)
00075 #define WOLFSENTRY_BYTE_STREAM_PTR(buf) (&(buf ## _p))
00076 #define WOLFSENTRY_BYTE_STREAM_SPC(buf) (&(buf ## spc))
00077 #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))
00078 #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))))
00079 #define WOLFSENTRY_SET_BITS(enumint, bits) ((enumint) |= (bits))
00080 #define WOLFSENTRY_CHECK_BITS(enumint, bits) (((enumint) & (bits)) == (bits))
00081 #define WOLFSENTRY_CLEAR_BITS(enumint, bits) ((enumint) &= ~(uint32_t)(bits))
00082 #define WOLFSENTRY_MASKIN_BITS(enumint, bits) ((enumint) & (bits))
00083 #define WOLFSENTRY_MASKOUT_BITS(enumint, bits) ((enumint) & ~(uint32_t)(bits))
00084 #define WOLFSENTRY_CLEAR_ALL_BITS(enumint) ((enumint) = 0)
00085 #ifndef BITS_PER_BYTE
00086 #define BITS_PER_BYTE 8
00087 #endif
00088 #define WOLFSENTRY_BITS_TO_BYTES(x) (((x) + 7U) >> 3U)
00089 /* helpers for stringifying the expanded value of a macro argument rather than its literal text: */
00090 #define _qq(x) #x
00091 #define _q(x) _qq(x)
00092 #ifndef WOLFSENTRY_THREADSAFE

```

```

00134
00135 #ifndef 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 */

```



# Index

## Action Subsystem, 80

- wolfentry\_action\_callback\_t, 82
- wolfentry\_action\_delete, 84
- wolfentry\_action\_drop\_reference, 85
- WOLFSENTRY\_ACTION\_FLAG\_DISABLED, 83
- WOLFSENTRY\_ACTION\_FLAG\_NONE, 83
- wolfentry\_action\_flags\_t, 83
- wolfentry\_action\_flush\_all, 85
- wolfentry\_action\_get\_flags, 86
- wolfentry\_action\_get\_label, 86
- wolfentry\_action\_get\_reference, 86
- wolfentry\_action\_insert, 87
- WOLFSENTRY\_ACTION\_RES\_ACCEPT, 83
- WOLFSENTRY\_ACTION\_RES\_BINDING, 84
- WOLFSENTRY\_ACTION\_RES\_CLOSED, 84
- WOLFSENTRY\_ACTION\_RES\_COMMENDABLE, 83
- WOLFSENTRY\_ACTION\_RES\_CONNECT, 83
- WOLFSENTRY\_ACTION\_RES\_CONNECTING\_OUT, 84
- WOLFSENTRY\_ACTION\_RES\_DEALLOCATED, 83
- WOLFSENTRY\_ACTION\_RES\_DEROGATORY, 83
- WOLFSENTRY\_ACTION\_RES\_DISCONNECT, 83
- WOLFSENTRY\_ACTION\_RES\_ERROR, 83
- WOLFSENTRY\_ACTION\_RES\_FALLTHROUGH, 83
- WOLFSENTRY\_ACTION\_RES\_INSERTED, 83
- WOLFSENTRY\_ACTION\_RES\_LISTENING, 84
- WOLFSENTRY\_ACTION\_RES\_NONE, 83
- WOLFSENTRY\_ACTION\_RES\_PORT\_RESET, 83
- WOLFSENTRY\_ACTION\_RES\_RECEIVED, 84
- WOLFSENTRY\_ACTION\_RES\_REJECT, 83
- WOLFSENTRY\_ACTION\_RES\_SENDING, 84
- WOLFSENTRY\_ACTION\_RES SOCK\_ERROR, 84
- WOLFSENTRY\_ACTION\_RES\_STOP, 83
- WOLFSENTRY\_ACTION\_RES\_STOPPED\_LISTENING, 84
- wolfentry\_action\_res\_t, 83
- WOLFSENTRY\_ACTION\_RES\_UNREACHABLE, 84
- WOLFSENTRY\_ACTION\_RES\_UPDATE, 83
- WOLFSENTRY\_ACTION\_RES\_USER\_BASE, 84
- WOLFSENTRY\_ACTION\_TYPE\_DECISION, 84
- WOLFSENTRY\_ACTION\_TYPE\_DELETE, 84

- WOLFSENTRY\_ACTION\_TYPE\_INSERT, 84
- WOLFSENTRY\_ACTION\_TYPE\_MATCH, 84
- WOLFSENTRY\_ACTION\_TYPE\_NONE, 84
- WOLFSENTRY\_ACTION\_TYPE\_POST, 84
- wolfentry\_action\_type\_t, 84
- WOLFSENTRY\_ACTION\_TYPE\_UPDATE, 84
- wolfentry\_action\_update\_flags, 87

## Address Family Subsystem, 101

### allocator

- wolfentry\_host\_platform\_interface, 139

### Allocator (Heap) Functions and Callbacks, 130

### b

- wolfentry\_kv\_pair, 140

## Building and Initializing wolfSentry for an application on FreeRTOS/LwIP, 3

### caller\_build\_settings

- wolfentry\_host\_platform\_interface, 139

### config

- wolfentry\_build\_settings, 137

## Configuring wolfSentry using a JSON document, 7

## Core Types and Macros, 43

## Diagnostics, Control Flow Helpers, and Compiler Attribute Helpers, 53

- WOLFSENTRY\_DEBUG\_CALL\_TRACE, 58

## Event Subsystem, 88

- wolfentry\_event\_action\_append, 91
- wolfentry\_event\_action\_delete, 91
- wolfentry\_event\_action\_insert\_after, 92
- wolfentry\_event\_action\_list\_done, 92
- wolfentry\_event\_action\_list\_next, 93
- wolfentry\_event\_action\_list\_start, 93
- wolfentry\_event\_action\_prepend, 94
- wolfentry\_event\_delete, 94
- wolfentry\_event\_drop\_reference, 95
- WOLFSENTRY\_EVENT\_FLAG\_IS\_PARENT\_EVENT, 90
- WOLFSENTRY\_EVENT\_FLAG\_IS\_SUBEVENT, 90
- WOLFSENTRY\_EVENT\_FLAG\_NONE, 90
- wolfentry\_event\_flags\_t, 90
- wolfentry\_event\_flush\_all, 95
- wolfentry\_event\_get\_config, 96
- wolfentry\_event\_get\_flags, 96
- wolfentry\_event\_get\_label, 96
- wolfentry\_event\_get\_reference, 97
- wolfentry\_event\_insert, 97

- wolfentry\_event\_set\_aux\_event, 98
- wolfentry\_event\_update\_config, 98
- wolfentry\_eventconfig\_check, 100
- WOLFSENTRY\_EVENTCONFIG\_FLAG\_COMMENDABLE\_CLEAR, 90
- WOLFSENTRY\_EVENTCONFIG\_FLAG\_DEROGATORY\_THRESHOLD, 90
- WOLFSENTRY\_EVENTCONFIG\_FLAG\_INHIBIT\_ACTIONS, 90
- WOLFSENTRY\_EVENTCONFIG\_FLAG\_NONE, 90
- wolfentry\_eventconfig\_flags\_t, 90
- wolfentry\_eventconfig\_init, 100
- JSON\_CALLBACKS, 135
- JSON\_CONFIG, 135
- JSON\_DOM\_PARSER, 135
- JSON\_INPUT\_POS, 136
- JSON\_PARSER, 136
- JSON\_VALUE, 136
- lwIP Callback Activation Functions, 134
- Object Subsystem, 109
  - wolfentry\_get\_object\_id, 110
  - wolfentry\_get\_object\_type, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_ACTION, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_ADDR\_FAMILY\_BYNAME, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_ADDR\_FAMILY\_BYNUMBER, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_EVENT, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_KV, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_ROUTE, 110
  - wolfentry\_object\_type\_t, 109
  - WOLFSENTRY\_OBJECT\_TYPE\_TABLE, 110
  - WOLFSENTRY\_OBJECT\_TYPE\_UNINITED, 110
  - wolfentry\_table\_n\_deletes, 110
  - wolfentry\_table\_n\_inserts, 111
- Route/Rule Subsystem, 58
  - WOLFSENTRY\_FORMAT\_FLAG\_ALWAYS\_NUMERIC, 65
  - WOLFSENTRY\_FORMAT\_FLAG\_NONE, 65
  - wolfentry\_format\_flags\_t, 64
  - wolfentry\_route\_bulk\_clear\_insert\_action\_status, 66
  - wolfentry\_route\_bulk\_insert\_actions, 66
  - wolfentry\_route\_delete, 66
  - wolfentry\_route\_delete\_by\_id, 67
  - wolfentry\_route\_drop\_reference, 68
  - wolfentry\_route\_event\_dispatch, 68
  - wolfentry\_route\_export, 69
  - wolfentry\_route\_exports\_render, 69
  - WOLFSENTRY\_ROUTE\_FLAG\_DELETE\_ACTIONS\_CALLED, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_DIRECTION\_IN, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_DIRECTION\_OUT, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_DONT\_COUNT\_CURRENT\_CONNECTIONS\_DEROGATORY, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_DONT\_COUNT\_HITS, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_GREENLISTED, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_IN\_TABLE, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_INSERT\_ACTIONS\_CALLED, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_LOCAL\_INTERFACE\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_NONE, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_PARENT\_EVENT\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_PENALTYBOXED, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_PENDING\_DELETE, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_PORT\_RESET, 66
  - WOLFSENTRY\_ROUTE\_FLAG\_REMOTE\_INTERFACE\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_FAMILY\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_LOCAL\_ADDR\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_LOCAL\_PORT\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_PROTO\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_REMOTE\_ADDR\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_SA\_REMOTE\_PORT\_WILDCARD, 65
  - WOLFSENTRY\_ROUTE\_FLAG\_TCPLIKE\_PORT\_NUMBERS, 65
  - wolfentry\_route\_flags\_t, 65
  - wolfentry\_route\_flush\_table, 70
  - wolfentry\_route\_get\_addrs, 70
  - wolfentry\_route\_get\_flags, 71
  - wolfentry\_route\_get\_main\_table, 71
  - wolfentry\_route\_get\_metadata, 71
  - wolfentry\_route\_get\_private\_data, 72
  - wolfentry\_route\_get\_reference, 72
  - wolfentry\_route\_insert, 73
  - wolfentry\_route\_parent\_event, 74
  - wolfentry\_route\_render, 74
  - wolfentry\_route\_set\_wildcard, 74
  - wolfentry\_route\_stale\_purge, 75
  - wolfentry\_route\_table\_default\_policy\_get, 75
  - wolfentry\_route\_table\_default\_policy\_set, 76
  - wolfentry\_route\_table\_fallthrough\_route\_get, 76
  - wolfentry\_route\_table\_iterate\_current, 77
  - wolfentry\_route\_table\_iterate\_end, 77
  - wolfentry\_route\_table\_iterate\_next, 77
  - wolfentry\_route\_table\_iterate\_prev, 78

- wolfsentry\_route\_table\_iterate\_seek\_to\_head, 78
  - wolfsentry\_route\_table\_iterate\_seek\_to\_tail, 79
  - wolfsentry\_route\_table\_iterate\_start, 79
  - wolfsentry\_route\_update\_flags, 79
- sem\_destroy\_cb\_t
  - Semaphore Function Callbacks, 133
- sem\_init\_cb\_t
  - Semaphore Function Callbacks, 133
- sem\_post\_cb\_t
  - Semaphore Function Callbacks, 133
- sem\_timedwait\_cb\_t
  - Semaphore Function Callbacks, 133
- sem\_trywait\_cb\_t
  - Semaphore Function Callbacks, 133
- sem\_wait\_cb\_t
  - Semaphore Function Callbacks, 133
- Semaphore Function Callbacks, 132
  - sem\_destroy\_cb\_t, 133
  - sem\_init\_cb\_t, 133
  - sem\_post\_cb\_t, 133
  - sem\_timedwait\_cb\_t, 133
  - sem\_trywait\_cb\_t, 133
  - sem\_wait\_cb\_t, 133
- semcbs
  - wolfsentry\_host\_platform\_interface, 139
- Startup/Configuration/Shutdown Subsystem, 44
  - WOLFSENTRY\_CLONE\_FLAG\_AS\_AT\_CREATION, 48
  - WOLFSENTRY\_CLONE\_FLAG\_NO\_ROUTES, 48
  - WOLFSENTRY\_CLONE\_FLAG\_NONE, 48
  - wolfsentry\_clone\_flags\_t, 48
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_DRY\_RUN, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_FINI, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_FLUSH\_ONLY, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_DURATION\_SECONDS, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_DURATION\_SECONDS\_FIRST, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_DURATION\_SECONDS\_LAST, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_MAXIMUM\_DURATION\_SECONDS, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_LOAD\_THEN\_COMMIT, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_NO\_FLUSH, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_NO\_ROUTES\_ON\_FAILURE, 49
  - WOLFSENTRY\_CONFIG\_LOAD\_FLAG\_NONE, 49
  - wolfsentry\_config\_load\_flags, 48
  - wolfsentry\_context\_clone, 49
  - wolfsentry\_context\_enable\_actions, 50
  - wolfsentry\_context\_exchange, 50
  - wolfsentry\_context\_flush, 50
  - wolfsentry\_context\_free, 51
  - wolfsentry\_context\_inhibit\_actions, 51
  - wolfsentry\_defaultconfig\_get, 51
  - wolfsentry\_defaultconfig\_update, 52
  - wolfsentry\_init, 52
  - WOLFSENTRY\_INIT\_FLAG\_LOCK\_SHARED\_ERROR\_CHECKING, 49
  - WOLFSENTRY\_INIT\_FLAG\_NONE, 49
  - wolfsentry\_init\_flags\_t, 49
  - wolfsentry\_shutdown, 53
- Thread Synchronization Subsystem, 111
  - wolfsentry\_lock\_alloc, 117
  - wolfsentry\_lock\_destroy, 118
  - WOLFSENTRY\_LOCK\_FLAG\_ABANDON\_RESERVATION\_TOO, 117
  - WOLFSENTRY\_LOCK\_FLAG\_AUTO\_DOWNGRADE, 117
  - WOLFSENTRY\_LOCK\_FLAG\_GET\_RESERVATION\_TOO, 117
  - WOLFSENTRY\_LOCK\_FLAG\_NONE, 116
  - WOLFSENTRY\_LOCK\_FLAG\_NONRECURSIVE\_MUTEX, 117
  - WOLFSENTRY\_LOCK\_FLAG\_NONRECURSIVE\_SHARED, 117
  - WOLFSENTRY\_LOCK\_FLAG\_PSHARED, 116
  - WOLFSENTRY\_LOCK\_FLAG\_RETAIN\_SEMAPHORE, 117
  - WOLFSENTRY\_LOCK\_FLAG\_SHARED\_ERROR\_CHECKING, 117
  - WOLFSENTRY\_LOCK\_FLAG\_TRY\_RESERVATION\_TOO, 117
  - wolfsentry\_lock\_flags\_t, 116
  - wolfsentry\_lock\_free, 118
  - wolfsentry\_lock\_get\_flags, 119
  - wolfsentry\_lock\_have\_either, 119
  - wolfsentry\_lock\_have\_mutex, 120
  - wolfsentry\_lock\_have\_shared, 120
  - wolfsentry\_lock\_have\_shared2mutex\_reservation, 121
  - wolfsentry\_lock\_init, 121
  - wolfsentry\_lock\_mutex, 122
  - wolfsentry\_lock\_mutex2shared, 122
  - wolfsentry\_lock\_mutex\_abstimed, 123
  - wolfsentry\_lock\_mutex\_timed, 123
  - wolfsentry\_lock\_shared, 124
  - wolfsentry\_lock\_shared2mutex, 124
  - wolfsentry\_lock\_shared2mutex\_abandon, 125
  - wolfsentry\_lock\_shared2mutex\_abstimed, 125
  - wolfsentry\_lock\_shared2mutex\_redeem, 126
  - wolfsentry\_lock\_shared2mutex\_redeem\_abstimed, 126
  - wolfsentry\_lock\_shared2mutex\_redeem\_timed, 127
  - wolfsentry\_lock\_shared2mutex\_reserve, 127
  - wolfsentry\_lock\_shared2mutex\_timed, 128
  - wolfsentry\_lock\_shared\_abstimed, 128
  - wolfsentry\_lock\_shared\_timed, 129
  - wolfsentry\_lock\_unlock, 129
  - WOLFSENTRY\_THREAD\_FLAG\_DEADLINE, 117



- WOLFSENTRY\_THREAD\_FLAG\_NONE, 117
- WOLFSENTRY\_THREAD\_FLAG\_READONLY, 117
- wolfentry\_thread\_flags\_t, 117
- Time Functions and Callbacks, 131
- timecbcs
  - wolfentry\_host\_platform\_interface, 139
- User-Defined Value Subsystem, 104
  - wolfentry\_kv\_validator\_t, 108
  - wolfentry\_user\_value\_get\_bytes, 108
  - wolfentry\_user\_value\_get\_json, 108
  - wolfentry\_user\_value\_get\_string, 108
- version
  - wolfentry\_build\_settings, 137
- wolfSentry Release History and Change Log, 17
- wolfentry/centijson\_dom.h, 145
- wolfentry/centijson\_sax.h, 147
- wolfentry/centijson\_value.h, 151
- wolfentry/wolfentry.h, 158, 180
- wolfentry/wolfentry\_af.h, 200, 203
- wolfentry/wolfentry\_errcodes.h, 204, 209
- wolfentry/wolfentry\_json.h, 214, 215
- wolfentry/wolfentry\_lwip.h, 216, 217
- wolfentry/wolfentry\_settings.h, 218, 221
- wolfentry/wolfentry\_util.h, 228, 230
- wolfSentry: the wolfSSL embedded firewall and IDPS, 1
- wolfentry\_action\_callback\_t
  - Action Subsystem, 82
- wolfentry\_action\_delete
  - Action Subsystem, 84
- wolfentry\_action\_drop\_reference
  - Action Subsystem, 85
- WOLFSENTRY\_ACTION\_FLAG\_DISABLED
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_FLAG\_NONE
  - Action Subsystem, 83
- wolfentry\_action\_flags\_t
  - Action Subsystem, 83
- wolfentry\_action\_flush\_all
  - Action Subsystem, 85
- wolfentry\_action\_get\_flags
  - Action Subsystem, 86
- wolfentry\_action\_get\_label
  - Action Subsystem, 86
- wolfentry\_action\_get\_reference
  - Action Subsystem, 86
- wolfentry\_action\_insert
  - Action Subsystem, 87
- WOLFSENTRY\_ACTION\_RES\_ACCEPT
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_BINDING
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_CLOSED
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_COMMENDABLE
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_CONNECT
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_CONNECTING\_OUT
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_DEALLOCATED
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_DEROGATORY
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_DISCONNECT
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_ERROR
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_FALLTHROUGH
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_INSERTED
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_LISTENING
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_NONE
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_PORT\_RESET
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_RECEIVED
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_REJECT
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_SENDING
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES SOCK\_ERROR
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_STOP
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_STOPPED\_LISTENING
  - Action Subsystem, 84
- wolfentry\_action\_res\_t
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_UNREACHABLE
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_RES\_UPDATE
  - Action Subsystem, 83
- WOLFSENTRY\_ACTION\_RES\_USER\_BASE
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_DECISION
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_DELETE
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_INSERT
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_MATCH
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_NONE
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_POST
  - Action Subsystem, 84
- wolfentry\_action\_type\_t
  - Action Subsystem, 84
- WOLFSENTRY\_ACTION\_TYPE\_UPDATE
  - Action Subsystem, 84



- wolfentry\_action\_update\_flags
  - Action Subsystem, [87](#)
- wolfentry\_allocator, [137](#)
- wolfentry\_build\_settings, [137](#)
  - config, [137](#)
  - version, [137](#)
- WOLFENTRY\_CLONE\_FLAG\_AS\_AT\_CREATION
  - Startup/Configuration/Shutdown Subsystem, [48](#)
- WOLFENTRY\_CLONE\_FLAG\_NO\_ROUTES
  - Startup/Configuration/Shutdown Subsystem, [48](#)
- WOLFENTRY\_CLONE\_FLAG\_NONE
  - Startup/Configuration/Shutdown Subsystem, [48](#)
- wolfentry\_clone\_flags\_t
  - Startup/Configuration/Shutdown Subsystem, [48](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_DRY\_RUN
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_FINI
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_FLUSH\_ONLY\_ROUTES
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_DUPLICATE\_ABORT
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_DUPLICATE\_ABORT
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_JSON\_DOM\_MAINTAIN\_COPIES
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_LOAD\_THEN\_COMMIT
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_NO\_FLUSH
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_NO\_ROUTES\_OR\_EVENTS
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFENTRY\_CONFIG\_LOAD\_FLAG\_NONE
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- wolfentry\_config\_load\_flags
  - Startup/Configuration/Shutdown Subsystem, [48](#)
- wolfentry\_context\_clone
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- wolfentry\_context\_enable\_actions
  - Startup/Configuration/Shutdown Subsystem, [50](#)
- wolfentry\_context\_exchange
  - Startup/Configuration/Shutdown Subsystem, [50](#)
- wolfentry\_context\_flush
  - Startup/Configuration/Shutdown Subsystem, [50](#)
- wolfentry\_context\_free
  - Startup/Configuration/Shutdown Subsystem, [51](#)
- wolfentry\_context\_inhibit\_actions
  - Startup/Configuration/Shutdown Subsystem, [51](#)
- WOLFENTRY\_DEBUG\_CALL\_TRACE
  - Diagnostics, Control Flow Helpers, and Compiler Attribute Helpers, [58](#)
- wolfentry\_defaultconfig\_get
  - Startup/Configuration/Shutdown Subsystem, [51](#)
- wolfentry\_defaultconfig\_update
  - Startup/Configuration/Shutdown Subsystem, [52](#)
- wolfentry\_event\_action\_append
  - Event Subsystem, [91](#)
- wolfentry\_event\_action\_delete
  - Event Subsystem, [91](#)
- wolfentry\_event\_action\_insert\_after
  - Event Subsystem, [92](#)
- wolfentry\_event\_action\_list\_done
  - Event Subsystem, [92](#)
- wolfentry\_event\_action\_list\_next
  - Event Subsystem, [93](#)
- wolfentry\_event\_action\_list\_start
  - Event Subsystem, [93](#)
- wolfentry\_event\_action\_prepend
  - Event Subsystem, [94](#)
- wolfentry\_event\_delete
  - Event Subsystem, [94](#)
- wolfentry\_event\_drop\_reference
  - Event Subsystem, [95](#)
- WOLFENTRY\_EVENT\_FLAG\_IS\_PARENT\_EVENT
  - Event Subsystem, [90](#)
- WOLFENTRY\_EVENT\_FLAG\_IS\_SUBEVENT
  - Event Subsystem, [90](#)
- WOLFENTRY\_EVENT\_FLAG\_NONE
  - Event Subsystem, [90](#)
- wolfentry\_event\_flags\_t
  - Event Subsystem, [90](#)
- wolfentry\_event\_flush\_all
  - Event Subsystem, [95](#)
- wolfentry\_event\_get\_config
  - Event Subsystem, [96](#)
- wolfentry\_event\_get\_flags
  - Event Subsystem, [96](#)
- wolfentry\_event\_get\_label
  - Event Subsystem, [96](#)
- wolfentry\_event\_get\_reference
  - Event Subsystem, [97](#)
- wolfentry\_event\_insert
  - Event Subsystem, [97](#)
- wolfentry\_event\_set\_aux\_event
  - Event Subsystem, [98](#)
- wolfentry\_event\_update\_config
  - Event Subsystem, [98](#)
- wolfentry\_eventconfig, [138](#)
- wolfentry\_eventconfig\_check
  - Event Subsystem, [100](#)
- WOLFENTRY\_EVENTCONFIG\_FLAG\_COMMENDABLE\_CLEARS\_DEFAULTS
  - Event Subsystem, [90](#)
- WOLFENTRY\_EVENTCONFIG\_FLAG\_DEROGATORY\_THRESHOLD
  - Event Subsystem, [90](#)
- WOLFENTRY\_EVENTCONFIG\_FLAG\_INHIBIT\_ACTIONS
  - Event Subsystem, [90](#)
- WOLFENTRY\_EVENTCONFIG\_FLAG\_NONE
  - Event Subsystem, [90](#)
- wolfentry\_eventconfig\_flags\_t
  - Event Subsystem, [90](#)
- wolfentry\_eventconfig\_init
  - Event Subsystem, [100](#)
- WOLFENTRY\_FORMAT\_FLAG\_ALWAYS\_NUMERIC
  - Route/Rule Subsystem, [65](#)

- WOLFSENTRY\_FORMAT\_FLAG\_NONE
  - Route/Rule Subsystem, [65](#)
- wolfentry\_format\_flags\_t
  - Route/Rule Subsystem, [64](#)
- wolfentry\_get\_object\_id
  - Object Subsystem, [110](#)
- wolfentry\_get\_object\_type
  - Object Subsystem, [110](#)
- wolfentry\_host\_platform\_interface, [139](#)
  - allocator, [139](#)
  - caller\_build\_settings, [139](#)
  - semcbs, [139](#)
  - timecbs, [139](#)
- wolfentry\_init
  - Startup/Configuration/Shutdown Subsystem, [52](#)
- WOLFSENTRY\_INIT\_FLAG\_LOCK\_SHARED\_ERROR\_CHECKING
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- WOLFSENTRY\_INIT\_FLAG\_NONE
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- wolfentry\_init\_flags\_t
  - Startup/Configuration/Shutdown Subsystem, [49](#)
- wolfentry\_kv\_pair, [140](#)
  - b, [140](#)
- wolfentry\_kv\_validator\_t
  - User-Defined Value Subsystem, [108](#)
- wolfentry\_lock\_alloc
  - Thread Synchronization Subsystem, [117](#)
- wolfentry\_lock\_destroy
  - Thread Synchronization Subsystem, [118](#)
- WOLFSENTRY\_LOCK\_FLAG\_ABANDON\_RESERVATION
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_AUTO\_DOWNGRADE
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_GET\_RESERVATION\_TOO
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_NONE
  - Thread Synchronization Subsystem, [116](#)
- WOLFSENTRY\_LOCK\_FLAG\_NONRECURSIVE\_MUTEX
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_NONRECURSIVE\_SHARED
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_PSHARED
  - Thread Synchronization Subsystem, [116](#)
- WOLFSENTRY\_LOCK\_FLAG\_RETAIN\_SEMAPHORE
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_SHARED\_ERROR\_CHECKING
  - Thread Synchronization Subsystem, [117](#)
- WOLFSENTRY\_LOCK\_FLAG\_TRY\_RESERVATION\_TOO
  - Thread Synchronization Subsystem, [117](#)
- wolfentry\_lock\_flags\_t
  - Thread Synchronization Subsystem, [116](#)
- wolfentry\_lock\_free
  - Thread Synchronization Subsystem, [118](#)
- wolfentry\_lock\_get\_flags
  - Thread Synchronization Subsystem, [119](#)
- wolfentry\_lock\_have\_either
  - Thread Synchronization Subsystem, [119](#)
- wolfentry\_lock\_have\_mutex
  - Thread Synchronization Subsystem, [120](#)
- wolfentry\_lock\_have\_shared
  - Thread Synchronization Subsystem, [120](#)
- wolfentry\_lock\_have\_shared2mutex\_reservation
  - Thread Synchronization Subsystem, [121](#)
- wolfentry\_lock\_init
  - Thread Synchronization Subsystem, [121](#)
- wolfentry\_lock\_mutex
  - Thread Synchronization Subsystem, [122](#)
- wolfentry\_lock\_mutex2shared
  - Thread Synchronization Subsystem, [122](#)
- wolfentry\_lock\_mutex\_abstimed
  - Thread Synchronization Subsystem, [123](#)
- wolfentry\_lock\_mutex\_timed
  - Thread Synchronization Subsystem, [123](#)
- wolfentry\_lock\_shared
  - Thread Synchronization Subsystem, [124](#)
- wolfentry\_lock\_shared2mutex
  - Thread Synchronization Subsystem, [124](#)
- wolfentry\_lock\_shared2mutex\_abandon
  - Thread Synchronization Subsystem, [125](#)
- wolfentry\_lock\_shared2mutex\_abstimed
  - Thread Synchronization Subsystem, [125](#)
- wolfentry\_lock\_shared2mutex\_redeem
  - Thread Synchronization Subsystem, [126](#)
- wolfentry\_lock\_shared2mutex\_redeem\_abstimed
  - Thread Synchronization Subsystem, [126](#)
- wolfentry\_lock\_shared2mutex\_redeem\_timed
  - Thread Synchronization Subsystem, [127](#)
- wolfentry\_lock\_shared2mutex\_reserve
  - Thread Synchronization Subsystem, [127](#)
- wolfentry\_lock\_shared2mutex\_timed
  - Thread Synchronization Subsystem, [128](#)
- wolfentry\_lock\_shared\_abstimed
  - Thread Synchronization Subsystem, [128](#)
- wolfentry\_lock\_shared\_timed
  - Thread Synchronization Subsystem, [129](#)
- wolfentry\_lock\_unlock
  - Thread Synchronization Subsystem, [129](#)
- WOLFSENTRY\_OBJECT\_TYPE\_ACTION
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_ADDR\_FAMILY\_BYNAME
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_ADDR\_FAMILY\_BYNUMBER
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_EVENT
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_KV
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_ROUTE
  - Object Subsystem, [110](#)
- wolfentry\_object\_type\_t
  - Object Subsystem, [109](#)
- WOLFSENTRY\_OBJECT\_TYPE\_TABLE
  - Object Subsystem, [110](#)
- WOLFSENTRY\_OBJECT\_TYPE\_UNINITED
  - Object Subsystem, [110](#)
- wolfentry\_route\_bulk\_clear\_insert\_action\_status

- Route/Rule Subsystem, [66](#)
- wolfentry\_route\_bulk\_insert\_actions
  - Route/Rule Subsystem, [66](#)
- wolfentry\_route\_delete
  - Route/Rule Subsystem, [66](#)
- wolfentry\_route\_delete\_by\_id
  - Route/Rule Subsystem, [67](#)
- wolfentry\_route\_drop\_reference
  - Route/Rule Subsystem, [68](#)
- wolfentry\_route\_endpoint, [140](#)
- wolfentry\_route\_event\_dispatch
  - Route/Rule Subsystem, [68](#)
- wolfentry\_route\_export
  - Route/Rule Subsystem, [69](#)
- wolfentry\_route\_exports, [141](#)
- wolfentry\_route\_exports\_render
  - Route/Rule Subsystem, [69](#)
- WOLFENTRY\_ROUTE\_FLAG\_DELETE\_ACTIONS\_CALLED
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_DIRECTION\_IN
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_DIRECTION\_OUT
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_DONT\_COUNT\_CURRENT\_CONNECTIONS
  - Route/Rule Subsystem, [66](#)
- WOLFENTRY\_ROUTE\_FLAG\_DONT\_COUNT\_HITS
  - Route/Rule Subsystem, [66](#)
- WOLFENTRY\_ROUTE\_FLAG\_GREENLISTED
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_IN\_TABLE
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_INSERT\_ACTIONS\_CALLED
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_LOCAL\_INTERFACE\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_NONE
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_PARENT\_EVENT\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_PENALTYBOXED
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_PENDING\_DELETE
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_PORT\_RESET
  - Route/Rule Subsystem, [66](#)
- WOLFENTRY\_ROUTE\_FLAG\_REMOTE\_INTERFACE\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_FAMILY\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_LOCAL\_ADDR\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_LOCAL\_PORT\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_PROTO\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_REMOTE\_ADDR\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_SA\_REMOTE\_PORT\_WILDCARD
  - Route/Rule Subsystem, [65](#)
- Route/Rule Subsystem, [65](#)
- WOLFENTRY\_ROUTE\_FLAG\_TCPLIKE\_PORT\_NUMBERS
  - Route/Rule Subsystem, [65](#)
- wolfentry\_route\_flags\_t
  - Route/Rule Subsystem, [65](#)
- wolfentry\_route\_flush\_table
  - Route/Rule Subsystem, [70](#)
- wolfentry\_route\_get\_addr
  - Route/Rule Subsystem, [70](#)
- wolfentry\_route\_get\_flags
  - Route/Rule Subsystem, [71](#)
- wolfentry\_route\_get\_main\_table
  - Route/Rule Subsystem, [71](#)
- wolfentry\_route\_get\_metadata
  - Route/Rule Subsystem, [71](#)
- wolfentry\_route\_get\_private\_data
  - Route/Rule Subsystem, [72](#)
- wolfentry\_route\_get\_reference
  - Route/Rule Subsystem, [72](#)
- wolfentry\_route\_insert
  - Route/Rule Subsystem, [73](#)
- wolfentry\_route\_metadata\_exports, [142](#)
- wolfentry\_route\_parent\_event
  - Route/Rule Subsystem, [74](#)
- wolfentry\_route\_render
  - Route/Rule Subsystem, [74](#)
- wolfentry\_route\_set\_wildcard
  - Route/Rule Subsystem, [74](#)
- wolfentry\_route\_stale\_purge
  - Route/Rule Subsystem, [75](#)
- wolfentry\_route\_table\_default\_policy\_get
  - Route/Rule Subsystem, [75](#)
- wolfentry\_route\_table\_default\_policy\_set
  - Route/Rule Subsystem, [76](#)
- wolfentry\_route\_table\_fallthrough\_route\_get
  - Route/Rule Subsystem, [76](#)
- wolfentry\_route\_table\_iterate\_current
  - Route/Rule Subsystem, [77](#)
- wolfentry\_route\_table\_iterate\_end
  - Route/Rule Subsystem, [77](#)
- wolfentry\_route\_table\_iterate\_next
  - Route/Rule Subsystem, [77](#)
- wolfentry\_route\_table\_iterate\_prev
  - Route/Rule Subsystem, [78](#)
- wolfentry\_route\_table\_iterate\_seek\_to\_head
  - Route/Rule Subsystem, [78](#)
- wolfentry\_route\_table\_iterate\_seek\_to\_tail
  - Route/Rule Subsystem, [79](#)
- wolfentry\_route\_table\_iterate\_start
  - Route/Rule Subsystem, [79](#)
- wolfentry\_route\_update\_flags
  - Route/Rule Subsystem, [79](#)
- wolfentry\_semcb, [142](#)
- wolfentry\_shutdown
  - Startup/Configuration/Shutdown Subsystem, [53](#)
- wolfentry\_sockaddr, [143](#)
- wolfentry\_table\_n\_deletes
  - Object Subsystem, [110](#)

- wolfentry\_table\_n\_inserts
  - Object Subsystem, [111](#)
- wolfentry\_thread\_context\_public, [144](#)
- WOLFENTRY\_THREAD\_FLAG\_DEADLINE
  - Thread Synchronization Subsystem, [117](#)
- WOLFENTRY\_THREAD\_FLAG\_NONE
  - Thread Synchronization Subsystem, [117](#)
- WOLFENTRY\_THREAD\_FLAG\_READONLY
  - Thread Synchronization Subsystem, [117](#)
- wolfentry\_thread\_flags\_t
  - Thread Synchronization Subsystem, [117](#)
- wolfentry\_timecbs, [144](#)
- wolfentry\_user\_value\_get\_bytes
  - User-Defined Value Subsystem, [108](#)
- wolfentry\_user\_value\_get\_json
  - User-Defined Value Subsystem, [108](#)
- wolfentry\_user\_value\_get\_string
  - User-Defined Value Subsystem, [108](#)