Instruction

Z-Ware SDK 7.13.0 Library C API Reference Manual

Document No.:	INS14416
Version:	10
Description:	The Z-Ware Library is a Z-Wave Plus v2 SmartStart Z-Wave for IP client.
Written By:	DCHOW;SAMBAT;YANYAN
Date:	2019-12-06
Reviewed By:	KSUNDARAM;YANYAN;JCC;SCBROWNI;TRBOYD
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2019-12-06	01:55:33	NTJ	Niels Johansen	

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20180713	DCHOW	ALL	- Cloned from INS14129-5 for SDK 7.00.00 - v9.03 -Modified tables "Controlled Z Wave CCs", "ZIPGW SDK 2.8x Supported Z Wave CCs", "Interface Types: IF_REC_TYPE_XXX", "Error Codes, ZW_ERR_XXX", "zwnetd_t structure", "if_rec_meter_t structure","rec union", "NODE_PROPTY_XXX", "zwnoded_t structure", "zwrep_group_fn Parameters", "zwif_group_get Parameters", "zwrep_wakeup_fn Parameters", "zwif_wakeup_get Parameters", "zwrep_switch_fn Parameters", "zwif_switch_set Parameters", "zwrep_level_fn Parameters", "ZW_METER_TYPE_XXX and ZW_METER_UNIT_XXX", "ZW_METER_SUP_UNIT_XXX", "zwmeter_dat_t structure", "zwmeter_cap_t structure", "zwif_meter_get Parameters", "zwrep_dlck_op_fn Parameters", "zwdlck_cfg_t structure", "zwrep_dlck_cfg_fn Parameters", "zwrep_thrmo_fan_md_fn Parameters", "zwrep_thrmo_md_fn Parameters", "zwrep_thrmo_setp_fn Parameters", "zwrep_thrmo_setp_range_fn Parameters", "zwif_thrmo_setp_sup_range_cache_get Parameters", "zwif_ind_rpt_set Parameters", "zwif_ind_get Parameters", "zwif_ind_set Parameters", "zwrep_barrier_fn Parameters", "zwrep_barrier_subsys_fn Parameters" -Added tables "if_rec_dlck_t structure", "Thermostat Setpoint Types: ZW_THRMO_SETP_TYP_XXX", "Thermostat Setpoint Unit", "Door lock Operation Mode", "Door lock Operation Type", "zwsw_ver_t structure", "zwrep_ind_fn Parameters", "zwind_data_t structure", "zwind_propty_val_t structure", "ZWIND_ID_XXX", "ZWIND_PPTY_ID_XXX" -Modified section "Firmware update". -Added sections "zwnet_identify", "zwnet_version", "zwnode_identify", "zwif_switch_mset", "zwif_dlck_cap_get", "zwif_dlck_cap_cache_get", "zwif_dlck_cap_free", "zwif_ind_sup_get", "zwif_ind_sup_cache_get", "zwif_ind_sup_free", "Sound Switch Interface API", "Time and Date Interface API". - control SOUND_SWITCH v1, INDICATOR CC v3, TIME CC v2
	20180718 20180802	DCHOW SNA	ALL	-Modified tables "zwnet_init_t structure", "Files", "ZIPGW SDK 2.1x Supported Z Wave CCs" -Modified sections "Role", "CC Support", "zwnet_identify", "zwnode_identify", "Node Update". - Added ZIPGW 7.x CCs, separated Z-Ware supported CCs.
	20180803	DCHOW	ALL	- v9.05
2	20180823	SNA DCHOW	ALL 17	Formatted for Silabs and fixed page numbers. Replaced S2 inclusion flow diagram as picture.
3	20181113	DCHOW	ALL	-Modified tables "Error Codes, ZW_ERR_XXX", "zwnet_notify_fn Parameters", "zwnet_node_fn Parameters", "zwusrcod_t structure" and "ZWIND_PPTY_ID_XXX". -Added table “ZW_USRCOD_XXX”. -Modified section "zwif_battery_rpt_set". -Support callbacks when node has been reset.
4	20181122	SNA		V9.05.04 SDK v7.00.02 beta: removed older ZIPGW info, Corrected mailbox CC support version; removed BBB references.

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
5	20190222	DCHOW	ALL	-v9.11 -Modified tables "Controlled Z Wave CCs", "Z Wave Alarm/Notification Event Parameter Type", "zwalrm_t Structure", "zwnet_add_sec2_grant_key Parameters", "zwnoded_t Structure", "zwcfg_info_t structure", "ZW_USRCOD_XXX", "zwusrcod_t structure", "zwif_usrcod_get Parameters", "zwif_usrcod_sup_get Parameters", "zwrep_usr_sup_fn Parameters", "zwif_usrcod_sup_cache_get", "zwrep_ind_sup_fn Parameters", "zwif_ind_set Parameters", "zwfw_updt_req_t Structure", "pl_info_t Structure", "zwnet_sts_t Structure", "zwnet_node_fn Parameters" -Added new entries in tables "Error Codes, ZW_ERR_XXX", "zwnet_notify_fn Parameters", "zwnet_sts_t Structure", "dev_global_sett_t Structure", "zwif_switch_set Parameters", "zwrep_color_sw_get_fn Parameters", "zwrep_basic_fn Parameters", "ZW_ALRM_XXX", "Z Wave Alarm/Notification Type", "– Z Wave Alarm/Notification Event", "zwif_snd_switch_rpt_set Parameters", "zwrep_snd_switch_config_fn Parameters", "ZW_ALRM_EVT_XXX" -Added tables "Additional Info Associated to Network Op and Status" -Modified sections "JSON File Format", "zwnet_add_sec2_grant_key", "Node Update" -Added sections "zwif_usrcod_rpt_set", "zwif_usrcod_ext_rpt_set", "zwif_usrcod_ext_set", "zwif_usrcod_ext_get", "zwif_usrcod_cap_get", "zwif_usrcod_cap_cache_get", "zwif_usrcod_kp_mod_rpt_set", "zwif_usrcod_kp_mod_set", "zwif_usrcod_kp_mod_get", "zwif_usrcod_ms_cod_rpt_set", "zwif_usrcod_ms_cod_set", "zwif_usrcod_ms_cod_get", "zwif_usrcod_chksum_rpt_set", "zwif_usrcod_chksum_get", "zwif_fw_downld_req", "Window Covering Interface API", "Node Update endpoint get/set CCs" - control User Code CC v2, Window Covering CC v1, Notification CC v8, N/W IMA CC v2
6	20190318	SNA	192	-v9.12 -Update supported CC table -Added firmware update completion status for low/unknown battery level
7	20190325	DCHOW	112	-v9.13 -Changed the description of ep_id in Table 193 - grp_member_t structure -Changed the description of grp_member in Table 195 - zwif_group_del Parameters
8	20190531	DCHOW	ALL	-v9.15 -Added parameters to zwif_switch_set API to support post-set polling -Added entries to tables ZW_THRMO_FAN_MD_XXX, ZW_FW_UPDT_ERR_XXX, "zwfw_info_t structure" -Modified tables "Controlled Z Wave CCs", "Supported Z-Wave CCs Pushed down from Z-Ware" -Added description on backoff intervals for "down" node in section "Background Polling" -Support and control Association CC v3 & Multichannel Association CC v4 -Control Firmware Update MD CC v6,- Thermostat Fan Mode CC v5, -Added section “Persistent Storage for Z-Ware Library”
9	20190816	SNA	ALL	Moved all non API & File format documentation out

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
	20190902	TWC	ALL	-v9.22 -Added IF_REC_TYPE_MULTI_CMD in table "Interface Types: IF_REC_TYPE_XXX" -Added entries in tables "ZW_ALRM_EVT_XXX", "rec Union", "ZWIND_ID_XXX", "ZWIND_PPTY_ID_XXX", "zwfw_info_t structure", "zwfw_updt_req_t Structure", "zwrep_snd_switch_tone_play_fn Parameters", "zwif_snd_switch_tone_play_set Parameters" -Added error code ZW_ERR_SEND_PENDING and changed definition of ZW_ERR_QUEUED -Added tables "if_rec_mcmd_t Structure" -Change section "zwif_group_add" to support assign return route completion callback. -Added multicast APIs: zwif_level_mset, zwif_level_mstart, zwif_level_mstop, zwif_dlck_op_mset, zwif_barrier_mset, zwif_wincvr_mset, zwif_wincvr_mstart, zwif_wincvr_mstop -Modified section "zwif_battery_rpt_set" -Added section "zwif_fw_updt_actv" -Control Firmware Update MD CC v7, Battery CC v2, Sound Switch CC v2, Door Lock Logging CC v1 -Updated "Home network file format"->"interfaces"->"Sound switch CC" -Updated "Home network file format"->"interfaces"->"Battery CC" -Added "Home network file format"->"interfaces"->"Door Lock Logging CC" -Added entry "send_encap" in "Device Database File Format"
10	20191128	TWC	ALL	-v10.03 -Removed entries support_multi_clients and wakeup_no_more_info_delay in Device Database Global Setting -Added state-numbers in Protection CC cache JSON object -Used Z/IP gateway mailbox: internal command queue related APIs (zwnode_cmd_q_xxx)are removed. -Support multi-cast -Allow the zwif_prot_tmout_set API to accept parameter "tmout" with value zero -Added usr_code_len to zwdlck_log_t structure -Support COMMAND_CLASS_NETWORK_MANAGEMENT_PROXY version 3
	20191205	SNA		Added techpub edits

Table of Contents

1 INTRODUCTION 1

1.1 Purpose 1
1.2 Audience and Prerequisites 1

2 ERROR CODES 2

3 PORTAL API 5

3.1 zwportal_init 5
3.2 zwportal_shutdown 6
3.3 zwportal_exit 6
3.4 zwportal_clnt_conn_close 7
3.5 zwportal_clnt_add 7
3.6 zwportal_clnt_rm 7
3.7 zwportal_clnt_find 7
3.8 zwportal_clnt_list_free 8
3.9 zwportal_clnt_list_get 8

4 DEVICE DATABASE API 9

4.1 zwdev_cfg_load 9
4.2 zwdev_cfg_free 29
4.3 zwdev_global_sett_free 29
4.4 zwdev_cfg_find 29

5 NETWORK API 30

5.1 Discovering ZIPGWs 30
5.1.1 zwnet_gw_discvr_start 30
5.1.2 zwnet_gw_discvr_stop 31
5.2 Network Initialization and Clean up 31
5.2.1 zwnet_init 31
5.2.2 zwnet_exit 39
5.2.3 zwnet_reset 39
5.3 Network Creation 39
5.3.1 zwnet_add 39
5.3.2 Secure Inclusion 41
5.3.2.1 zwnet_add_sec2_accept 41
5.3.2.2 zwnet_add_sec2_grant_key 41
5.3.3 SmartStart Provisioning 42
5.3.3.1 zwnet_pl_add 42
5.3.3.2 zwnet_pl_get 45
5.3.3.3 zwnet_pl_del 46
5.3.3.4 zwnet_pl_list_get 46
5.3.3.5 zwnet_pl_list_del 46

5.4 Network Management 47
5.4.1 zwnet_initiate 47
5.4.2 zwnet_fail 47
5.4.3 zwnet_update 47
5.4.4 zwnet_abort 48
5.5 Network Attributes and Traversal 48
5.5.1 zwnet_get_desc 48
5.5.2 zwnet_version 49
5.5.3 zwnet_get_node 49
5.5.4 zwnet_get_node_ by_id 49
5.5.5 zwnet_get_ep_ by_id 50
5.5.6 zwnet_get_if_ by_id 50
5.5.7 zwnet_all_node_sts_get 50
5.5.8 zwnet_node_sts_get 50
5.6 Advanced Network APIs 51
5.6.1 zwnet_migrate 51
5.6.2 zwnet_initiate_classic 51
5.6.3 zwnet_health_chk 51
5.6.4 zwnet_identify 51
5.6.5 zwnet_get_user 52
5.6.6 zwnet_send_nif 52
5.6.7 zwnet_poll_rm 52
5.6.8 zwnet_poll_rm_mul 52
5.6.9 zwnet_pref_set 53
5.6.10 zwnet_pref_get 53
5.6.11 zwnet_client_pref_set 53
5.6.12 zwnet_client_pref_get 53
5.6.13 zwnet_sec2_get_dsk 54
5.7 Network Utilities APIs 54
5.7.1 zwnet_ip_aton 54
5.7.2 zwnet_ip_ntoa 54
5.7.3 zwnet_local_addr_get 55
5.7.4 zwnet_listen_port_get 55

6 NODE API 56

6.1 zwnoded_t 56
6.2 zwnode_get_net 57
6.3 zwnode_get_next 58
6.4 zwnode_get_ep 58
6.5 zwnode_update 58
6.6 zwnode_identify 58
6.7 zwnode_get_ext_ver 59
6.8 Advanced Node APIs 59
6.8.1 zwnode_mul_cmd_ctl_set 59
6.8.2 zwnode_mul_cmd_ctl_get 59

7 ENDPOINT API 60

7.1 zwepd_t 60
7.2 zwep_get_node 60
7.3 zwep_get_next 61
7.4 zwep_get_if 61
7.5 zwep_nameloc_set 61

8 INTERFACE API 62

8.1 zwifd_t 62
8.2 zwif_get_ep 63
8.3 zwif_get_next 63
8.4 zwif_exec 63
8.5 zwif_xxx_poll 63

9 MANAGEMENT CCS BASED INTERFACES 65

9.1 Group Interface API 65
9.1.1 zwif_group_sup_get 65
9.1.2 zwif_group_actv_get 65
9.1.3 zwif_group_get 66
9.1.4 zwif_group_add 66
9.1.5 zwif_group_del 67
9.1.6 zwif_group_info_get 67
9.1.7 zwif_group_info_free 68
9.2 Group Command Interface API 69
9.2.1 zwif_group_cmd_sup_get 69
9.2.2 zwif_group_cmd_get 69
9.2.3 zwif_group_cmd_set 70
9.3 Battery Interface API 70
9.3.1 zwif_battery_rpt_set 70
9.3.2 zwif_battery_get 71
9.3.3 zwif_battery_health_rpt_set 72
9.3.4 zwif_battery_health_get 72
9.4 Time and Date Interface API 73
9.4.1 zwif_time_rpt_set 73
9.4.2 zwif_time_get 73
9.4.3 zwif_date_rpt_set 73
9.4.4 zwif_date_get 74
9.4.5 zwif_tz_dst_rpt_set 74
9.4.6 zwif_tz_dst_get 75
9.5 Firmware Update Interface API 75
9.5.1 zwif_fw_info_get 75
9.5.2 zwif_fw_updt_req 76
9.5.3 zwif_fw_updt_actv 79
9.5.4 zwif_fw_downld_req 79
9.6 Indicator Interface API 81

9.6.1 zwif_ind_rpt_set 81
9.6.2 zwif_ind_get 85
9.6.3 zwif_ind_set 85
9.6.4 zwif_ind_sup_get 85
9.6.5 zwif_ind_sup_cache_get 86
9.6.6 zwif_ind_sup_free 86
9.7 Wakeup Interface API 86
9.7.1 zwif_wakeup_get 87
9.7.2 zwif_wakeup_set 87
9.8 Status Interface API 87
9.8.1 zwif_appl_busy_rpt_set 88
9.8.2 zwif_appl_reject_rpt_set 88

10 NETWORK CC BASED INTERFACES 89

10.1 Z/IP Gateway Interface API 89
10.1.1 zwif_gw_mode_set 89
10.1.2 zwif_gw_mode_get 90
10.1.3 zwif_gw_cfg_lock 90
10.1.4 zwif_gw_unsolicit_set 90
10.1.5 zwif_gw_unsolicit_get 91
10.2 Z/IP Portal Interface API 91
10.2.1 zwif_gw_cfg_set 91
10.2.2 zwif_gw_cfg_get 92
10.3 Power Level Interface API 92
10.3.1 zwif_power_level_rpt_set 92
10.3.2 zwif_power_level_get 93
10.3.3 zwif_power_level_set 93
10.3.4 zwif_power_level_test_rpt_set 93
10.3.5 zwif_power_level_test_get 94
10.3.6 zwif_power_level_test_set 94

11 APPLICATION CC BASED INTERFACES 95

11.1 Basic Interface API 95
11.1.1 zwif_basic_rpt_set 95
11.1.2 zwif_basic_get 96
11.1.3 zwif_basic_set 96
11.2 Switch Interface API 96
11.2.1 zwif_switch_rpt_set 96
11.2.2 zwif_switch_get 97
11.2.3 zwif_switch_set 97
11.2.4 zwif_switch_mset 98
11.3 Level Interface API 98
11.3.1 zwif_level_rpt_set 98
11.3.2 zwif_level_sup_get 99
11.3.3 zwif_level_sup_cache_get 100
11.3.4 zwif_level_get 100

11.3.5 zwif_level_set....100
11.3.6 zwif_level_mset....101
11.3.7 zwif_level_start....101
11.3.8 zwif_level_mstart....102
11.3.9 zwif_level_stop....102
11.3.10 zwif_level_mstop....102
11.4 Color Switch Interface API....102
11.4.1 zwif_color_sw_rpt_set....103
11.4.2 zwif_color_sw_get....103
11.4.3 zwif_color_sw_set....104
11.4.4 zwif_color_sw_start....104
11.4.5 zwif_color_sw_stop....105
11.4.6 zwif_color_sw_sup_get....105
11.4.7 zwif_color_sw_sup_cache_get....105
11.5 Window Covering Interface API....106
11.5.1 zwif_wincvr_rpt_set....106
11.5.2 zwif_wincvr_get....108
11.5.3 zwif_wincvr_set....108
11.5.4 zwif_wincvr_mset....108
11.5.5 zwif_wincvr_start....109
11.5.6 zwif_wincvr_mstart....109
11.5.7 zwif_wincvr_stop....110
11.5.8 zwif_wincvr_mstop....110
11.5.9 zwif_wincvr_sup_get....110
11.5.10 zwif_wincvr_sup_cache_get....111
11.6 Barrier Operator Interface API....111
11.6.1 zwif_barrier_rpt_set....111
11.6.2 zwif_barrier_get....112
11.6.3 zwif_barrier_set....112
11.6.4 zwif_barrier_mset....112
11.6.5 zwif_barrier_notif_sup_get....112
11.6.6 zwif_barrier_notif_sup_cache_get....113
11.6.7 zwif_barrier_notif_rpt_set....113
11.6.8 zwif_barrier_notif_cfg_get....114
11.6.9 zwif_barrier_notif_cfg_set....114
11.7 Sound Switch Interface API....114
11.7.1 zwif_snd_switch_rpt_set....115
11.7.2 zwif_snd_switch_config_get....115
11.7.3 zwif_snd_switch_config_set....116
11.7.4 zwif_snd_switch_tone_play_get....116
11.7.5 zwif_snd_switch_tone_play_set....116
11.7.6 zwif_snd_switch_tone_info_get....117
11.8 Binary Sensor Interface API....118
11.8.1 zwif_bsensor_rpt_set....118
11.8.2 zwif_bsensor_get....118

11.8.3 zwif_bsensor_sup_get 118
11.8.4 zwif_bsensor_sup_cache_get 119
11.9 Alarm/Notification Interface API 119
11.9.1 zwif_alrm_rpt_set 119
11.9.2 zwif_alrm_get 130
11.9.3 zwif_alrm_set 130
11.9.4 zwif_alrm_sup_get 131
11.9.5 zwif_alrm_sup_cache_get 131
11.9.6 zwif_alrm_sup_evt_get 131
11.9.7 zwif_alrm_sup_evt_cache_get 132
11.9.8 zwif_alrm_vtype_sup_get 132
11.10 Alarm Sensor Interface API 132
11.10.1 zwif_alrm_snsr_rpt_set 132
11.10.2 zwif_alrm_snsr_get 133
11.10.3 zwif_alrm_snsr_sup_get 134
11.10.4 zwif_alrm_snsr_sup_cache_get 134
11.11 Sensor Interface API 134
11.11.1 zwif_sensor_rpt_set 134
11.11.2 zwif_sensor_get 138
11.11.3 zwif_sensor_sup_get 138
11.11.4 zwif_sensor_unit_get 138
11.11.5 zwif_sensor_unit_cache_get 139
11.11.6 zwif_sensor_sup_cache_get 139
11.12 Central Scene Interface API 140
11.12.1 zwif_csc_rpt_set 140
11.12.2 zwif_csc_sup_get 141
11.12.3 zwif_csc_cfg_rpt_set 142
11.12.4 zwif_csc_cfg_get 143
11.12.5 zwif_csc_cfg_set 143
11.13 Pulse Meter Interface API 143
11.13.1 zwif_pulsemeter_rpt_set 143
11.13.2 zwif_pulsemeter_get 144
11.14 Meter Interface API 144
11.14.1 zwif_meter_rpt_set 144
11.14.2 zwif_meter_get 145
11.14.3 zwif_meter_sup_get 146
11.14.4 zwif_meter_sup_cache_get 147
11.14.5 zwif_meter_reset 147
11.14.6 zwif_meter_set_admin 147
11.14.7 zwif_meter_get_desc 148
11.15 Door Lock Interface API 148
11.15.1 zwif_dlck_op_rpt_set 148
11.15.2 zwif_dlck_op_get 150
11.15.3 zwif_dlck_op_set 150
11.15.4 zwif_dlck_op_mset 150

11.15.5 zwif_dlck_cfg_set 151
11.15.6 zwif_dlck_cfg_get 151
11.15.7 zwif_dlck_cap_get 152
11.15.8 zwif_dlck_cap_cache_get 153
11.15.9 zwif_dlck_cap_free 153
11.16 Door Lock Logging Interface API 153
11.16.1 zwif_lcklog_rpt_set 153
11.16.2 zwif_lcklog_get 155
11.16.3 zwif_lcklog_sup_get 155
11.16.4 zwif_lcklog_sup_cache_get 156
11.17 User Code Interface API 156
11.17.1 zwif_usrcod_rpt_set 156
11.17.2 zwif_usrcod_get 157
11.17.3 zwif_usrcod_set 157
11.17.4 zwif_usrcod_sup_get 157
11.17.5 zwif_usrcod_sup_cache_get 158
11.17.6 zwif_usrcod_ext_rpt_set 158
11.17.7 zwif_usrcod_ext_get 159
11.17.8 zwif_usrcod_ext_set 159
11.17.9 zwif_usrcod_cap_get 160
11.17.10 zwif_usrcod_cap_cache_get 160
11.17.11 zwif_usrcod_kp_mod_rpt_set 161
11.17.12 zwif_usrcod_kp_mod_get 162
11.17.13 zwif_usrcod_kp_mod_set 162
11.17.14 zwif_usrcod_ms_cod_rpt_set 162
11.17.15 zwif_usrcod_ms_cod_get 162
11.17.16 zwif_usrcod_ms_cod_set 163
11.17.17 zwif_usrcod_chksum_rpt_set 163
11.17.18 zwif_usrcod_chksum_get 163
11.18 Thermostat Fan Mode Interface API 163
11.18.1 zwif_thrmo_fan_md_rpt_set 164
11.18.2 zwif_thrmo_fan_md_get 165
11.18.3 zwif_thrmo_fan_md_set 165
11.18.4 zwif_thrmo_fan_md_sup_get 165
11.18.5 zwif_thrmo_fan_md_sup_cache_get 166
11.19 Thermostat Fan State Interface API 166
11.19.1 zwif_thrmo_fan_sta_rpt_set 166
11.19.2 zwif_thrmo_fan_sta_get 167
11.20 Thermostat Mode Interface API 167
11.20.1 zwif_thrmo_md_rpt_set 167
11.20.2 zwif_thrmo_md_get 168
11.20.3 zwif_thrmo_md_set 169
11.20.4 zwif_thrmo_md_sup_get 169
11.20.5 zwif_thrmo_md_sup_cache_get 169
11.21 Thermostat Operating State Interface API 170

11.21.1 zwif_thrmo_op_sta_rpt_set....170
11.21.2 zwif_thrmo_op_sta_get....171
11.21.3 zwif_thrmo_op_sta_log_sup_get....171
11.21.4 zwif_thrmo_op_sta_log_sup_cache_get....171
11.21.5 zwif_thrmo_op_sta_log_rpt_set....172
11.21.6 zwif_thrmo_op_sta_log_get....172
11.22 Thermostat Setback Interface API....173
11.22.1 zwif_thrmo_setb_rpt_set....173
11.22.2 zwif_thrmo_setb_get....173
11.22.3 zwif_thrmo_setb_set....174
11.23 Thermostat Setpoint Interface API....174
11.23.1 zwif_thrmo_setp_rpt_set....174
11.23.2 zwif_thrmo_setp_get....175
11.23.3 zwif_thrmo_setp_set....175
11.23.4 zwif_thrmo_setp_sup_get....176
11.23.5 zwif_thrmo_setp_sup_cache_get....176
11.23.6 zwif_thrmo_setp_sup_range_get....176
11.23.7 zwif_thrmo_setp_sup_range_cache_get....177
11.24 Configuration Interface API....177
11.24.1 zwif_config_rpt_set....177
11.24.2 zwif_config_get....178
11.24.3 zwif_config_set....178
11.24.4 zwif_config_bulk_rpt_set....178
11.24.5 zwif_config_bulk_get....179
11.24.6 zwif_config_bulk_set....180
11.24.7 zwif_config_prm_reset....180
11.24.8 zwif_config_info_get....180
11.24.9 zwif_config_info_free....181
11.25 Clock Interface API....182
11.25.1 zwif_clock_rpt_set....182
11.25.2 zwif_clock_get....182
11.25.3 zwif_clock_set....183
11.26 Climate Control Schedule Interface API....183
11.26.1 zwif_clmt_ctl_schd_rpt_set....183
11.26.2 zwif_clmt_ctl_schd_get....184
11.26.3 zwif_clmt_ctl_schd_set....184
11.26.4 zwif_clmt_ctl_schd_chg_rpt_set....184
11.26.5 zwif_clmt_ctl_schd_chg_get....185
11.26.6 zwif_clmt_ctl_schd_ovr_rpt_set....185
11.26.7 zwif_clmt_ctl_schd_ovr_get....185
11.26.8 zwif_clmt_ctl_schd_ovr_set....186
11.27 AV Interface API....186
11.27.1 zwif_av_set....186
11.27.2 zwif_av_caps....186
11.28 Protection Interface API....187

11.28.1 zwif_prot_rpt_set....187
11.28.2 zwif_prot_get....188
11.28.3 zwif_prot_set....188
11.28.4 zwif_prot_sup_get....188
11.28.5 zwif_prot_sup_cache_get....189
11.28.6 zwif_prot_ec_rpt_set....189
11.28.7 zwif_prot_ec_get....190
11.28.8 zwif_prot_ec_set....190
11.28.9 zwif_prot_tmout_rpt_set....190
11.28.10 zwif_prot_tmout_get....191
11.28.11 zwif_prot_tmout_set....191

12 DEVICE DATABASE FILE FORMAT....192

13 HOME NETWORK FILE FORMAT....201

13.1 Introduction....201
13.2 Top Level Entities....201
13.3 Network....201
13.4 Nodes....202
13.5 Endpoints....203
13.6 Interfaces....204
13.6.1 Association Group Info CC....204
13.6.2 Configuration CC....205
13.6.3 Association and Multi Channel Association CC....205
13.6.4 Central Scene CC....206
13.6.5 Multi Level Sensor CC....206
13.6.6 Indicator CC....207
13.6.7 Thermostat Fan Mode CC....208
13.6.8 Thermostat Mode CC....208
13.6.9 Thermostat Setpoint CC....209
13.6.10 Thermostat Operating State CC....210
13.6.11 Thermostat Fan State CC....210
13.6.12 Multi Level Switch CC....211
13.6.13 Binary Switch CC....211
13.6.14 Binary Sensor CC....212
13.6.15 Simple AV Control CC....212
13.6.16 Alarm or Notification CC....212
13.6.17 Protection CC....213
13.6.18 User Code CC....214
13.6.19 Meter CC....215
13.6.20 Meter Table Monitor CC....216
13.6.21 Door Lock CC....216
13.6.22 Door Lock Logging CC....218
13.6.23 Alarm Sensor CC....218
13.6.24 Barrier Operator CC....219
13.6.25 Color Switch CC....219

13.6.26 Sound Switch CC....220
13.6.27 Window Covering CC....221
13.6.28 Battery CC....221
13.6.29 Basic CC....222

REFERENCES....224

Table of Tables

Table 1 – Error Codes, ZW_ERR_XXX....2
Table 2 – Low-Level Library Error Codes....3
Table 3 – zwportal_init Parameters....5
Table 4 – zwportal_init_t Structure....5
Table 5 – clnt_prof_t Structure....5
Table 6 – zwportal_cb_t Parameters....6
Table 7 – zwportal_shutdown Parameters....6
Table 8 – zwportal_exit Parameters....6
Table 9 – zwportal_clnt_conn_close Parameters....7
Table 10 – zwportal_clnt_add Parameters....7
Table 11 – zwportal_clnt_rm Parameters....7
Table 12 – zwportal_clnt_find Parameters....7
Table 13 – zwportal_clnt_list_free Parameters....8
Table 14 – clnt_prof_lst structure....8
Table 15 – zwportal_clnt_list_get Parameters....8
Table 16 – zwdev_cfg_load Parameters....9
Table 17 – dev_rec_t Structure....9
Table 18 – DEV_XXX Meaning and Values....9
Table 19 – ep_rec_t Structure....10
Table 20 – redir_rec_t Structure....10
Table 21 – if_rec_t Structure....10
Table 22 – Interface Types: IF_REC_TYPE_XXX....11
Table 23 – rec Union....11
Table 24 – if_rec_grp_t Structure....11
Table 25 – if_rec_config_t Structure....12
Table 26 – if_rec_cfg_set_t Structure....12
Table 27 – if_rec_cfg_info_t Structure....12
Table 28 – if_rec_bsnsr_t Structure....12
Table 29 – if_rec_bsnsr_match_t Structure....13
Table 30 – Table 31 – if_rec_snsr_t Structure....13
Table 32 – if_rec_meter_t Structure....13
Table 33 – if_rec_alarm_match_t Structure....13
Table 34 – if_rec_alarm_result_t Structure....13
Table 35 – if_rec_alarm_rev_match_t Structure....14

Table 36 – if_rec_basic_match_t Structure 14
Table 37 – if_rec_alrm_snsr_match_t Structure 14
Table 38 – zwsetp_temp_range_t Structure 15
Table 39 – if_rec_dlck_t Structure 15
Table 40 – if_rec_mcmd_t Structure 15
Table 41 – Binary Sensor Types: ZW_BSENSOR_TYPE_XXX 15
Table 42 – Multi-Level Sensor Type: ZW_SENSOR_TYPE_XXX 16
Table 43 – Multi-Level Sensor Unit 17
Table 44 – Meter Type 20
Table 45 – Meter Supported Unit Bit-Mask 20
Table 46 – Z-Wave Alarm/Notification Type 21
Table 47 – Z-Wave Alarm/Notification Event 21
Table 48 – Z-Wave Alarm/Notification Event Parameter Type 27
Table 49 – Alarm Sensor Type 27
Table 50 – Thermostat Setpoint Types: ZW_THRMO_SETP_TYP_XXX 27
Table 51 – Thermostat Setpoint Unit 28
Table 52 – Door Lock Operation Mode 28
Table 53 – Door lock Operation Type 28
Table 54 – dev_cfg_error_t Structure 28
Table 55 – zwdev_cfg_free Parameters 29
Table 56 – zwdev_global_sett_free Parameters 29
Table 57 – zwdev_cfg_find Parameters 29
Table 58 – dev_rec_srch_key_t Structure 29
Table 59 – zwnet_gw_discvr_start Parameters 30
Table 60 – zwnet_gw_discvr_cb_t Parameters 30
Table 61 – zwnet_gw_discvr_stop Parameters 31
Table 62 – zwnet_init Parameters 31
Table 63 – zwnet_init_t Structure 31
Table 64 – dev_cfg_usr_t Structure 33
Table 65 – zwnet_dev_rec_find_fn Parameters 33
Table 66 – dev_global_sett_t Structure 33
Table 67 – zwnet_xxx Generic Parameters 34
Table 68 – zwnet_unhandled_cmd_fn Parameters 34
Table 69 – sup_cmd_cls_t Structure 34
Table 70 – zwnet_notify_fn Parameters 34
Table 71 – zwnet_sts_t Structure 36
Table 72 – NW_CHG_UPDT_XXX 36
Table 73 – zw_health_prg_t Structure 37
Table 74 – zw_health_rpt_t Structure 37
Table 75 – zw_health_sts_t Structure 37
Table 76 – Additional Information Associated to Network Op and Status 37
Table 77 – zwnet_node_fn Parameters 38
Table 78 – zwnet_appl_fn Parameters 38
Table 79 – pl_info_fn Parameters 38
Table 80 – zwnet_net_err_fn Parameters 38

Table 81 – print_fn Parameter 39
Table 82 – zwnet_dev_rec_find_fn Parameter 39
Table 83 – zwnet_exit Parameters 39
Table 84 – zwnet_add Parameters 39
Table 85 – sec2_add_prm_t Structure 40
Table 86 – add_node_sec2_fn Parameters 40
Table 87 – sec2_add_cb_prm_t Structure....40
Table 88 – sec2_keys_req_cb_prm_t Structure 40
Table 89 – sec2_dsk_cb_prm_t Structure 41
Table 90 – zwnet_add_sec2_accept Parameters 41
Table 91 – zwnet_add_sec2_grant_key Parameters....41
Table 92 – zwnet_pl_add Parameters 42
Table 93 – pl_info_t Structure 42
Table 94 – pl_prod_type_t Structure 44
Table 95 – pl_prod_id_t Structure....44
Table 96 – pl_uuid_t Structure 44
Table 97 – pl_nw_sts_t Structure 45
Table 98 – zwnet_pl_get Parameters 45
Table 99 – pl_info_fn Parameters 45
Table 100 – pl_lst_ent_t Structure 45
Table 101 – zwnet_pl_del Parameters 46
Table 102 – zwnet_pl_list_get Parameters 46
Table 103 – pl_list_fn Parameters 46
Table 104 – zwnet_pl_list_del Parameters....46
Table 105 – zwnet_initiate Parameters....47
Table 106 – zwnet_fail Parameters 47
Table 107 – zwnet_get_desc Parameters....48
Table 108 – zwnetd_t Structure 48
Table 109 – zwnet_version Parameters 49
Table 110 – zwnet_get_node Parameters....49
Table 111 – zwnet_get_node_by_id Parameters 49
Table 112 – zwnet_get_ep_by_id Parameters 50
Table 113 – zwnet_get_if_by_id Parameters 50
Table 114 – zwnet_all_node_sts_get Parameters....50
Table 115 – zwnet_node_sts_get Parameters 50
Table 116 – zwnet_initiate_classic Parameters....51
Table 117 – zwnet_get_user Parameters 52
Table 118 – zwnet_send_nif Parameters 52
Table 119 – zwnet_poll_rm Parameters....52
Table 120 – zwnet_poll_rm_mul Parameters 52
Table 121 – zwnet_pref_set Parameters....53
Table 122 – zwnet_pref_get Parameters 53
Table 123 – zwnet_client_pref_set Parameters....53
Table 124 – zwnet_client_pref_get Parameters....53
Table 125 – zwnet_sec2_get_dsk Parameters 54

Table 126 – get_dsk_fn Parameters 54
Table 127 – zwnet_ip_aton Parameters 54
Table 128 – zwnet_ip_ntoa Parameters 54
Table 129 – zwnet_local_addr_get Parameters 55
Table 130 – zwnet_listen_port_get Parameters 55
Table 131 – zwnoded_t Structure 56
Table 132 – NODE_PROPTY_XXX 56
Table 133 – dev_id_t Structure 57
Table 134 – zwsw_ver_t Structure 57
Table 135 – zwnode_get_net Parameters 57
Table 136 – zwnode_get_next Parameters 58
Table 137 – zwnode_get_ep Parameters 58
Table 138 – zwnode_update Parameters 58
Table 139 – zwnode_identify Parameters 58
Table 140 – zwnode_get_ext_ver Parameters 59
Table 141 – ext_ver_t Structure 59
Table 142 – zwnode_mul_cmd_ctl_set Parameters 59
Table 143 – zwnode_mul_cmd_ctl_get Parameters 59
Table 144 – zwepd_t Structure 60
Table 145 – zwplus_info_t structure 60
Table 146 – zwep_get_node Parameters 60
Table 147 – zwep_get_next Parameters 61
Table 148 – zwep_get_if Parameters 61
Table 149 – zwep_nameloc_set Parameters 61
Table 150 – zw_nameloc_t Structure 61
Table 151 – zwifd_t Structure 62
Table 152 – zwif_get_ep Parameters 63
Table 153 – zwif_get_next Parameters 63
Table 154 – zwif_exec Parameters 63
Table 155 – zwif_xxx_poll Parameters 63
Table 156 – zwpoll_req_t Structure 64
Table 157 – zwpoll_cmplt_fn Parameters 64
Table 158 – zwif_group_sup_get Parameters 65
Table 159 – zwrep_group_sup_fn Parameters 65
Table 160 – zwif_group_actv_get Parameters 65
Table 161 – zwrep_group_actv_fn Parameters 65
Table 162 – zwif_group_get Parameters 66
Table 163 – zwrep_group_fn Parameters 66
Table 164 – grp_member_t structure 66
Table 165 – zwif_group_add Parameters 66
Table 166 – zwif_grp_rr_fn Parameters 67
Table 167 – RRA_XXX 67
Table 168 – zwif_group_del Parameters 67
Table 169 – zwif_group_info_get Parameters 68
Table 170 – if_grp_info_dat_t Structure 68

Table 171 – zw_grp_info_t Structure 68
Table 172 – grp_cmd_ent_t Structure....68
Table 173 – zwif_group_info_free Parameters 68
Table 174 – zwif_group_cmd_sup_get Parameters 69
Table 175 – zwrep_grp_cmd_sup_fn Parameters....69
Table 176 – zwgrp_cmd_cap_t structure 69
Table 177 – zwif_group_cmd_get Parameters....69
Table 178 – zwrep_grp_cmd_fn Parameters....69
Table 179 – zwif_group_cmd_set Parameters 70
Table 180 – zwif_battery_rpt_set Parameters 70
Table 181 – zwrep_batt_lvl_fn Parameters....70
Table 182 – zwbatt_dat_t Structure....71
Table 183 – BATT_STS_XXX 71
Table 184 – BATT_RECHG_XXX....71
Table 185 – zwif_battery_get Parameters 71
Table 186 – zwif_battery_health_rpt_set Parameters....72
Table 187 – zwrep_batt_health_fn Parameters....72
Table 188 – zwbatt_health_t Structure....72
Table 189 – zwif_battery_health_get Parameters 72
Table 190 – zwif_time_rpt_set Parameters 73
Table 191 – zwrep_time_fn Parameters 73
Table 192 – zwif_time_get Parameters....73
Table 193 – zwif_date_rpt_set Parameters....73
Table 194 – zwrep_date_fn Parameters....74
Table 195 – zwif_date_get Parameters....74
Table 196 – zwif_tz_dst_rpt_set Parameters 74
Table 197 – zwrep_tz_dst_fn Parameters 74
Table 198 – tmzone_info_t Structure....74
Table 199 – dst_info_t Structure....74
Table 200 – zwif_tz_dst_get Parameters 75
Table 201 – zwif_fw_info_get Parameters....75
Table 202 – zwrep_fw_info_fn Parameters....75
Table 203 – zwfw_info_t structure....76
Table 204 – zwif_fw_updt_req Parameters 76
Table 205 – zwfw_updt_req_t Structure....76
Table 206 – zwrep_fw_updt_sts_fn Parameters....77
Table 207 – ZW_FW_UPDT_ERR_XXX 77
Table 208 – zwrep_fw_updt_cmplt_fn Parameters 77
Table 209 – ZW_FW_UPDT_CMPLT_XXX 78
Table 210 – zwrep_fw_tgt_restart_fn Parameters 78
Table 211 – ZW_FW_UPDT_RESTART_XXX 78
Table 212 – zwif_fw_updt_actv Parameters 79
Table 213 – zwrep_fw_actv_fn Parameters 79
Table 214 – zwfw_actv_sts_t Structure....79
Table 215 – ZW_FW_ACTV_STS_XXX 79

Table 216 – zwif_fw_downld_req Parameters 79
Table 217 – zwfw_downld_req_t Structure 80
Table 218 – zwfw_downld_sts_fn Parameters 80
Table 219 – ZW_FW_DL_RQ_XXX 80
Table 220 – zwfw_downld_cmplt_fn Parameters 80
Table 221 – ZW_FW_DL_CMPLT_XXX 81
Table 222 – zwif_ind_rpt_set Parameters 81
Table 223 – zwrep_ind_fn Parameters 81
Table 224 – zwind_data_t Structure 82
Table 225 – zwind_propty_val_t Structure 82
Table 226 – ZWIND_ID_XXX 82
Table 227 – ZWIND_PPTY_ID_XXX 84
Table 228 – zwif_ind_get Parameters 85
Table 229 – zwif_ind_set Parameters 85
Table 230 – zwif_ind_sup_get Parameters 85
Table 231 – zwrep_ind_sup_fn Parameters 86
Table 232 – zwind_sup_t Structure 86
Table 233 – zwif_ind_sup_cache_get Parameters 86
Table 234 – zwif_ind_sup_free Parameters 86
Table 235 – zwif_wakeup_get Parameters 87
Table 236 – zwrep_wakeup_fn Parameters 87
Table 237 – zwif_wakeup_t Structure 87
Table 238 – zwif_wakeup_set Parameters 87
Table 239 – zwif_appl_busy_rpt_set Parameters 88
Table 240 – zwrep_appl_busy_fn Parameters 88
Table 241 – ZW_BSY_STS_XXX 88
Table 242 – zwif_appl_reject_rpt_set Parameters 88
Table 243 – zwrep_appl_reject_fn Parameters 88
Table 244 – ZW_RJ_STS_XXX 88
Table 245 – zwif_gw_mode_set Parameters 89
Table 246 – zwgw_portal_prof_t structure 89
Table 247 – zwif_gw_mode_get Parameters 90
Table 248 – zwrep_gw_mode_fn Parameters 90
Table 249 – zwif_gw_mode_get Parameters 90
Table 250 – zwif_gw_unsolicit_set Parameters 90
Table 251 – zwif_gw_unsolicit_get Parameters 91
Table 252 – zwrep_gw_unsolicit_fn Parameters 91
Table 253 – zwif_gw_cfg_set Parameters 91
Table 254 – zwportal_cfg_t Structure 91
Table 255 – zwrep_cfg_sts_fn Parameters 92
Table 256 – zwif_gw_mode_get Parameters 92
Table 257 – zwrep_gw_cfg_fn Parameters 92
Table 258 – zwif_power_level_rpt_set Parameters 92
Table 259 – zwrep_power_level_fn Parameters 93
Table 260 – zwif_power_level_get Parameters 93

Table 261 – zwif_power_level_set Parameters....93
Table 262 – zwif_power_level_test_rpt_set Parameters....93
Table 263 – zwrep_power_level_test_fn Parameters....93
Table 264 – POWERLEVEL_TEST_XXX....94
Table 265 – zwif_power_level_test_get Parameters....94
Table 266 – zwif_power_level_test_set Parameters....94
Table 267 – zwif_basic_rpt_set Parameters....95
Table 268 – zwrep_basic_fn Parameters....95
Table 269 – zwbasic_t Structure....95
Table 270 – zwif_basic_get Parameters....96
Table 271 – zwif_basic_set Parameters....96
Table 272 – zwif_switch_rpt_set Parameters....96
Table 273 – zwrep_switch_fn Parameters....96
Table 274 – zwswitch_t Structure....97
Table 275 – zwif_switch_get Parameters....97
Table 276 – ZWIF_GET_BMSK_XXX....97
Table 277 – zwif_switch_set Parameters....97
Table 278 – zwif_switch_mset Parameters....98
Table 279 – zw_postset_fn Parameters....98
Table 280 – zwif_level_rpt_set Parameters....99
Table 281 – zwrep_level_fn Parameters....99
Table 282 – zwlevel_dat_t structure....99
Table 283 – zwif_level_sup_get Parameters....99
Table 284 – zwrep_lvl_sup_fn Parameters....99
Table 285 – zwif_level_sup_cache_get Parameters....100
Table 286 – zwif_level_get Parameters....100
Table 287 – zwif_level_set Parameters....100
Table 288 – zwif_level_mset Parameters....101
Table 289 – zwif_level_start Parameters....101
Table 290 – zwlevel_t structure....101
Table 291 – zwif_level_mstart Parameters....102
Table 292 – zwif_level_stop Parameters....102
Table 293 – zwif_level_mstop Parameters....102
Table 294 – zwif_color_sw_rpt_set Parameters....103
Table 295 – zwrep_color_sw_get_fn Parameters....103
Table 296 – zwcolor_t Structure....103
Table 297 – COL_SW_COMP_ID_XXX....103
Table 298 – zwif_color_sw_get Parameters....104
Table 299 – zwif_color_sw_set Parameters....104
Table 300 – zwif_color_sw_start Parameters....104
Table 301 – zwcol_ctl_t structure....104
Table 302 – COL_SW_LVL_XXX....105
Table 303 – zwif_color_sw_stop Parameters....105
Table 304 – zwif_color_sw_sup_get Parameters....105
Table 305 – zwrep_color_sw_sup_fn Parameters....105

Table 306 – zwif_color_sw_sup_cache_get Parameters 105
Table 307 – zwif_wincvr_rpt_set Parameters 106
Table 308 – zwrep_wincvr_fn Parameters 106
Table 309 – wincvr_dat_t Structure 106
Table 310 – WIN_COVER_ID_XXX 106
Table 311 – zwif_wincvr_get Parameters 108
Table 312 – zwif_wincvr_set Parameters 108
Table 313 – zwif_wincvr_mset Parameters 109
Table 314 – zwif_wincvr_start Parameters 109
Table 315 – zwif_wincvr_mstart Parameters 109
Table 316 – zwif_wincvr_stop Parameters 110
Table 317 – zwif_wincvr_mstop Parameters 110
Table 318 – zwif_wincvr_sup_get Parameters 110
Table 319 – zwrep_wincvr_sup_fn Parameters 110
Table 320 – zwif_wincvr_sup_cache_get Parameters 111
Table 321 – zwif_barrier_rpt_set Parameters 111
Table 322 – zwrep_barrier_fn Parameters 111
Table 323 – ZW_BAR_STA_XXX 111
Table 324 – zwif_barrier_get Parameters 112
Table 325 – zwif_barrier_set Parameters 112
Table 326 – zwif_barrier_mset Parameters 112
Table 327 – zwif_barrier_notif_sup_get Parameters 113
Table 328 – zwrep_barrier_notif_sup_fn Parameters 113
Table 329 – ZW_BAR_NOTIF_TYP_XXX 113
Table 330 – zwif_barrier_notif_sup_cache_get Parameters 113
Table 331 – zwif_barrier_notif_rpt_set Parameters 113
Table 332 – zwrep_barrier_subsys_fn Parameters 114
Table 333 – zwif_barrier_notif_cfg_get Parameters 114
Table 334 – zwif_barrier_notif_cfg_set Parameters 114
Table 335 – zwif_snd_switch_rpt_set Parameters 115
Table 336 – zwrep_snd_switch_config_fn Parameters 115
Table 337 – zwrep_snd_switch_tone_play_fn Parameters 115
Table 338 – zwif_snd_switch_config_get Parameters 116
Table 339 – zwif_snd_switch_config_set Parameters 116
Table 340 – zwif_snd_switch_tone_play_get Parameters 116
Table 341 – zwif_snd_switch_tone_play_set Parameters 116
Table 342 – zwif_snd_switch_tone_info_get Parameters 117
Table 343 – zwrep_snd_switch_tone_info_fn Parameters 117
Table 344 – zwsnd_switch_tone_info_t Structure 117
Table 345 – if_snd_switch_tone_info_t Structure 117
Table 346 – zwif_bsensor_rpt_set Parameters 118
Table 347 – zwrep_bsensor_fn Parameters 118
Table 348 – zwif_bsensor_get Parameters 118
Table 349 – zwif_bsensor_sup_get Parameters 118
Table 350 – zwrep_bsensor_sup_fn Parameters 119

Table 351 – zwif_bsensor_sup_cache_get Parameters....119
Table 352 – zwif_alrm_rpt_set Parameters....119
Table 353 – zwrep_alrm_fn Parameters....119
Table 354 – zwalrm_t Structure....120
Table 355 – ZW_ALRM_STS_XXX....120
Table 356 – ZW_ALRM_XXX....121
Table 357 – ZW_ALRM_EVT_XXX....121
Table 358 – zwif_alrm_get Parameters....130
Table 359 – zwif_alrm_set Parameters....130
Table 360 – zwif_alrm_sup_get Parameters....131
Table 361 – zwrep_alrm_sup_fn Parameters....131
Table 362 – zwif_alrm_sup_cache_get Parameters....131
Table 363 – zwif_alrm_sup_evt_get Parameters....131
Table 364 – zwrep_alrm_evt_fn Parameters....132
Table 365 – zwif_alrm_sup_evt_cache_get Parameters....132
Table 366 – zwif_alrm_vtype_sup_get Parameters....132
Table 367 – zwif_alrm_snsr_rpt_set Parameters....133
Table 368 – zwrep_alrm_snsr_fn Parameters....133
Table 369 – zw_alrm_snsr_t Structure....133
Table 370 – ZW_ALRM_SNSR_TYPE_XXX....133
Table 371 – zwif_alrm_snsr_get Parameters....133
Table 372 – zwif_alrm_snsr_sup_get Parameters....134
Table 373 – zwrep_alrm_snsr_sup_fn Parameters....134
Table 374 – zwif_alrm_snsr_sup_cache_get Parameters....134
Table 375 – zwif_sensor_rpt_set Parameters....135
Table 376 – zwrep_sensor_fn Parameters....135
Table 377 – zwsensor_t Structure....135
Table 378 – ZW_SENSOR_TYPE_XXX, ZW_SENSOR_UNIT_XXX_YYY....135
Table 379 – zwif_sensor_get Parameters....138
Table 380 – zwif_sensor_sup_get Parameters....138
Table 381 – zwrep_sensor_sup_fn Parameters....138
Table 382 – zwif_sensor_unit_get Parameters....138
Table 383 – zwrep_sensor_unit_fn Parameters....139
Table 384 – zwif_sensor_unit_cache_get Parameters....139
Table 385 – zwif_sensor_sup_cache_get Parameters....139
Table 386 – if_sensor_data_t Structure....139
Table 387 – zwif_csc_rpt_set Parameters....140
Table 388 – zwrep_csc_fn Parameters....140
Table 389 – zwcsc_notif_t Structure....140
Table 390 – ZW_CSC_KEY_ATTRIB_XXX....141
Table 391– zwif_csc_sup_get Parameters....141
Table 392 – zwrep_csc_sup_fn Parameters....141
Table 393– zwif_csc_cfg_rpt_set Parameters....142
Table 394 – zwrep_csc_cfg_fn Parameters....143
Table 395– zwif_csc_cfg_get Parameters....143

Table 396– zwif_csc_cfg_set Parameters....143
Table 397 – zwif_pulsemeter_rpt_set Parameters....143
Table 398 – zwrep_pulsemeter_fn Parameters....144
Table 399 – zwif_pulsemeter_get Parameters....144
Table 400 – zwif_meter_rpt_set Parameters....144
Table 401 – zwrep_meter_fn Parameters....144
Table 402 – zwmeter_dat_t Structure....144
Table 403 – ZW_METER_TYPE_XXX and ZW_METER_UNIT_XXX....145
Table 404 – zwif_meter_get Parameters....145
Table 405 – zwif_meter_sup_get Parameters....146
Table 406 – zwrep_meter_sup_fn Parameters....146
Table 407 – zwmeter_cap_t Structure....146
Table 408 – ZW_METER_SUP_UNIT_XXX....147
Table 409 – zwif_meter_sup_cache_get Parameters....147
Table 410 – zwif_meter_reset Parameters....147
Table 411 – zwif_meter_set_admin Parameters....148
Table 412 – zwif_meter_get_desc Parameters....148
Table 413 – zwrep_meterd_fn Parameters....148
Table 414 – zwmeter_t Structure....148
Table 415 – zwif_dlck_op_rpt_set Parameters....148
Table 416 – zwrep_dlck_op_fn Parameters....149
Table 417 – zwdlck_op_t Structure....149
Table 418 – ZW_DOOR_XXX....149
Table 419 – ZW_COND_XXX_MASK....150
Table 420 – zwif_dlck_op_get Parameters....150
Table 421 – zwif_dlck_op_set Parameters....150
Table 422 – zwif_dlck_op_mset Parameters....150
Table 423 – zwif_dlck_cfg_set Parameters....151
Table 424 – zwdlck_cfg_t Structure....151
Table 425 – zwif_dlck_cfg_get Parameters....151
Table 426 – zwrep_dlck_cfg_fn Parameters....152
Table 427 – zwif_dlck_cap_get Parameters....152
Table 428 – zwrep_dlck_cap_fn Parameters....152
Table 429 – zwdlck_cap_t Structure....152
Table 430 – ZW_DLCK_CAP_XXX_MASK....153
Table 431 – zwif_dlck_cap_cache_get Parameters....153
Table 432 – zwif_dlck_cap_free Parameters....153
Table 433 – zwif_lcklog_rpt_set Parameters....154
Table 434 – zwrep_lcklog_fn Parameters....154
Table 435 – zwdlck_log_t Structure....154
Table 436 – ZWLCK_EVT_XXX....154
Table 437 – zwif_lcklog_get Parameters....155
Table 438 – zwif_lcklog_sup_get Parameters....155
Table 439 – zwrep_lcklog_cap_fn Parameters....156
Table 440 – zwif_lcklog_sup_cache_get Parameters....156

Table 441 – zwif_usrcod_rpt_set Parameters 156
Table 442 – zwrep_usr_cod_fn Parameters 156
Table 443 – zwusrcod_t structure 156
Table 444 – ZW_USRCOD_XXX 157
Table 445 – zwif_usrcod_get Parameters 157
Table 446 – zwif_usrcod_set Parameters 157
Table 447 – zwif_usrcod_sup_get Parameters 158
Table 448 – zwrep_usr_sup_fn Parameters 158
Table 449 – zwif_usrcod_sup_cache_get Parameters 158
Table 450 – zwif_usrcod_ext_rpt_set Parameters 158
Table 451 – zwrep_usr_cod_ext_fn Parameters 158
Table 452 – zwusrcod_ext_t structure 159
Table 453 – zwif_usrcod_ext_get Parameters 159
Table 454 – zwif_usrcod_ext_set Parameters 159
Table 455 – zwif_usrcod_cap_get Parameters 160
Table 456 – zwrep_usr_cap_fn Parameters 160
Table 457 – zwusrcod_cap_t structure 160
Table 458 – ZW_USRCOD_CAP_XXX Bitmask 160
Table 459 – zwif_usrcod_cap_cache_get Parameters 161
Table 460 – zwif_usrcod_kp_mod_rpt_set Parameters 161
Table 461 – zwrep_usr_kpmod_fn Parameters 161
Table 462 – ZW_KEYPAD_MOD_XXX 161
Table 463 – zwif_usrcod_kp_mod_get Parameters 162
Table 464 – zwif_usrcod_kp_mod_set Parameters 162
Table 465 – zwif_usrcod_ms_cod_rpt_set Parameters 162
Table 466 – zwrep_ms_cod_fn Parameters 162
Table 467 – zwif_usrcod_ms_cod_get Parameters 162
Table 468 – zwif_usrcod_ms_cod_set Parameters 163
Table 469 – zwif_usrcod_chksum_rpt_set Parameters 163
Table 470 – zwrep_chksum_fn Parameters 163
Table 471 – zwif_usrcod_chksum_get Parameters 163
Table 472 – zwif_thrmo_fan_md_rpt_set Parameters 164
Table 473 – zwrep_thrmo_fan_md_fn Parameters 164
Table 474 – ZW_THRMO_FAN_MD_XXX 164
Table 475 – zwif_thrmo_fan_md_get Parameters 165
Table 476 – zwif_thrmo_fan_md_set Parameters 165
Table 477 – zwif_thrmo_fan_md_sup_get Parameters 165
Table 478 – zwrep_thrmo_fan_md_sup_fn Parameters 165
Table 479 – zwif_thrmo_fan_md_sup_cache_get Parameters 166
Table 480 – zwif_thrmo_fan_sta_rpt_set Parameters 166
Table 481 – zwrep_thrmo_fan_sta_fn Parameters 166
Table 482 – ZW_THRMO_FAN_STA_XXX 166
Table 483 – zwif_thrmo_fan_sta_get Parameters 167
Table 484 – zwif_thrmo_md_rpt_set Parameters 167
Table 485 – zwrep_thrmo_md_fn Parameters 167

Table 486 – ZW_THRMO_MD_XXX 168
Table 487 – zwif_thrmo_md_get Parameters 168
Table 488 – zwif_thrmo_md_set Parameters 169
Table 489 – zwif_thrmo_md_sup_get Parameters 169
Table 490 – zwrep_thrmo_md_sup_fn Parameters 169
Table 491 – zwif_thrmo_md_sup_cache_get Parameters 169
Table 492 – zwif_thrmo_op_sta_rpt_set Parameters 170
Table 493 – zwrep_thrmo_op_sta_fn Parameters 170
Table 494 – ZW_THRMO_OP_STA_XXX 170
Table 495 – zwif_thrmo_op_sta_get Parameters 171
Table 496 – zwif_thrmo_op_sta_log_sup_get Parameters 171
Table 497 – zwrep_thrmo_op_sta_log_sup_fn Parameters 171
Table 498 – zwif_thrmo_op_sta_log_sup_cache_get Parameters 171
Table 499 – zwif_thrmo_op_sta_log_rpt_set Parameters 172
Table 500 – zwrep_thrmo_op_sta_log_fn Parameters 172
Table 501 – zwthrmo_op_sta_log_t Structure 172
Table 502 – zwif_thrmo_op_sta_log_get Parameters 172
Table 503 – zwif_thrmo_setb_rpt_set Parameters 173
Table 504 – zwrep_thrmo_setb_fn Parameters 173
Table 505 – ZW_THRMO_SETB_TYP_XXX 173
Table 506 – ZW_THRMO_SETB_STA_XXX 173
Table 507 – zwif_thrmo_setb_get Parameters 173
Table 508 – zwif_thrmo_setb_set Parameters 174
Table 509 – zwif_thrmo_setp_rpt_set Parameters 174
Table 510 – zwrep_thrmo_setp_fn Parameters 174
Table 511 – zwsetp_t structure 174
Table 512 – ZW_THRMO_SETP_TYP_XXX 175
Table 513 – ZW_THRMO_SETP_UNIT_XXX 175
Table 514 – zwif_thrmo_setp_get Parameters 175
Table 515 – zwif_thrmo_setp_set Parameters 175
Table 516 – zwif_thrmo_setp_sup_get Parameters 176
Table 517 – zwrep_thrmo_setp_sup_fn Parameters 176
Table 518 – zwif_thrmo_setp_sup_cache_get Parameters 176
Table 519 – zwif_thrmo_setp_sup_range_get Parameters 176
Table 520 – zwrep_thrmo_setp_range_fn Parameters 177
Table 521 – zwif_thrmo_setp_sup_range_cache_get Parameters 177
Table 522 – zwif_config_rpt_set Parameters 177
Table 523 – zwrep_config_fn Parameters 178
Table 524 – zwconfig_t Structure 178
Table 525 – zwif_config_get Parameters 178
Table 526 – zwif_config_set Parameters 178
Table 527 – zwif_config_bulk_rpt_set Parameters 178
Table 528 – zwrep_cfg_bulk_fn Parameters 179
Table 529 – zwcfg_bulk_t Structure 179
Table 530 – zwif_config_bulk_get Parameters 179

Table 531 – zwif_config_bulk_set Parameters 180
Table 532 – zwif_config_prm_reset Parameters 180
Table 533 – zwif_config_info_get Parameters 180
Table 534 – zwcfg_info_cap_t structure 180
Table 535 – zwcfg_info_t structure 180
Table 536 – gen_dat_u Union 181
Table 537 – zwif_config_info_free Parameters 182
Table 538 – zwif_clock_rpt_set Parameters 182
Table 539 – zwrep_clock_fn Parameters 182
Table 540 – ZW_CLOCK_XXX 182
Table 541 – zwif_clock_get Parameters 182
Table 542 – zwif_clock_set Parameters 183
Table 543 – zwif_clmt_ctl_schd_rpt_set Parameters 183
Table 544 – zwrep_clmt_ctl_schd_fn Parameters 183
Table 545 – zwcc_shed_t Structure 183
Table 546 – zwcc_shed_swpt_t Structure 184
Table 547 – zwif_clmt_ctl_schd_get Parameters 184
Table 548 – zwif_clmt_ctl_schd_set Parameters 184
Table 549 – zwif_clmt_ctl_schd_chg_rpt_set Parameters 184
Table 550 – zwrep_clmt_ctl_schd_chg_fn Parameters 184
Table 551 – zwif_clmt_ctl_schd_chg_get Parameters 185
Table 552 – zwif_clmt_ctl_schd_ovr_rpt_set Parameters 185
Table 553 – zwrep_clmt_ctl_schd_ovr_fn Parameters 185
Table 554 – zwcc_shed_t Structure 185
Table 555 – zwif_clmt_ctl_schd_ovr_get Parameters 185
Table 556 – zwif_clmt_ctl_schd_ovr_set Parameters 186
Table 557 – zwif_av_set Parameters 186
Table 558 – zwif_av_caps Parameters 186
Table 559 – zwrep_av_fn Parameters 187
Table 560 – zwif_prot_rpt_set Parameters 187
Table 561 – zwrep_prot_fn Parameters 187
Table 562 – ZW_LPROT_XXX 187
Table 563 – ZW_RFPROT_XXX 188
Table 564 – zwif_prot_get Parameters 188
Table 565 – zwif_prot_set Parameters 188
Table 566 – zwif_prot_sup_get Parameters 188
Table 567 – zwrep_prot_sup_fn Parameters 188
Table 568 – zwprot_sup_t structure 189
Table 569 – zwif_prot_sup_cache_get Parameters 189
Table 570 – zwif_prot_ec_rpt_set Parameters 189
Table 571 – zwrep_prot_ec_fn Parameters 189
Table 572 – zwif_prot_ec_get Parameters 190
Table 573 – zwif_prot_ec_set Parameters 190
Table 574 – zwif_prot_tmout_rpt_set Parameters 190
Table 575 – zwrep_prot_tmout_fn Parameters 190

Table 576 – zwif_prot_tmout_get Parameters 191
Table 577 – zwif_prot_tmout_set Parameters....191
Table 578 – Top level entities....201
Table 579 – Network object 201
Table 580 – Node object....202
Table 581 – Z-Wave software version object 203
Table 582 – Endpoint object....203
Table 583 – Common Interface object 204
Table 584 – Interface object for Association Group Info CC....204
Table 585 –Association Group Info object....204
Table 586 – Command object....204
Table 587 – Interface object for Configuration CC 205
Table 588 – Configuration parameter object 205
Table 589 – Interface object for Association and Multi channel Association CC....205
Table 590 –Association Group Cache object 206
Table 591 –Group member object 206
Table 592 – Interface object for Central Scene CC 206
Table 593 –Scene object....206
Table 594 – Interface object for Multi Level Sensor CC....206
Table 595 –Multi level sensor object....207
Table 596 – Multi level sensor Cache object 207
Table 597 – Interface object for Indicator CC....207
Table 598 –Indicator object....207
Table 599 – Indicator Cache object 207
Table 600 – Indicator property object....207
Table 601 – Interface object for Thermostat Fan Mode CC 208
Table 602 – Thermostat Fan Mode Cache object....208
Table 603 – Interface object for Thermostat Mode CC 208
Table 604 – Thermostat Mode Cache object....208
Table 605 – Interface object for Thermostat Setpoint CC 209
Table 606 – Thermostat Setpoint object 209
Table 607 – Thermostat Setpoint Cache object....209
Table 608 – Interface object for Thermostat Operating State CC 210
Table 609 – Thermostat Operating State Cache object for “Operating State”....210
Table 610 – Thermostat Operating State Cache object for “Logging”....210
Table 611 – Interface object for Thermostat Fan State CC....210
Table 612 – Thermostat Fan State Cache object 211
Table 613 – Interface object for Multi Level Switch CC....211
Table 614 – Multi Level Switch Cache object 211
Table 615 – Interface object for Binary Switch CC....211
Table 616 – Binary Switch Cache object....211
Table 617 – Interface object for Binary Sensor CC 212
Table 618 – Binary Sensor Cache object....212
Table 619 – Interface object for Simple AV Control CC 212
Table 620 – Interface object for Alarm CC....212

Table 621 –Alarm object....212
Table 622 – Alarm Cache object....213
Table 623 – Alarm event parameter object....213
Table 624 – Interface object for Protection CC....213
Table 625 – Protection Cache object for “Protection State”....214
Table 626 – Protection Cache object for “Exclusive Control”....214
Table 627 – Protection Cache object for “Timeout”....214
Table 628 – Interface object for User Code CC....214
Table 629 – User Code Cache object....215
Table 630 – Interface object for Meter CC....215
Table 631 – Meter Cache object....215
Table 632 – Interface object for Meter Table Monitor CC....216
Table 633 – Interface object for Door Lock CC....216
Table 634 – Door Lock Cache object for “Operation Status”....217
Table 635 – Door Lock Cache object for “Configuration”....217
Table 636 – Interface object for Door Lock Logging CC....218
Table 637 – Interface object for Alarm Sensor CC....218
Table 638 – Alarm Sensor Cache object....219
Table 639 – Interface object for Barrier Operator CC....219
Table 640 – Barrier Operator Cache object....219
Table 641 – Interface object for Color Switch CC....219
Table 642 – Color Switch Cache object....219
Table 643 – Interface object for Sound Switch CC....220
Table 644 – Sound Switch object....220
Table 645 – Sound Switch Cache object for “Tone Played”....220
Table 646 – Sound Switch Cache object for “Tone Configuration”....220
Table 647 – Interface object for Window Covering CC....221
Table 648 – Window Covering Cache object....221
Table 649 – Interface object for Battery CC....221
Table 650 – Battery Cache object....221
Table 651 – Interface object for Basic CC....222
Table 652 – Basic Cache object....223

1 Introduction

1.1 Purpose

This document is the Z-Ware Library C API reference manual.

1.2 Audience and Prerequisites

This document is for Z-Wave Partners and assumes they are already comfortable with the Z-Wave protocol and network installation. As such, the document does not go into detail on these matters. References are also made to the Z-Wave Device Class (DC) and Command Class (CC) definitions and the Z-Wave for Internet Protocol (Z/IP) API, upon which this API is built.

2 Error Codes

The error codes used in APIs, ZW_ERR_XXX are listed in the table below. They are defined in the header file, zip_api.h. They can also be found in Doxygen documentation under the “Modules->Network APIs” Zw_error_codes section. Besides, occasionally, the error code returned may be from a low-level library function call. These error codes are listed in the table “Low-Level Library Error Codes”.

Table 1 – Error Codes, ZW_ERR_XXX

Error code	Description	Value
ZW_ERR_NONE *	Operation succeeded.	0
ZW_ERR_QUEUED *	Success: The command is queued into mailbox and will only be sent when the device wakes up (no transmission status callback).	1
ZW_ERR_CACHE_AVAIL *	Success: The data is available in the cache (no live get from the target device).	2
ZW_ERR_SEND_PENDING *	Success: The command is pending for sending in a short while (no transmission status callback).	3
ZW_ERR_FAILED	Operation failed.	-1
ZW_ERR_WRONG_IF	Wrong interface.	-10
ZW_ERR_NO_RESP	No response from controller.	-11
ZW_ERR_MEMORY	Out of memory.	-12
ZW_ERR_NODE_NOT_FOUND	Node not found.	-13
ZW_ERR_CLASS_NOT_FOUND	CC not found.	-14
ZW_ERR_INTF_NOT_FOUND	Interface to a class not found.	-15
ZW_ERR_INTF_NO_REP_HDLR	Interface report handler not found.	-16
ZW_ERR_LAST_OP_NOT_DONE	Last operation uncompleted yet, try again.	-17
ZW_ERR_NOT_IN_LIST	Node not in protocol layer failed node ID list.	-18
ZW_ERR_OP_FAILED	The requested operation failed.	-19
ZW_ERR_EP_NOT_FOUND	Endpoint not found.	-20
ZW_ERR_RPT_NOT_FOUND	The report command of an interface not found.	-21
ZW_ERR_NET_NOT_FOUND	Network not found.	-22
ZW_ERR_CMD_VERSION	Incorrect CC version.	-23
ZW_ERR_PENDING	Operation pending, it cannot be canceled now.	-24
ZW_ERR_VALUE	The parameter value is invalid.	-25
ZW_ERR_QUEUE_FULL	The queue is full.	-26
ZW_ERR_UNSUPPORTED	The requested function is unsupported for this node.	-27
ZW_ERR_FILE_OPEN	Open file error.	-28
ZW_ERR_FILE_WRITE	Write file error.	-29
ZW_ERR_FILE_EOF	The end-of-file was reached.	-30
ZW_ERR_FILE	File is corrupted.	-31
ZW_ERR_FILE_HOME_ID	File home ID doesn't match.	-32
ZW_ERR_EXPIRED	Expired.	-33
ZW_ERR_NO_RES	No resource for mutex, semaphore, timer, etc.	-34
ZW_ERR_EVENT	Event is not processed.	-35

ZW_ERR_TOO_LARGE	Data size is too large.	-36
ZW_ERR_TOO_SMALL	Data size is too small.	-37
ZW_ERR_TIMEOUT	Timeout.	-38
ZW_ERR_TRANSMIT	Transmission failed.	-39
ZW_ERR_NONCE_NOT_FOUND	Security nonce not found.	-40
ZW_ERR_AUTH	Authentication error.	-41
ZW_ERR_SEQ_NUMBER	Incorrect sequence number.	-42
ZW_ERR_BUSY	Busy, try again later.	-43
ZW_ERR_SEC_SCHEME	Security scheme unsupported.	-44
ZW_ERR_TRANSPORT_INI	Initialization error on transport layer.	-45
ZW_ERR_FRAME_INI	Initialization error on frame layer.	-46
ZW_ERR_SESSION_INI	Initialization error on session layer.	-47
ZW_ERR_APPL_INI	Initialization error on application layer.	-48
ZW_ERR_UNEXPECTED	The error was unexpected under normal circumstances.	-49
ZW_ERR_NETWORK_IF	Network interface not configured properly.	-50
ZW_ERR_IP_ADDR	The IP address is invalid.	-51
ZW_ERR_VERSION	Wrong version number.	-52
ZW_ERR_INTF_NO_DATA	Interface data is missing.	-53
ZW_ERR_FILE_READ	Read file error.	-54
ZW_ERR_PARSE_FILE	Parsing file failed.	-55
ZW_ERR_MISSING_ENTRY	Missing mandatory entry.	-56
ZW_ERR_DEVCFG_NOT_FOUND	Device specific configuration record not found.	-57
ZW_ERR_DISALLOWED	The operation is disallowed under certain circumstances.	-58
ZW_ERR_PSK_TOO_SHORT	DTLS pre-shared key length is too short.	-59
ZW_ERR_NO_CACHE_AVAIL	The data is unavailable in the cache.	-60
ZW_ERR_NOT_APPLICABLE	Not applicable and should be skipped or ignored.	-61
ZW_ERR_SHUTDOWN	The system is shutting down.	-62
ZW_ERR_POST_SET_POLL	Post-set polling error.	-63
ZW_ERR_ORDER	Out of order.	-64
ZW_ERR_CMD_UNSUPP_TGT	Command sent by the association group is unsupported at target node or endpoint.	-65
ZW_ERR_UNSUPP_S2_TGT	The association group node does not have the S2 target node highest key scheme HEALTH_CHK.	-66
ZW_ERR_UNSUPP_S2_SRC	The target node does not have the association group node S2 highest key scheme	-67
ZW_ERR_DIFF_HIGHEST_SEC	The association group node highest key scheme is different from target node's highest key scheme	-68

Note: Error code with (*) that has a value equal or greater than zero is not an error. In fact, it is an indication of success with more detailed description.

Table 2 – Low-Level Library Error Codes

Error code	Description	Value

ZWHCI_ERROR_MEMORY	Out of memory.	-100
ZWHCI_ERROR_RESOURCE	Out of resource.	-101
TRANSPORT_ERROR_SIGNALING	Could not signal write thread to perform actual write.	-102
FRAME_ERROR_MULTIPLE_WRITE	Write while the previous write in progress is not allowed.	-103
FRAME_ERROR_SEND_TIMER	Send timer not functioning.	-104
SESSION_ERROR_PREVIOUS_ COMMAND_UNCOMPLETED	Previous command uncompleted, retry later.	-105
SESSION_ERROR_DEST_BUSY	The message has not timed out yet.The destination host may have a long response time.	-106
SESSION_ERROR_UNREACHABLE	Frame failed to reach destination host.	-107
SESSION_ERROR_SND_FRM_TMOUT	Send frame timeout due to no ACK received.	-108
SESSION_ERROR_SYSTEM	System error, the program should exit.	-109
SESSION_ERROR_INVALID_RESP	The response command ID doesn't match with the sent command ID.	-110
SESSION_ERROR_SEND_BUSY	Send error at lower layer due to controller busy.	-111
APPL_TX_STATUS_TIMEOUT	There is no transmit status callback from lower layer.	-112
APPL_OPER_ALREADY_ACTIVE	The requested operation is already active.	-113
APPL_OPER_NOT_STARTED	The requested operation fails to start.	-114
APPL_ERROR_WAIT_CB	Waiting for transmit complete callback function, retry later.	-115
APPL_INVALID_ADDR	Invalid IPv6 address.	-116
APPL_INVALID_NODE_ID	Invalid node ID.	-117
APPL_ERROR_RSLV_NODE_ID	Couldn't resolve node ID to IP address.	-118
ZWHCI_ERROR_TOO_LARGE	The value supplied is too large.	-119
ZWHCI_ERROR_WRITE	Write failed.	-120
ZWHCI_ERROR_READ	Read failed.	-121
ZWHCI_ERROR_TIMEOUT	Time out.	-122
ZWHCI_ERROR_VERIFY	Verification failed.	-123
ZWHCI_ERROR_SHUTDOWN	System is shutting down.	-124

3 Portal API

These APIs are only used when the Z-Ware Library is in the Portal configuration.

3.1 zwportal_init

This call initializes the portal to listen for ZIPGW connections at well know TCP port 44123. The callback function zwportal_cb will be invoked when a ZIPGW connection has successfully completed the TLS handshake and the portal-gateway proprietary handshaking. The portal application may use the callback parameters to create an instance of Z/IP host controller using zwnet_init API.

Table 3 – zwportal_init Parameters

Attribute	Type	I/O	Description
init_prm	zwportal_init_t *	I	Portal initialization parameters.
return	void *	O	Context on success, NULL on failure. Caller is required to call zwportal_shutdown() and zwportal_exit() with the returned context if it is not null.

Table 4 – zwportal_init_t Structure

Attribute	Type	I/O	Description
clnt_prof	clnt_prof_t*	I	An array of client profiles.
prof_cnt	int	I	Number of client profiles in clnt_prof.
zwportal_cb	zwportal_cb_t	I	Callback function when a new gateway has completed the TLS handshake and the portal-gateway proprietary handshaking.
usr_param	void *	I	User defined parameter used in callback function.
ca_file	char *	I	File path to CA certificate that is used to sign ZIPGW public certificate.
ssl_file	char *	I	File path to SSL certificate (portal public certificate).
pvt_key_file	char *	I	File path to portal private key.
svr_port	uint16_t	I	Portal listening port.

Table 5 – clnt_prof_t Structure

Attribute	Type	I/O	Description
clnt_id	uint8_t[8]	I	Client's ID (currently using MAC address formatted as an IEEE EUI-64 identifier).
clnt_pin	uint8_t[8]	I	Client's PIN (password).
clnt_ipv6_addr	uint8_t[16]	I	Client's IPv6 address. May be all-zeroes IPv6 address.
clnt_dflt_gw	uint8_t[16]	I	Client's default IPv6 gateway.
clnt_pan_prefix	uint8_t[16]	I	Client's PAN interface prefix with /64 prefix length. May be all-zeroes IPv6 address.

clnt_unsolicited_dst	uint8_t[16]	I	Client's forwarding destination address for unsolicited message.
svr_ipv6_addr	uint8_t[16]	I	Server's IPv6 address.
clnt_unsolicited_port	uint16_t	I	Client's forwarding destination port for unsolicited.message. Should be 4123.
clnt_ipv6_prefix	uint8_t	I	Client's IPv6 address prefix length.
svr_ipv6_prefix	uint8_t	I	Server's IPv6 address prefix length.

Table 6 – zwportal_cb_t Parameters

Attribute	Type	I/O	Description
clnt_fd	int	I	Client socket file descriptor.
clnt_ssl	void *	I	Client SSL object pointer.
clnt_prof	clnt_prof_t*	I	Client profile used.
usr_param	void *	I	User defined parameter passed when calling zwportal_init().
return	int	O	Non-zero if the new client connection is accepted; zero if it is rejected.

3.2 zwportal_shutdown

This call closes listening socket at the well know TCP port 44123 to prevent any new gateway connections. Take note that this function should not be called in the callback function which was passed to the zwportal_init function.

Table 7 – zwportal_shutdown Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init().
return	int	O	Zero on success, non-zero on failure.

3.3 zwportal_exit

This call shuts down existing TLS connections and frees the resources used. Take note that this function should not be called in the callback function which was passed to the zwportal_init function.

Table 8 – zwportal_exit Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init ()
return	int	O	Zero on success, non-zero on failure.

3.4 zwportal_clnt_conn_close

This call closes client's socket and SSL connection.

Table 9 – zwportal_clnt_conn_close Parameters

Attribute	Type	I/O	Description
sfd	int	I	Client socket file descriptor
ssl	void *	I	Client SSL object pointer

3.5 zwportal_clnt_add

Add a client profile to the portal internal list, overwriting old entry with the same gateway.

Table 10 – zwportal_clnt_add Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init ().
clnt_prof	clnt_prof_t *	I	Client profile.
return	int	O	Non-zero on success, zero on failure.

3.6 zwportal_clnt_rm

Remove a client profile from the portal internal list.

Table 11 – zwportal_clnt_rm Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init ()
gw_id	uint8_t*	I	Gateway ID
return	int	O	Non-zero on success, zero on failure.

3.7 zwportal_clnt_find

Find a client profile in the portal internal list based on gateway ID.

Table 12 – zwportal_clnt_find Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init ()
clnt_prof	clnt_prof_t *	I/O	Client profile buffer with the gateway ID set as input key to be searched. On success, this buffer will be returned with found entry.
return	int	O	Non-zero on success, zero on failure.

3.8 zwportal_clnt_list_free

Free client profiles list.

Table 13 – zwportal_clnt_list_free Parameters

Attribute	Type	I/O	Description
lst_hd	clnt_prof_lst *	I	List head of the client profiles list

Table 14 – clnt_prof_lst structure

Attribute	Type	I/O	Description
next	clnt_prof_lst *	I	Next portal client profile in the linked-list
clnt_prof	clnt_prof_t	I	Client profile

3.9 zwportal_clnt_list_get

Get all the client profiles in the portal internal list.

Table 15 – zwportal_clnt_list_get Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwportal_init ().
lst_hd	clnt_prof_lst **	O	List head of the client profiles list.
return	int	O	Number of client profiles in the list; negative value on failure. Caller must call zwportal_clnt_list_free() to free the linked-list if return value is greater than zero.

4 Device Database API

These APIs load the device database.

4.1 zwdev_cfg_load

Load and store device-specific configurations.

Table 16 – zwdev_cfg_load Parameters

Attribute	Type	I/O	Description
cfg_file	const char *	I	Device-specific configurations file in JSON format.
records	dev_rec_t **	O	Device-specific configuration records sorted according to manufacturer ID, product type ID and product ID
record_cnt	uint16_t *	O	Number of device-specific configuration records stored in "records".
global_sett	dev_global_sett_t**	O	Device global settings.
global_sett_cnt	uint16_t *	O	Number of entries in device global settings.
err_loc	dev_cfg_error_t *	O	Parse error location for "device_records" in JSON file.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX. **Note**: Caller must free the memory allocated to "records" using zwdev_cfg_free() and memory allocated to "global_sett" using zwdev_global_sett_free() if return value is ZW_ERR_NONE.

Table 17 – dev_rec_t Structure

Attribute	Type	I/O	Description
vid	uint32_t	O	Vendor or Manufacturer ID
ptype	uint32_t	O	Product Type ID
pid	uint32_t	O	Product ID
category	uint8_t	O	Device category, DEV_XXX
ep_rec	ep_rec_t *	O	Endpoint records

Table 18 – DEV_XXX Meaning and Values

Device Category	Description	#
DEV_CATEGORY_UNKNOWN	Unknown or uncategorized device	0
DEV_SENSOR_ALARM	Sensor alarm	1
DEV_ON_OFF_SWITCH	On/off switch	2
DEV_POWER_STRIP	Power strip	3
DEV_SIREN	Siren	4
DEV_VALVE	Valve	5
DEV_SIMPLE_DISPLAY	Simple display	6
DEV_DOORLOCK_KEYPAD	Door lock with keypad	7
DEV_SUB_ENERGY_METER	Sub energy meter	8

DEV_ADV_WHL_HOME_ENER_METER	Advanced whole home energy meter	9
DEV_SIM_WHL_HOME_ENER_METER	Simple whole home energy meter	10
DEV_SENSOR	Sensor	11
DEV_LIGHT_DIMMER	Light dimmer switch	12
DEV_WIN_COVERING_NO_POS	Window covering no position/endpoint	13
DEV_WIN_COVERING_EP	Window covering end point aware	14
DEV_WIN_COVERING_POS_EP	Window covering position/endpoint-aware	15
DEV_FAN_SWITCH	Fan switch	16
DEV_RMT_CTL_MULTIPURPOSE	Remote control - multipurpose	17
DEV_RMT_CTL_AV	Remote control - AV	18
DEV_RMT_CTL_SIMPLE	Remote control - simple	19
DEV_UNRECOG_GATEWAY	Gateway (unrecognized by client)	20
DEV_CENTRAL_CTLR	Central controller	21
DEV_SET_TOP_BOX	Set top box	22
DEV_TV	TV	23
DEV_SUB_SYS_CTLR	Sub system controller	24
DEV_GATEWAY	Gateway	25
DEV_THERMOSTAT_HVAC	Thermostat - HVAC	26
DEV_THERMOSTAT_SETBACK	Thermostat - setback	27
DEV_WALL_CTLR	Wall controller	28

Table 19 – ep_rec_t Structure

Attribute	Type	I/O	Description
next	ep_rec_t *	O	Point to next endpoint record
id	uint8_t	O	Endpoint ID (starting from 0 for virtual endpoint, 1 and onwards for real endpoints)
new_if	uint16_t	O	New CC to be created/added to the endpoint, if any
new_if_ver	uint8_t	O	The "user defined version" for the new CC, if any
new_if_propty	uint8_t	O	New CC (interface) property (bit-mask)
redir_rec	redir_rec_t *	O	Command redirection records
if_rec	if_rec_t *	O	Interface records

Table 20 – redir_rec_t Structure

Attribute	Type	I/O	Description
next	redir_rec_t *	O	Next redirection record
if_id	uint16_t	O	Interface ID / CC ID.
cmd	int	O	Command value 0-255. -1 means don't care.
target_ep	uint8_t	O	Target endpoint ID to redirect to.

Table 21 – if_rec_t Structure

Attribute	Type	I/O	Description

next	if_rec_t *	O	Point to next interface record.
type	uint8_t	O	Interface type, IF_REC_TYPE_XXX.
usr_def_ver	uint8_t	O	User defined version.
propty	uint8_t	O	Interface type specific property (bit-mask).
tmout	uint16_t	O	Interface type specific timeout value in seconds.
rec	rec union	O	Union of interface records; the actual interface record used is indicated by the *type* attribute of this structure.

Table 22 – Interface Types: IF_REC_TYPE_XXX

Interface Record Type	Description	#
IF_REC_TYPE_GROUP	Group	1
IF_REC_TYPE_CONFIG	Configuration	2
IF_REC_TYPE_BIN_SENSOR	Binary Sensor	3
IF_REC_TYPE_SENSOR	Sensor	4
IF_REC_TYPE_METER	Meter	5
IF_REC_TYPE_ALARM	Alarm/Notification	6
IF_REC_TYPE_BASIC	Basic	7
IF_REC_TYPE_ALRM_SNSR	Alarm Sensor	8
IF_REC_TYPE_THRMO_SETP	Thermostat setpoint	9
IF_REC_TYPE_DOOR_LOCK	Door lock	10
IF_REC_TYPE_MULTI_CMD	Multi-command	11

Table 23 – rec Union

Attribute	Type	I/O	Description
grp	if_rec_grp_t *	O	Group interface record
config	if_rec_config_t *	O	Configuration interface record
bsnsr	if_rec_bsnsr_t *	O	Binary Sensor interface record
snsr	if_rec_snsr_t *	O	Sensor interface record
meter	if_rec_meter_t *	O	Meter interface record
alarm	if_rec_alarm_match_t*	O	Alarm//Notification interface record
basic	if_rec_basic_match_t*	O	Basic interface record
alrm_snsr	if_rec_alrm_snsr_match_t*	O	Alarm sensor interface record
thrmo_setp	if_rec_thrmo_setp_t*	O	Thermostat setpoint interface record
door_lck	if_rec_dlck_t *	O	Door lock interface record
mcmd	if_rec_mcmd_t *	O	Multi-command encapsulation record

Table 24 – if_rec_grp_t Structure

Attribute	Type	I/O	Description

grp_cnt	uint8_t	O	Number of group ID in the *grp_id* array
grp_id	uint8_t[7]	O	Group IDs for the controller to set its node ID into the groups

Table 25 – if_rec_config_t Structure

Attribute	Type	I/O	Description	
next	if_rec_config_t *	O	Point to next configuration interface record.	
rec_type	uint16_t	O	Record type CONFIG_REC_TYPE_XXX.	
			XXX	Description
			SET	Configuration record type set.
			INFO	Configuration record type information.
rec	union	O	Record as indicated by record type; union of the following:	
cfg_set	if_rec_cfg_set_t	O	For CONFIG_REC_TYPE_SET	
cfg_info	if_rec_cfg_info_t	O	For CONFIG_REC_TYPE_INFO	

Table 26 – if_rec_cfg_set_t Structure

Attribute	Type	I/O	Description
param_num	uint8_t	O	Parameter number
param_size	uint8_t	O	Data size: 1, 2, or 4 bytes
param_val	int32_t	O	Configuration parameter value (signed integer)

Table 27 – if_rec_cfg_info_t Structure

Attribute	Type	I/O	Description
param_num	uint8_t	O	Parameter number
param_size	uint8_t	O	Data size: 1,2,or 4 bytes
param_min	int32_t	O	Minimum configuration parameter value (signed integer)
param_max	int32_t	O	Maximum configuration parameter value (signed integer)
param_deflt	int32_t	O	Default configuration parameter value (signed integer)
param_name	char *	O	Parameter name
param_info	char *	O	Parameter information

Table 28 – if_rec_bsnsr_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Supported binary sensor type ZW_BSENSOR_TYPE_XXX (for version 1 only)
rec_match	if_rec_bsnsr_match_t *	O	Record matching condition

Table 29 – if_rec_bsnsr_match_t Structure

Attribute	Type	I/O	Description
pResult	void *	O	The result record that maps to, e.g., if_rec_alarm_result_t*
type	int	O	Binary sensor type ZW_BSENSOR_TYPE_XXX
state	int	O	Binary sensor state.
resultType	int	O	Type of the resulting conversion. IF_REC_TYPE_XXX

Table 30 – Table 31 – if_rec_snsr_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Sensor type ZW_SENSOR_TYPE_XXX
unit	uint8_t	O	Sensor unit ZW_SENSOR_UNIT_XXX

Table 32 – if_rec_meter_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	ZW_METER_TYPE_XXX.
unit_supp	uint8_t	O	Supported unit bit-mask : ZW_METER_SUP_UNIT_XXX.
rate_type	uint8_t	O	ZW_METER_RATE_XXX.
reset_cap	uint8_t	O	Meter reset capability: 1=capable to reset; 0=incapable of reset.

Table 33 – if_rec_alarm_match_t Structure

Attribute	Type	I/O	Description
next	if_rec_alarm_match_t *	O	Point to next match record
pResult	if_rec_alarm_result_t *	O	The result alarm record that maps to
pRevMatch	if_rec_alarm_rev_match_t *	O	Device specific alarm reverse-matching record (It is the OR result of match and result struct)
match_valid	int	O	Flag to indicate whether the match entries below are valid
type	int	O	Vendor proprietary alarm type
level_low	int	O	Vendor proprietary alarm level or lower limit of a range
level_high	int	O	Vendor proprietary alarm level higher limit of a range
isRange	int	O	Flag to indicate vendor proprietary alarm level is a range
ex_type	int	O	Z-Wave Alarm/Notification type (ZW_ALRM_XXX)
ex_event	int	O	Z-Wave Alarm/Notification event (ZW_ALRM_EVT_XXX)
ex_evt_len	int	O	Z-Wave Alarm/Notification event parameter length. Zero if the event has no parameter
pEx_evt_prm	uint8_t *	O	Pointer to Z-Wave Alarm/Notification event parameter

Table 34 – if_rec_alarm_result_t Structure

Attribute	Type	I/O	Description
type	int	O	Vendor proprietary alarm type
level	int	O	Vendor proprietary alarm level
ex_type	int	O	Z-Wave alarm/Notification type (ZW_ALRM_XXX)
type_name	char *	O	Device specific alarm type name.
ex_event	int	O	Z-Wave alarm/Notification event (ZW_ALRM_EVT_XXX).
level_name	char *	O	Device specific alarm level name.
ex_evt_len	int	O	Z-Wave alarm/Notification event parameter length. Zero if the event has no parameter.
ex_evt_type	int	O	Z-Wave alarm/Notification event parameter type (ZW_ALRM_PARAM_XXX).
pEx_evt_prm	uint8_t *	O	pointer to Z-Wave alarm/Notification event parameter.

Table 35 – if_rec_alarm_rev_match_t Structure

Attribute	Type	I/O	Description
type	int	O	Vendor proprietary alarm type
ex_type	int	O	Z-Wave alarm/Notification type (ZW_ALRM_XXX)
ex_event	int	O	Z-Wave alarm/Notification event (ZW_ALRM_EVT_XXX)

Table 36 – if_rec_basic_match_t Structure

Attribute	Type	I/O	Description
next	if_rec_basic_match_t *	O	Point to next match record.
pResult	void *	O	The result record that maps to, e.g., if_rec_alarm_result_t*.
command	int	O	Basic CC command. eg. BASIC_SET.
value	int	O	Command value 0-255. -1 means don't care.
resultType	Int	O	Type of the resulted conversion. IF_REC_TYPE_xxx.

Table 37 – if_rec_alrm_snsr_match_t Structure

Attribute	Type	I/O	Description
next	if_rec_alrm_snsr_match_t *	O	Point to next match record
pResult	void *	O	The result record that maps to. eg. if_rec_alarm_result_t*
type	int	O	Alarm sensor type. (ZW_ALRM_SNSR_TYPE_XXX)
state_low	int	O	Alarm sensor state or lower limit of a range. -1 means don't care.
state_high	int	O	Alarm sensor state higher limit of a range
isRange	int	O	Flag to indicate Alarm sensor state is a range
resultType	int	O	Type of the resulted conversion. IF_REC_TYPE_xxx

if_rec_thrmo_setp_t Structure

Attribute	Type	I/O	Description

type_cnt	uint8_t	O	Number of thermostat setpoint-supported temperature ranges
temp_range	zwsetp_temp_range_t *	O	Temperature ranges

Table 38 – zwsetp_temp_range_t Structure

Attribute	Type	I/O	Description
min	zwsetp_t	O	Minimum setpoint temperature
max	zwsetp_t	O	Maximum setpoint temperature

Table 39 – if_rec_dlck_t Structure

Attribute	Type	I/O	Description
op_type_cnt	uint8_t	O	Number of supported door lock operation types in "op_type" buffer.
mode_cnt	uint8_t	O	Number of supported door lock modes in "mode" buffer.
out_hdl	uint8_t	O	Supported outside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if the corresponding handle can be enabled and disabled; otherwise, the corresponding handle cannot be enabled or disabled.
in_hdl	uint8_t	O	Supported inside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if the corresponding handle can be enabled and disabled; otherwise, the corresponding handle cannot be enabled or disabled.
op_type	uint8_t *	O	Door lock operation type (ZW_DOOR_OP_XXX) buffer.
mode	uint8_t *	O	Door lock mode (ZW_DOOR_XXX except ZW_DOOR_UNKNOWN) buffer.

Table 40 – if_rec_mcmd_t Structure

Attribute	Type	I/O	Description
send_encap	uint8_t	O	Control whether to send consecutive commands using multi-command encapsulation

Table 41 – Binary Sensor Types: ZW_BSENSOR_TYPE_XXX

Binary Sensor Type	Description	#
ZW_BSENSOR_TYPE_GP	General purpose sensor	1
ZW_BSENSOR_TYPE_SMOKE	Smoke sensor	2
ZW_BSENSOR_TYPE_CO	CO sensor	3
ZW_BSENSOR_TYPE_CO2	CO2 sensor	4
ZW_BSENSOR_TYPE_HEAT	Heat sensor	5
ZW_BSENSOR_TYPE_WATER	Water sensor	6
ZW_BSENSOR_TYPE_FREEZE	Freeze sensor	7
ZW_BSENSOR_TYPE_TAMPER	Tamper sensor	8
ZW_BSENSOR_TYPE_AUX	Aux sensor	9
ZW_BSENSOR_TYPE_DR_WIN	Door/Window sensor	10

ZW_BSENSOR_TYPE_TILT	Tilt sensor	11
ZW_BSENSOR_TYPE_MOTION	Motion sensor	12
ZW_BSENSOR_TYPE_GLASS_BRK	Glass break sensor	13
ZW_BSENSOR_TYPE_1ST_SUP	Return first sensor type on supported list	

Table 42 – Multi-Level Sensor Type: ZW_SENSOR_TYPE_XXX

Multi-level Sensor Type	Description	#
ZW_SENSOR_TYPE_TEMP	Temperature sensor	1
ZW_SENSOR_TYPE_GP	General purpose sensor	2
ZW_SENSOR_TYPE_LUM	Luminance sensor	3
ZW_SENSOR_TYPE_POW	Power sensor	4
ZW_SENSOR_TYPE_HUMD	Relative humidity sensor	5
ZW_SENSOR_TYPE_VELO	Velocity sensor	6
ZW_SENSOR_TYPE_DIR	Direction sensor	7
ZW_SENSOR_TYPE_ATM	Atmospheric pressure sensor	8
ZW_SENSOR_TYPE_BARO	Barometric pressure sensor	9
ZW_SENSOR_TYPE_SLR	Solar radiation sensor	10
ZW_SENSOR_TYPE_DEW	Dew point sensor	11
ZW_SENSOR_TYPE_RAIN	Rain rate sensor	12
ZW_SENSOR_TYPE_TIDE	Tide level sensor	13
ZW_SENSOR_TYPE_WGT	Weight sensor	14
ZW_SENSOR_TYPE_VOLT	Voltage sensor	15
ZW_SENSOR_TYPE_CUR	Current sensor	16
ZW_SENSOR_TYPE_CO2	CO2-level sensor	17
ZW_SENSOR_TYPE_AIR	Air flow sensor	18
ZW_SENSOR_TYPE_TANK	Tank capacity sensor	19
ZW_SENSOR_TYPE_DIST	Distance sensor	20
ZW_SENSOR_TYPE_AGL	Angle Position sensor	21
ZW_SENSOR_TYPE_ROT	Rotation sensor	22
ZW_SENSOR_TYPE_WTR_TEMP	Water temperature sensor	23
ZW_SENSOR_TYPE_SOIL_TEMP	Soil temperature sensor	24
ZW_SENSOR_TYPE_SEIS_INT	Seismic intensity sensor	25
ZW_SENSOR_TYPE_SEIS_MAG	Seismic magnitude sensor	26
ZW_SENSOR_TYPE_UV	Ultraviolet sensor	27
ZW_SENSOR_TYPE_ELEC_RES	Electrical resistivity sensor	28
ZW_SENSOR_TYPE_ELEC_COND	Electrical conductivity sensor	29
ZW_SENSOR_TYPE_LOUDNESS	Loudness sensor	30
ZW_SENSOR_TYPE_MOIST	Moisture sensor	31
ZW_SENSOR_TYPE_FREQ	Frequency sensor	32
ZW_SENSOR_TYPE_TIME	Time sensor	33
ZW_SENSOR_TYPE_TGT_TEMP	Target temperature sensor	34
ZW_SENSOR_TYPE_PM_2_5	Particulate matter 2.5 sensor	35
ZW_SENSOR_TYPE_F_CH2O	Formaldehyde CH2O-level sensor	36

ZW_SENSOR_TYPE_RAD_CONT	Radon Concentration sensor	37
ZW_SENSOR_TYPE_METH_DENS	Methane Density CH4 sensor	38
ZW_SENSOR_TYPE_VOC	Volatile Organic Compound sensor	39
ZW_SENSOR_TYPE_CO_LVL	Carbon Monoxide CO-level sensor	40
ZW_SENSOR_TYPE_SOIL_HUMD	Soil Humidity sensor	41
ZW_SENSOR_TYPE_SOIL_REAC	Soil Reactivity sensor	42
ZW_SENSOR_TYPE_SOIL_SAL	Soil Salinity sensor	43
ZW_SENSOR_TYPE_HEART_RT	Heart Rate sensor	44
ZW_SENSOR_TYPE_BLOOD_PRS	Blood Pressure sensor	45
ZW_SENSOR_TYPE_MUSCLE_MS	Muscle Mass sensor	46
ZW_SENSOR_TYPE_FAT_MS	Fat Mass sensor	47
ZW_SENSOR_TYPE_BONE_MS	Bone Mass sensor	48
ZW_SENSOR_TYPE_TBW	Total Body Water sensor	49
ZW_SENSOR_TYPE_BMR	Basic Metabolic Rate sensor	50
ZW_SENSOR_TYPE_BMI	Body Mass Index sensor	51
ZW_SENSOR_TYPE_ACCEL_X	Acceleration, X-axis sensor	52
ZW_SENSOR_TYPE_ACCEL_Y	Acceleration, Y-axis sensor	53
ZW_SENSOR_TYPE_ACCEL_Z	Acceleration, Z-axis sensor	54
ZW_SENSOR_TYPE_SMOKE_DEN	Smoke Density sensor	55
ZW_SENSOR_TYPE_WATER_FLW	Water Flow sensor	56
ZW_SENSOR_TYPE_WATER_PRS	Water Pressure sensor	57
ZW_SENSOR_TYPE_RF_SGN	RF Signal Strength sensor	58
ZW_SENSOR_TYPE_PM_10	Particulate Matter 10 sensor	59
ZW_SENSOR_TYPE_RESPI_RATE	Respiratory rate sensor	60
ZW_SENSOR_TYPE_REL_MOD	Relative Modulation level sensor	61
ZW_SENSOR_TYPE_BOILER_WTR_TEMP	Boiler water temperature sensor	62
ZW_SENSOR_TYPE_DHW_TEMP	Domestic Hot Water (DHW) temperature sensor	63
ZW_SENSOR_TYPE_OUTSIDE_TEMP	Outside temperature sensor	64
ZW_SENSOR_TYPE_EXHAUST_TEMP	Exhaust temperature sensor	65
ZW_SENSOR_TYPE_WATER_CHLOR_LVL	Water Chlorine level sensor	66
ZW_SENSOR_TYPE_WATER_ACID	Water acidity sensor	67
ZW_SENSOR_TYPE_WATER_OXI_RED	Water Oxidation reduction potential sensor	68

Table 43 – Multi-Level Sensor Unit

Multi-level Sensor Type	Unit	#
ZW_SENSOR_TYPE_TEMP	Celsius (C)	0
	Fahrenheit (F)	1
ZW_SENSOR_TYPE_GP	Percentage value	0

	Dimensionless value	1
ZW_SENSOR_TYPE_LUM	Percentage value	0
	Lux	1
ZW_SENSOR_TYPE_POW	W	0
	Btu/h	1
ZW_SENSOR_TYPE_HUMD	Percentage value	0
	g/m3	1
ZW_SENSOR_TYPE_VELO	m/s	0
	mph	1
ZW_SENSOR_TYPE_DIR	Direction unit: 0 to 360 degrees. 0 = no wind, 90 = east,180 = south, 270 = west, and 360 = north	0
ZW_SENSOR_TYPE_ATM	kPa	0
	inches of Mercury	1
ZW_SENSOR_TYPE_BARO	kPa	0
	inches of Mercury	1
ZW_SENSOR_TYPE_SLR	W/m2	0
ZW_SENSOR_TYPE_DEW	Celsius (C)	0
	Fahrenheit (F)	1
ZW_SENSOR_TYPE_RAIN	mm/h	0
	in/h	1
ZW_SENSOR_TYPE_TIDE	m	0
	feet	1
ZW_SENSOR_TYPE_WGT	kg	0
	pounds	1
ZW_SENSOR_TYPE_VOLT	V	0
	mV	1
ZW_SENSOR_TYPE_CUR	A	0
	mA	1
ZW_SENSOR_TYPE_CO2	ppm	0
ZW_SENSOR_TYPE_AIR	m3/h	0
	cfm (cubic feet per minute)	1
ZW_SENSOR_TYPE_TANK	l	0
	cbm	1
	US gallons	2
ZW_SENSOR_TYPE_DIST	m	0
	cm	1
	feet	2
ZW_SENSOR_TYPE_AGL	Percentage value	0
	Degrees relative to north pole of standing eye view	1
	Degrees relative to south pole of standing eye view	2
ZW_SENSOR_TYPE_ROT	rpm (revolutions per minute)	0

	Hz (Hertz)	1
ZW_SENSOR_TYPE_WTR_TEMP	Celsius (C)	0
	Fahrenheit (F)	1
ZW_SENSOR_TYPE_SOIL_TEMP	Celsius (C)	0
	Fahrenheit (F)	1
ZW_SENSOR_TYPE_SEIS_INT	Mercalli	0
	European Macroseismic	1
	Liedu	2
	Shindo	3
ZW_SENSOR_TYPE_SEIS_MAG	Local (ML)	0
	Moment (MW)	1
	Surface wave (MS)	2
	Body wave (MB)	3
ZW_SENSOR_TYPE_UV	UV index	0
ZW_SENSOR_TYPE_ELEC_RES	ohm metre	0
ZW_SENSOR_TYPE_ELEC_COND	siemens per metre (S/m)	0
ZW_SENSOR_TYPE_LOUDNESS	Absolute loudness (dB)	0
	A-weighted decibels (dBA)	1
ZW_SENSOR_TYPE_MOIST	Percentage value	0
	Volume water content (m3/m3)	1
	Impedance (k ohm)	2
	Water activity (aw)	3
ZW_SENSOR_TYPE_FREQ	Hz - Must be used until 4.294967295 GHz	0
	kHz- Must be used after 4.294967295 GHz	1
ZW_SENSOR_TYPE_TIME	Seconds	0
ZW_SENSOR_TYPE_TGT_TEMP	Celsius (C)	0
	Fahrenheit (F)	1
ZW_SENSOR_TYPE_PM_2_5	Moles per cubic meter (mol/m^3)	0
	Absolute microgram/cubic meter (ug/m^3)	1
ZW_SENSOR_TYPE_F_CH2O	Moles per cubic meter (mol/m^3)	0
ZW_SENSOR_TYPE_RAD_CONT	Becquerel/cubic meter (bq/m3)	0
	Picocuries/liter (pCi/L)	1
ZW_SENSOR_TYPE_METH_DENS	Moles per cubic meter (mol/m^3)	0
ZW_SENSOR_TYPE_VOC	Moles per cubic meter (mol/m^3)	0
ZW_SENSOR_TYPE_CO_LVL	Moles per cubic meter (mol/m^3)	0
ZW_SENSOR_TYPE_SOIL_HUMD	Percentage value	0
ZW_SENSOR_TYPE_SOIL_REAC	Acidity (pH)	0
ZW_SENSOR_TYPE_SOIL_SAL	Moles per cubic meter (mol/m^3)	0
ZW_SENSOR_TYPE_HEART_RT	Beats/minute (Bpm)	0
ZW_SENSOR_TYPE_BLOOD_PRS	Systolic mmHg (upper number)	0
	Diastolic mmHg (lower number)	1
ZW_SENSOR_TYPE_MUSCLE_MS	Kilograms (kg)	0
ZW_SENSOR_TYPE_FAT_MS	Kilograms (kg)	0
ZW_SENSOR_TYPE_BONE_MS	Kilograms (kg)	0

ZW_SENSOR_TYPE_TBW	Kilograms (kg)	0
ZW_SENSOR_TYPE_BMR	Joules (J)	0
ZW_SENSOR_TYPE_BMI	BMI Index	0
ZW_SENSOR_TYPE_ACCEL_X	(m/s^2)	0
ZW_SENSOR_TYPE_ACCEL_Y	(m/s^2)	0
ZW_SENSOR_TYPE_ACCEL_Z	(m/s^2)	0
ZW_SENSOR_TYPE_SMOKE_DEN	Percentage value	0
ZW_SENSOR_TYPE_WATER_FLW	Liters/hour (l/h)	0
ZW_SENSOR_TYPE_WATER_PRS	Kilopascals (kPa)	0
ZW_SENSOR_TYPE_RF_SGN	RSSI (Percentage value)	0
	(dBm)	1
ZW_SENSOR_TYPE_PM_10	Moles per cubic meter (mol/m3)	0
	Micrograms per cubic meter (µg/m3)	1
ZW_SENSOR_TYPE_RESPI_RATE	Breaths per minute (bpm)	0
ZW_SENSOR_TYPE_REL_MOD	Percentage value (%)	0
ZW_SENSOR_TYPE_BOILER_WTR_TEMP	Celsius (C)	0
ZW_SENSOR_TYPE_DHW_TEMP	Celsius (C)	0
ZW_SENSOR_TYPE_OUTSIDE_TEMP	Celsius (C)	0
ZW_SENSOR_TYPE_EXHAUST_TEMP	Celsius (C)	0
ZW_SENSOR_TYPE_WATER_CHLOR_LVL	Milligram per liter (mg/l)	0
ZW_SENSOR_TYPE_WATER_ACID	Acidity (pH)	0
ZW_SENSOR_TYPE_WATER_OXI_RED	MilliVolt (mV)	0

Table 44 – Meter Type

Meter Type	Description	#
ZW_METER_TYPE_ELEC	Electric meter	1
ZW_METER_TYPE_GAS	Gas meter	2
ZW_METER_TYPE_WATER	Water meter	3

Table 45 – Meter Supported Unit Bit-Mask

Multi-level Sensor Type	Unit	Bit-mask
ZW_METER_TYPE_ELEC	kWh	0x01
	kVAh	0x02
	W	0x04
	Pulse count	0x08
	V	0x10
	A	0x20
	Power factor	0x40
ZW_METER_TYPE_GAS	Cubic meters	0x01
	Cubic feet	0x02
	Pulse count	0x08
ZW_METER_TYPE_WATER	Cubic meters	0x01
	Cubic feet	0x02

	US gallons	0x04
	Pulse count	0x08

Table 46 – Z-Wave Alarm/Notification Type

Z-Wave Alarm Type	Description	#
ZW_ALRM_SMOKE	Smoke alarm	1
ZW_ALRM_CO	Carbon monoxide alarm	2
ZW_ALRM_CO2	Carbon dioxide alarm	3
ZW_ALRM_HEAT	Heat alarm	4
ZW_ALRM_WATER	Water alarm	5
ZW_ALRM_LOCK	Lock access control alarm	6
ZW_ALRM_BURGLAR	Burglar alarm or home security	7
ZW_ALRM_POWER	Power management alarm	8
ZW_ALRM_SYSTEM	System alarm	9
ZW_ALRM_EMERGENCY	Emergency alarm	10
ZW_ALRM_CLOCK	Alarm clock	11
ZW_ALRM_APPLIANCE	Home appliance alarm	12
ZW_ALRM_HEALTH	Home health alarm	13
ZW_ALRM_SIREN	Siren alarm	14
ZW_ALRM_WATER_VLV	Water Valve alarm	15
ZW_ALRM_WEATHER	Weather alarm	16
ZW_ALRM_IRRIGATION	Irrigation alarm	17
ZW_ALRM_GAS	Gas alarm	18
ZW_ALRM_PEST_CTL	Pest control	19
ZW_ALRM_LIGHT_SNSR	Light sensor	20
ZW_ALRM_WATER_QM	Water quality monitoring	21
ZW_ALRM_HOME_MNTR	Home monitoring	22

Table 47 – Z-Wave Alarm/Notification Event

Z-Wave Alarm/Notification type	Z-Wave Alarm/Notification Event	#
All	Unknown event.	254
	Event inactive (push mode) / Previous Events cleared (pull mode).	0
Smoke alarm	Smoke detected with location.	1
	Smoke detected.	2
	Smoke alarm test.	3
	Replacement required.	4
	Replacement required, End-of-life	5
	Maintenance required, planned periodic inspection	7
	Maintenance required, dust in device	8
Carbon monoxide alarm	Carbon monoxide detected with location.	1
	Carbon monoxide detected.	2
	Carbon monoxide test. Event parameter value: 1=OK, 2=Failed	3

	Replacement required.	4
	Replacement required, End-of-life	5
	Alarm silenced	6
	Maintenance required, planned periodic inspection	7
Carbon dioxide alarm	Carbon dioxide detected with location.	1
	Carbon dioxide detected.	2
	Carbon dioxide test. Event parameter value: 1=OK, 2=Failed	3
	Replacement required.	4
	Replacement required, End-of-life	5
	Alarm silenced	6
	Maintenance required, planned periodic inspection	7
Heat alarm	Overheat detected with location info.	1
	Overheat detected.	2
	Rapid temperature rise detected with location.	3
	Rapid temperature rise detected.	4
	Underheat detected with location.	5
	Underheat detected.	6
	Heat alarm test	7
	Replacement required, End-of-life	8
	Alarm silenced	9
	Maintenance required, dust in device	10
	Maintenance required, planned periodic inspection	11
	Rapid temperature fall with location info	12
	Rapid temperature fall	13
Water alarm	Water leak detected with location.	1
	Water leak detected.	2
	Water level drop detected with location.	3
	Water level dropped detected.	4
	Replace Water Filter.	5
	Water flow. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold, 4=max	6
	Water pressure. Event parameter meaning same as those for water flow.	7
	Water temperature. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold	8
	Water level. Event parameter meaning same as those for water temperature.	9
	Sump pump active	10
	Sump pump failure	11
Lock access control alarm	Manual lock operation.	1
	Manual unlock operation.	2
	RF lock operation.	3
	RF unlock operation.	4
	Keypad lock operation with user identifier.	5

	Keypad unlock operation with user identifier.	6
	Manual not fully locked operation.	7
	RF not fully locked operation.	8
	Auto lock locked operation.	9
	Auto lock not fully operation.	10
	Lock jammed.	11
	All user codes deleted.	12
	Single user code deleted.	13
	New user code added.	14
	New user code not added due to duplicate code.	15
	Keypad temporary disabled.	16
	Keypad busy.	17
	New program code entered - unique code for lock configuration.	18
	Manually enter user access code exceeds code limit.	19
	Unlock by RF with invalid user code.	20
	Locked by RF with invalid user code.	21
	Window/door is open.	22
	Window/door is closed.	23
	Window/door handle is open	24
	Window/door handle is closed	25
	User Code entered via keypad (with event param: ZW_ALRM_PARAM_USRID)	32
	Barrier performing Initialization process. Event parameter value: 0=Completed, 0xFF=In progress	64
	Barrier operation (Open / Close) force has been exceeded.	65
	Barrier motor has exceeded manufacturer's operational time limit. Event parameter value: 0 to 0x7F = 0 to 127 seconds; 0x80 to 0xFE = 1 to 127 minutes	66
	Barrier operation has exceeded physical mechanical limits. (For example: barrier has opened past the open limit)	67
	Barrier unable to perform requested operation due to UL requirements.	68
	Barrier Unattended operation has been disabled per UL requirements	69
	Barrier failed to perform Requested operation, device malfunction	70
	Barrier Vacation Mode. Event parameter value: 0=disabled, 0xFF=enabled	71
	Barrier Safety Beam Obstacle. Event parameter value: 0=no obstruction, 0xFF=obstruction	72
	Barrier Sensor Not Detected / Supervisory Error. Event parameter value: 0=sensor not defined, 1 to 0xFF=sensor ID	73
	Barrier Sensor Low Battery Warning. Event parameter value: 0=sensor not defined, 1 to 0xFF=sensor ID	74
	Barrier detected short in Wall Station wires	75
	Barrier associated with non-Z-Wave remote control	76

Burglar alarm	Intrusion detected with location.	1
	Intrusion detected.	2
	Tampering, product covering removed.	3
	Tampering, Invalid Code.	4
	Glass breakage detected with location.	5
	Glass breakage detected.	6
	Motion detected with location info.	7
	Motion detected with unknown location info.	8
Power management alarm	Power has been applied.	1
	AC mains disconnected.	2
	AC mains re-connected.	3
	Surge Detection.	4
	Voltage Drop/Drift detected.	5
	Over-current detected.	6
	Over-voltage detected.	7
	Over-load detected.	8
	Load error.	9
	Replace battery soon.	10
	Replace battery now.	11
	Battery is charging.	12
	Battery is fully charged.	13
	Charge battery soon.	14
	Charge battery now.	15
	Back-up battery is low	16
	Battery fluid is low	17
System alarm	System hardware failure.	1
	System software failure.	2
	System hardware failure with OEM proprietary failure code.	3
	System software failure with OEM proprietary failure code.	4
Emergency alarm	Contact police.	1
	Contact fire service.	2
	Contact medical service.	3
Alarm clock	Wake up.	1
	Timer ended.	2
	Time remaining. Event parameter value (3 bytes): byte-0 unit = hours, byte-1 unit = minutes, byte-2 unit = seconds	3
Barrier	Barrier performing Initialization process.	64
	Barrier operation (Open / Close) force has been exceeded.	65
	Barrier motor has exceeded manufacturer’s operational time limit.	66
	Barrier operation has exceeded physical mechanical limits. (For example, barrier has opened past the open limit.)	67
	Barrier unable to perform requested operation due to UL requirements.	68
	Barrier Unattended operation has been disabled per UL requirements.	69

	Barrier failed to perform Requested operation, device malfunction.	70
	Barrier Vacation Mode.	71
	Barrier Safety Beam Obstacle.	72
	Barrier Sensor Not Detected / Supervisory Error.	73
	Barrier Sensor Low Battery Warning.	74
	Barrier detected short in Wall Station wires.	75
	Barrier associated with non-Z-Wave remote control.	76
Appliance	Program started.	1
	Program in progress.	2
	Program completed.	3
	Replace main filter.	4
	Failure to set target temperature.	5
	Supplying water.	6
	Water supply failure.	7
	Boiling.	8
	Boiling failure.	9
	Washing.	10
	Washing failure.	11
	Rinsing.	12
	Rinsing failure.	13
	Draining.	14
	Draining failure.	15
	Spinning.	16
	Spinning failure	17
	Drying	18
	Drying failure.	19
	Fan failure.	20
	Compressor failure.	21
Home Health	Leaving Bed.	1
	Sitting on bed.	2
	Lying on bed.	3
	Posture changed.	4
	Sitting on edge of bed.	5
	Volatile Organic Compound level. Event parameter value (pollution level):1=clean, 2=Slightly polluted, 3=Moderately polluted, 4=Highly polluted	6
	Sleep apnea detected. Event parameter value (breath level): 1=low breath, 2=No breath at all	7
	Sleep stage 0 detected: Dreaming/REM	8
	Sleep stage 1 detected: Light sleep, non-REM 1	9
	Sleep stage 2 detected: Medium sleep, non-REM 2	10
	Sleep stage 3 detected: Deep sleep, non-REM 3	11
Siren	Siren Active.	1
Water Valve	Valve Operation. Event parameter value: 0=closed, 1=open	1

	Master Valve Operation. Event parameter value: 0=closed, 1=open	2
	Valve Short Circuit.	3
	Master Valve Short Circuit.	4
	Valve Current Alarm. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold, 4=max	5
	Master Valve Current Alarm. Event parameter value same as Valve Current Alarm	6
Weather	Rain.	1
	Moisture.	2
	Freeze	3
Irrigation	Schedule Started. Event parameter value is schedule ID	1
	Schedule Finished. Event parameter value is schedule ID	2
	Valve Table Run Started. Event parameter value is valve table ID	3
	Valve Table Run Finished. Event parameter value is valve table ID	4
	Device is not Configured.	5
Gas	Combustible Gas detected with location info.	1
	Combustible Gas detected with unknown location info.	2
	Toxic Gas detected with location info.	3
	Toxic Gas detected with unknown location info.	4
	Gas Alarm Test.	5
	Replacement Required.	6
Pest Control	Trap armed with location info	1
	Trap armed	2
	Trap re-arm required with location info	3
	Trap re-arm required	4
	Pest detected with location info	5
	Pest detected	6
	Pest exterminated with location info	7
	Pest exterminated	8
Light Sensor	Light detected	1
	Light color transition detected	2
Water Quality Monitoring	Chlorine alarm. Event parameter value: 1=Below low threshold, 2=Above high threshold	1
	Acidity (pH). Event parameter value: 1=Below low threshold, 2=Above high threshold, 3=Decreasing pH, 4=Increasing pH	2
	Water Oxidation alarm. Event parameter value: 1=Below low threshold, 2=Above high threshold	3
	Chlorine empty	4
	Acidity empty	5
	Waterflow measuring station shortage detected	6
	Waterflow clear water shortage detected	7
	Disinfection system error detected. Event parameter value (bit-mask): bit 0~3: System 1~4 disorder detected, bit 4~7: System 1~4 salt shortage	8

	Filter cleaning ongoing. Event parameter value: 1~255= Filter number	9
	Heating operation ongoing	10
	Filter pump operation ongoing	11
	Freshwater operation ongoing	12
	Dry protection operation active	13
	Water tank is empty	14
	Water tank level is unknown	15
	Water tank is full	16
	Collective disorder	17
Home Monitoring	Home occupied with location info	1
	Home occupied	2

Table 48 – Z-Wave Alarm/Notification Event Parameter Type

Z-Wave Alarm/Notification Event Parameter Type	Description	#
ZW_ALRM_PARAM_LOC	Node location UTF-8 string (NULL terminated).	1
ZW_ALRM_PARAM_USRID	User ID. 1 or 2 bytes long; if 2 bytes, the first byte is the MSB	2
ZW_ALRM_PARAM_OEM_ERR_CODE	OEM proprietary system failure code.	3
ZW_ALRM_PARAM_PROPRIETARY	Proprietary event parameters.	4
ZW_ALRM_PARAM_EVENT_ID	Event ID which is no more active.	5
ZW_ALRM_PARAM_UNKNOWN	Unknown alarm event parameters. It could be from a higher version of CC, or the device violates the spec and send parameters when Spec does not define.	255

Table 49 – Alarm Sensor Type

Alarm Sensor Type	Description	Assigned Number
ZW_ALRM_SNSR_TYPE_GP	General Purpose Alarm	0
ZW_ALRM_SNSR_TYPE_SMOKE	Smoke Alarm	1
ZW_ALRM_SNSR_TYPE_CO	CO Alarm	2
ZW_ALRM_SNSR_TYPE_CO2	CO2 Alarm	3
ZW_ALRM_SNSR_TYPE_HEAT	Heat Alarm	4
ZW_ALRM_SNSR_TYPE_WATER_LEAK	Water Leak Alarm	5

Table 50 – Thermostat Setpoint Types: ZW_THRMO_SETP_TYP_XXX

Thermostat Setpoint Type	Description	Assigned Number
ZW_THRMO_SETP_TYP_HEATING	Heating	1
ZW_THRMO_SETP_TYP_COOLING	Cooling	2
ZW_THRMO_SETP_TYP_FURNACE	Furnace	7
ZW_THRMO_SETP_TYP_DRY	Dry air	8

ZW_THRMO_SETP_TYP_MOIST	Moist air	9
ZW_THRMO_SETP_TYP_AUTO_CHANGEOVER	Auto changeover	10
ZW_THRMO_SETP_TYP_ENE_SAVE_HEAT	Energy Save Heating	11
ZW_THRMO_SETP_TYP_ENE_SAVE_COOL	Energy Save Cooling	12
ZW_THRMO_SETP_TYP_AWAY_HEAT	Away heating	13
ZW_THRMO_SETP_TYP_AWAY_COOL	Away cooling	14
ZW_THRMO_SETP_TYP_FULL_POWER	Full power	15

Table 51 – Thermostat Setpoint Unit

Thermostat Setpoint Unit	Description	Assigned Number
ZW_THRMO_SETP_UNIT_C	Celsius	0
ZW_THRMO_SETP_UNIT_F	Fahrenheit	1

Table 52 – Door Lock Operation Mode

Door lock Operation Mode	Description	Assigned Number
ZW_DOOR_UNSEC	Door unsecured.	0
ZW_DOOR_UNSEC_TMOUT	Door unsecured with timeout. Fallback to secured mode after timeout has expired.	1
ZW_DOOR_UNSEC_IN	Door unsecured for inside door handles.	16
ZW_DOOR_UNSEC_IN_TMOUT	Door unsecured for inside door handles with timeout.	17
ZW_DOOR_UNSEC_OUT	Door unsecured for outside door handles.	32
ZW_DOOR_UNSEC_OUT_TMOUT	Door unsecured for outside door handles with timeout.	33
ZW_DOOR_SEC	Door secured.	255

Table 53 – Door lock Operation Type

Door lock Operation Type	Description	Assigned Number
ZW_DOOR_OP_CONST	Constant operation.	1
ZW_DOOR_OP_TIMED	Timed operation.	2

Table 54 – dev_cfg_error_t Structure

Attribute	Type	I/O	Description
dev_ent	unsigned	O	Device entry number (starting from 1)
ep_ent	unsigned	O	Endpoint entry number (starting from 1)
if_ent	unsigned	O	Interface entry number (starting from 1)

4.2 zwdev_cfg_free

Free device-specific configuration records.

Table 55 – zwdev_cfg_free Parameters

Attribute	Type	I/O	Description
records	dev_rec_t *	I	Device specific configuration records
record_cnt	int	I	Number of records stored in "records" array

4.3 zwdev_global_sett_free

Free device global settings.

Table 56 – zwdev_global_sett_free Parameters

Attribute	Type	I/O	Description
global_sett	dev_global_sett_t *	I	Device setting records
global_sett_cnt	uint16_t	I	Number of records stored in " global_sett" array

4.4 zwdev_cfg_find

Search for a match in device-specific configuration records.

Table 57 – zwdev_cfg_find Parameters

Attribute	Type	I/O	Description
srch_key	dev_rec_srch_key_t *	I	Search key
records	dev_rec_t *	I	Device-specific configuration records sorted according to manufacturer ID, product type ID, and product ID.
record_cnt	int	I	Number of records stored in "records" array.
matched_rec	dev_rec_t *	O	The matched record; either exact match or partial match as explained in the note below. **Note**: This function supports "don't care" cases in device-specific configuration records. The search priority is as follows (in the format (Manf ID, Product Type, Product ID)) : (V, V, V), (V, V, X), (V, X, X), (X, X, X) where V="valid value" and X="don't care".
Return	int	O	Non-zero if a match is found; else returns zero.

Table 58 – dev_rec_srch_key_t Structure

Attribute	Type	I/O	Description
vid	uint32_t	O	Vendor or Manufacturer ID
ptype	uint32_t	O	Product Type ID
pid	uint32_t	O	Product ID

5 Network API

The network is seen through the eyes of a ZIPGW attached controller.

5.1 Discovering ZIPGWs

5.1.1 zwnet_gw_discvr_start

This call starts a network scanning for ZIPGW IP addresses using the ZIPGW Discovery protocol. Once the scanning is done, the result will be available through a callback function. Take note that this function can be called without calling the zwnet_init function first.

Table 59 – zwnet_gw_discvr_start Parameters

Attribute	Type	I/O	Description
cb	zwnet_gw_discvr_cb_t	I	Callback function when the scanning has completed.
usr_param	void *	I	User-defined parameter used in callback function.
ipv4	int	I	Flag to indicate whether to use IPv4 as transport IP protocol. 1= use IPv4; 0= use IPv6.
use_mdns	int	I	Flag to indicate whether to use MDNS for gateway discovery. Note: MDNS gateway discovery is only supported in ZIPGW version 2.
return	void *	O	Context on success, NULL on failure. Caller is required to call zwnet_gw_discvr_stop() with the returned context if it is not null.

Table 60 – zwnet_gw_discvr_cb_t Parameters

Attribute	Type	I/O	Description
gw_addr	uint8_t *	I	Gateway addresses. If the ipv4 flag is set, each gateway address is 4-bytes long; otherwise, each gateway address is 16-bytes long.
gw_cnt	uint8_t	I	Number of gateway addresses returned in gw_addr.
ipv4	int	I	Flag to indicate the gw_addr parameter is IPv4 or IPv6. 1=IPv4; 0=IPv6.
usr_param	void *	I	User-defined parameter passed when calling zwnet_gw_discvr_start().
rpt_num	int	I	Report number that this callback is delivering the gateway addresses report; start from 1.
total_rpt	int	I	Total reports that will be delivered by callbacks. Each callback delivers one report. Zero is returned if there is no valid IP to facilitate gateway discovery.
gw_name	char * *	I	Gateway names corresponding to the gw_addr. If NULL, it means gateway name information is unavailable.

5.1.2 zwnet_gw_discvr_stop

This call stops the network scanning for ZIPGW IP addresses and frees the resources used in network scanning. Take note that this function should not be called in the callback function that was passed to the zwnet_gw_discvr_start function as parameter cb.

Table 61 – zwnet_gw_discvr_stop Parameters

Attribute	Type	I/O	Description
ctx	void *	I	The context returned from the call to zwnet_gw_discvr_start().
return	int	O	Zero on success, non-zero on failure.

5.2 Network Initialization and Clean up

5.2.1 zwnet_init

This call runs a state-machine to acquire the ZIPGW attached controller's Home ID, Node ID, HAN address, and node list of the HAN. An internal network data structure is created and initialized with each of the node IDs found in the acquired node list. A user application could get access to the controller Home ID and Node ID by calling zwnet_get_desc API only after the zwnet_notify_fn callback function returns status is ZW_ERR_NONE.

To populate the internal network data structure with endpoints and interfaces, this API tries to retrieve the node information from an internally maintained database. For those nodes found in the controller routing table but without corresponding node information in the database, the node information state-machine is invoked to get the information (CCs supported, CC version, node name and location, manufacturer ID, product type ID, product ID and multi-instance/channel endpoints, etc.) directly from the node device.

Table 62 – zwnet_init Parameters

Attribute	Type	I/O	Description
init	zwnet_init_t *	I	User filled initialization information.
net	zwnet_t **	O	Handle to network for use in other zwnet_xxx API calls.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 63 – zwnet_init_t Structure

Attribute	Type	I/O	Description
user	void *	I	User context used in callbacks.
host_port	uint16_t	I	Host listening and sending port.
use_ipv4	Int	I	Flag to indicate whether to use IPv4 as transport IP protocol. 1=use IPv4; 0=use IPv6.
zip_router	uint8_t[16]	I	ZIPGW (gateway) IPv4/IPv6 address in numeric format.
notify	zwnet_notify_fn	I	Network operation notification.
node	zwnet_node_fn	I	Node add/delete/status callback.
appl_tx	zwnet_appl_fn	I	Application transmit data status callback.

inif_cb	pl_info_fn	I	Unsolicited included node information frame (INIF) callback when a Smart Start device (which has joined a foreign network but is listed in the local provisioning list) is powering up.
unhandled_cmd	zwnet_unhandled_cmd_fn	I	Unhandled command callback.
print_txt_fn	print_fn	I	Print text function pointer.
portal_fd#	int	I	Network file descriptor to connect to ZIPGW using TLS.
portal_ssl#	void *	I	SSL object pointer to connect to ZIPGW using TLS.
display_ctx#	void *	I	Display context for the print_txt_fn.
net_err#	zwnet_net_err_fn	I	Unrecoverable network error callback, application should close this instance of Z-Ware object, i.e. call zwnet_exit().
portal_prof#	clnt_prof_t	I	Profile of the ZIPGW that is connected to the portal.
net_info_dir	const char *	I	Full path of directory for storing network and node information file (a.k.a. network persistent storage). The file is generated by the library with filename in the format “nifXXXXXXXX.dat”, where XXXXXXXX denotes Z-Wave network home ID.
pref_dir	const char *	I	Full path of directory for storing network/user preference files.
dev_cfg_file	const char *	I	Device-specific configurations file (a.k.a device database) in JSON format. If it is NULL, device-specific configurations will be managed by user application. In this case, *dev_cfg_usr must* be valid.
cmd_cls_cfg_file	const char *	I	Optional CC configuration file. Enable specific CC probing after a new node inclusion and during background polling. If NULL, ALL supported CCs are enabled. The current supported CC configuration file is distributed in the “config” folder of the source distribution with the name “cmd_class.cfg”.
dev_cfg_usr	dev_cfg_usr_t *	I	Device-specific configurations (managed by user application). If it is NULL, device-specific configurations will be managed by Z-Ware Library internally. In this case, *dev_cfg_file* must be valid.
err_loc	dev_cfg_error_t	O	Error location while parsing device-specific configuration file. Note: The error is not due to JSON format parsing error.
sup_cmd_cls	sup_cmd_cls_t *	I	User application implemented CCs.NOTE: if the controller has already implemented the CC, the user's request for that CC will be ignored.
sup_cmd_cls_cnt	uint8_t	I	User application implemented CCs count.
s2_unsolicited_cb	add_node_sec2_fn	I	Callback to report unsolicited joining device requested keys and/or status of Device Specific Key (DSK)

dtls_psk*	uint8_t[32]	I	DTLS pre-shared key in binary format, MUST be at least 16 bytes.
dtls_psk_len*	uint8_t	I	DTLS pre-shared key length (bytes). If length is zero, no DTLS will be used, i.e., communication will be insecured.

Note: Attributes marked with (#) are only available for the Portal version; whereas those marked with (*) are only available for the CE version.

Typical zwnet operations take in the network handle returned from this call and return the success status of the operation as show in the table below. The parameters for such operations are not further documented.

Table 64 – dev_cfg_usr_t Structure

Attribute	Type	I/O	Description
dev_rec_find_fn	zwnet_dev_rec_find_fn	O	User-supplied function to find device record
dev_cfg_ctx	void *	O	User-specified device configuration context for use in dev_rec_find_fn
dev_glob_sett	dev_global_sett_t *	O	Optional device global settings
global_sett_cnt	uint16_t	O	Global settings count in dev_glob_sett

Table 65 – zwnet_dev_rec_find_fn Parameters

Attribute	Type	I/O	Description
dev_cfg_ctx	void *	I	User-specified device configuration context.
srch_key	dev_rec_srch_key_t *	I	Search key.
matched_rec	dev_rec_t *	O	The matched record; either exact match or partial match as explained in the note below. **Note**: This function supports "don't care" cases in device-specific configuration records. The search priority is as follows (in the format (Manf ID, Product Type, Product ID)): (V, V, V), (V, V, X), (V, X, X), (X, X, X) where V="valid value"; X="don't care".
return	int	O	Non-zero if a match is found; else returns zero.

Table 66 – dev_global_sett_t Structure

Attribute	Type	I/O	Description	
type	uint16_t	O	Setting type, GLOB_SET_TYPE_XXX:	
			XXX	Description
			WKUP_INTV	Wakeup interval to set for a newly added sleeping node.
sett	union	O	Settings as indicated by setting type; union of the following:	
wkup_intv	uint32_t		For GLOB_SET_TYPE_WKUP_INTV	

Table 67 – zwnet_xxx Generic Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init
return	int	O	ZW_ERR_XXX

Table 68 – zwnet_unhandled_cmd_fn Parameters

Attribute	Type	I/O	Description
user	void *	I	User context
src_node	uint8_t	I	Source node ID
src_ep	uint8_t	I	Source endpoint ID
cmd	uint8_t *	I	Buffer that stores the unhandled command
cmd_len	Uint16_t	I	Length of the unhandled command

Table 69 – sup_cmd_cls_t Structure

Attribute	Type	I/O	Description	
cls	uint16_t	O	CC	
ver	uint8_t	O	Version of the CC	
propty	uint8_t	O	Properties of the interface (bit-mask): BITMASK_CMD_CLS_XXX	
			BITMASK_CMD_CLS_INSECURE	CC is insecure
			BITMASK_CMD_CLS_SECURE	CC is secure

Table 70 – zwnet_notify_fn Parameters

Attribute	Type	I/O	Description	
user	void *	I	from zwnet_init	
op	uint8_t	I	operation ZWNET_OP_XXX	
			XXX	Description
			NONE	Idle.
			INITIALIZE	Initializing local Z/IP controller.
			ADD_NODE	Adding a node.
			RM_NODE	Removing a node.
			RP_NODE	Replacing a failed node.
			RM_FAILED_ID	Removing a failed node.
			INITIATE	Initiating in response to or anticipation of some operation.
			UPDATE	Updating network topology from SUC/SIS.
			RESET	Restoring ZIPGW attached controller to factory default settings.
			NODE_UPDATE	Updating node information.
			MIGRATE	Migrating primary controller.
			SEND_NIF	Sending node information frame.
			NW_CHANGED	Network change detection.
			NODE_CACHE_UPT	Update node cached info. (Internally used; application won't receive this notification.)

<table>
<tr><td></td><td></td><td></td><td>SAVE_NW</td><td colspan="2">Save network and node information to persistent storage. (Internally used; application won’t receive this notification.)</td></tr>
<tr><td></td><td></td><td></td><td>SLP_NODE_POLL</td><td colspan="2">Sleeping device node information polling. (Internally used; application won’t receive this notification.)</td></tr>
<tr><td></td><td></td><td></td><td>FW_UPDT</td><td colspan="2">Firmware update.</td></tr>
<tr><td></td><td></td><td></td><td>HEALTH_CHK</td><td colspan="2">Network health check.</td></tr>
<tr><td></td><td></td><td></td><td>NODE_RESET</td><td colspan="2">Remove node which has been reset.</td></tr>
<tr><td></td><td></td><td></td><td>FW_DOWNLD</td><td colspan="2">Firmware download</td></tr>
<tr><td>sts</td><td>uint16_t</td><td>I</td><td colspan="3">Status of current operation. When upper-byte is zero, the lower byte is the current status of the operation (see table below). When upper-byte is non-zero, it represents the total number of nodes to get detailed node information, while the lower-byte represents the number of nodes that have completed getting detailed node information.</td></tr>
<tr><td></td><td></td><td></td><td>Operation: ZWNET_OP_</td><td>Status OP_</td><td>Description</td></tr>
<tr><td></td><td></td><td></td><td>All</td><td>DONE</td><td>Operation completed.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>FAILED</td><td>Operation failed.</td></tr>
<tr><td></td><td></td><td></td><td>ADD_NODE</td><td>ADD_NODE_PROTOCOL_DONE</td><td>Protocol part done.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>ADD_NODE_GET_NODE_INFO</td><td>Getting node detailed information.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>ADD_NODE_PROTOCOL_START</td><td>Smart Start add node Z-Wave protocol started.</td></tr>
<tr><td></td><td></td><td></td><td>RP_NODE</td><td>RP_NODE_PROTOCOL_DONE</td><td>Protocol part done.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>RP_NODE_GET_NODE_INFO</td><td>Getting node detailed information.</td></tr>
<tr><td></td><td></td><td></td><td>INITIATE</td><td>INI_PROTOCOL_DONE</td><td>Protocol part done.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>INI_GET_NODE_INFO</td><td>Getting node detailed information.</td></tr>
<tr><td></td><td></td><td></td><td>UPDATE</td><td>NU_TOPOLOGY</td><td>Network topology update started.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>NU_NEIGHBOR</td><td>Node neighbor update started.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>NU_GET_NODE_INFO</td><td>Node information update started.</td></tr>
<tr><td></td><td></td><td></td><td>FW_UPDT</td><td>FW_UPLOAD_STARTED</td><td>Uploading firmware to device started.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>FW_UPLOADING</td><td>Uploading firmware to device in progress.</td></tr>
<tr><td></td><td></td><td></td><td>HEALTH_CHK</td><td>HEALTH_CHK_STARTED</td><td>Network health check started.</td></tr>
</table>

				HEALTH_CHK_PROGRESS	Network health check in progress.
				HEALTH_CHK_CMPLT	Network health check completed.
			FW_DOWNLD	FW_DOWNLD_STARTED	Downloading firmware from device (for backup purposes) has started.
				FW_DOWNLOADING	Downloading firmware from device in progress.
info	zwnet_sts_t *	I	Additional information for the specified op and sts; NULL if there is no additional info. For a cross reference of this additional information to the network op and sts, see table "Additional Info Associated to Network Op and Status"		
ret	int	I	ZW_ERR_XXX.		

Table 71 – zwnet_sts_t Structure

Attribute	Type	I/O	Description	
type	int	O	Type of info, ZWNET_STS_INFO_XXX. See the description in "info" below.	
info	union	O	Union of the following as indicated by "type":	
			Description	For type = ZWNET_STS_INFO_XXX
s2_dsk	char []		The S2 DSK associated with the Smart Start adding node protocol started status.	SS_START
progress_percent	uint8_t		Firmware update progress in %	FW_UPDT_PRG
health_chk_progress	zw_health_prg_t		Network health check progress	HEALTH_CHK_PRG
health_chk_rpt	zw_health_rpt_t		Network health check completion report	HEALTH_CHK_RPT
node_id	uint8_t		Node ID involved in the corresponding network operation	NODE_ID
nw_change_updt_sts	uint8_t		Network change/update status, NW_CHG_UPDT_XXX	NW_CHANGE_UPDT
progress_bytes	uint32_t		Firmware download progress in number of bytes downloaded	FW_DL_PRG

Table 72 – NW_CHG_UPDT_XXX

Network change/update status	Description

NW_CHG_UPDT_NONE	No change in network id and controller
NW_CHG_UPDT_NW_ID	Network id has changed. Application needs to refresh the network structure
NW_CHG_UPDT_CTLR	Z-Wave controller has changed. Application needs to call zwnet_get_desc API to refresh controller's properties.

Table 73 – zw_health_prg_t Structure

Attribute	Type	I/O	Description
node_cnt	uint8_t	O	Number of health check completed nodes
total	uint8_t	O	Total number of nodes scheduled for health check

Table 74 – zw_health_rpt_t Structure

Attribute	Type	I/O	Description
sts_cnt	uint8_t	O	Total number of node health status in the array
sts	zw_health_sts_t []	O	Node health status array

Table 75 – zw_health_sts_t Structure

Attribute	Type	I/O	Description	
node_id	uint8_t	O	Node ID	
sts_cat	uint8_t	O	Status category derived from network health value:	
			NW_HEALTH_XXX	Description
			GREEN	Network health is good.
			YELLOW	Network health is acceptable, but latency can be observed occasionally.
			RED	Network health is insufficient because frames are dropped.
			CRITICAL	Network health is critical because Z-Wave node is not responding at all.
value	uint8_t	O	Calculated network health value.	

Table 76 – Additional Information Associated to Network Op and Status

Additional Information Type, ZWNET_STS_INFO_XXX	Network Operation, ZWNET_OP_XXX	Status OP_XXX
SS_START	ADD_NODE	ADD_NODE_PROTOCOL_START
FW_UPDT_PRG	FW_UPDT	FW_UPLOADING
HEALTH_CHK_PRG	HEALTH_CHK	HEALTH_CHK_PROGRESS
HEALTH_CHK_RPT	HEALTH_CHK	HEALTH_CHK_CMPLT
NODE_ID	NODE_RESET, RM_NODE, RM_FAILED_ID	DONE, FAILED

	ADD_NODE, RP_NODE, NODE_UPDATE	DONE
NW_CHANGE_UPDT	NW_CHANGED, UPDATE	DONE
FW_DL_PRG	FW_DOWNLD	FW_DOWNLOADING

Table 77 – zwnet_node_fn Parameters

Attribute	Type	I/O	Description	
user	void *	I	From zwnet_init	
node	zwnoded_t *	I	Node descriptor of the node that was added, removed, or updated	
status	uint8_t	I	Status, ZWNET_NODE_XXX	
			ZWNET_NODE_XXX	Description
			ADDED	Node was added. Application can retrieve detailed node information now.
			REMOVED	Node was removed.
			UPDATED	Node was updated. Application can retrieve updated detailed node information now.
			STATUS_ALIVE	Node status has changed to "alive"
			STATUS_DOWN	Node status has changed to "down"
			STATUS_SLEEP	Node status has changed to "sleeping"
			RESET	Received notification from node that it has been reset

Table 78 – zwnet_appl_fn Parameters

Attribute	Type	I/O	Description	
user	void *	I	From zwnet_init	
tx_sts	uint8_t	I	Transmission status ZWNET_TX_XXX	
			OK	Successful.
			NO_ACK	Send frame timeout due to no ACK received.
			SYS_ERR	System error, the program should exit or restart.
			DEST_BUSY	Message has not timed out yet. The destination host may have a long response time (e.g. sleeping node).
			NOROUTE	Frame failed to reach destination host.

Table 79 – pl_info_fn Parameters

Attribute	Type	I/O	Description
usr_ctx	void *	I	User context
lst_ent	struct pl_lst_ent *	I	Provisioning list entry

Table 80 – zwnet_net_err_fn Parameters

Attribute	Type	I/O	Description
user	void *	I	User context which was passed to zwnet_init

Table 81 – print_fn Parameter

Attribute	Type	I/O	Description
msg	void *	I	Pointer to text messages

Table 82 – zwnet_dev_rec_find_fn Parameter

Attribute	Type	I/O	Description
dev_cfg_ctx	void *	I	User-specified device configuration context.
vid	uint16_t	I	Vendor or Manufacturer ID.
ptype	uint16_t	I	Product Type ID.
pid	uint16_t	I	Product ID.
dev_rec	dev_rec_t *	O	Device record found.
return	int	O	Non-zero on found; zero on not found.

5.2.2 zwnet_exit

Clean up network and save detailed node information into file.

Table 83 – zwnet_exit Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init

5.2.3 zwnet_reset

Reset the state of the ZIPGW attached controller, losing all network information. Upon completion, it will get the controller's node information.

5.3 Network Creation

5.3.1 zwnet_add

Called by the inclusion controller to add/remove an initiating node to/from the network. On successful addition, the new node is queried for its endpoints and interfaces, secure and otherwise. If the interfaces are type sensor, the new node will be configured based on either the settings passed during network initialization or the default values. It should result in callbacks to zwnet_node_fn followed by zwnet_notify_fn.

Table 84 – zwnet_add Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init.

add	uint8_t	I	Operation 0 – Remove. 1 – Add.
sec2_param	sec2_add_prm_t *	I	Parameters for adding node with Security 2 protocol. This parameter is ignored when non-security 2 ZIPGW is detected or when removing a node.
incl_on_behalf	int	I	Flag to indicate enter into inclusion on-behalf (iob) mode; 1=enter into iob mode, 0=normal add node. This parameter is ignored when parameter add = 0 (i.e. remove node).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 85 – sec2_add_prm_t Structure

Attribute	Type	I/O	Description
usr_param	void *	O	User-defined parameter used in callback.
cb	add_node_sec2_fn	O	Callback to report joining device requested keys and/or status of Device Specific Key (DSK).
dsk	char *	O	Optional Device Specific Key (DSK) of the joining node for verification. Note that, if this is not NULL, no DSK callback will be executed. The format of the DSK must be as shown in the example: 34028-23669-20938-46346-33746-07431-56821-14553

Table 86 – add_node_sec2_fn Parameters

Attribute	Type	I/O	Description
usr_param	void *	I	User-supplied parameter when calling zwnet_add or zwnet_fail
cb_param	sec2_add_cb_prm_t *	I	DSK related callback parameters

Table 87 – sec2_add_cb_prm_t Structure

Attribute	Type	I/O	Description	
cb_type	uint8_t	O	Callback type as in S2_CB_TYPE_XXX :	
			XXX	Description
			REQ_KEY	Joining device requested keys
			DSK	Joining device DSK keys
req_key	sec2_keys_req_cb_prm_t	O	For cb_type=S2_CB_TYPE_REQ_KEY; the joining device requested keys and CSA request	
dsk	sec2_dsk_cb_prm_t	O	For cb_type=S2_CB_TYPE_DSK; the joining device DSK keys	

Table 88 – sec2_keys_req_cb_prm_t Structure

Attribute	Type	I/O	Description
req_keys	uint8_t	O	Requested keys (bit mask) by the joining node.

req_csa	uint8_t	O	Flag to indicate joining node is requesting Client-side Authentication (CSA).
csa_pin	char [12]	O	CSA 10-digit PIN to be entered into the joining node if req_csa is non-zero. Example: 34028-23669 (Note: The hyphen is for display purposes, it is not part of the PIN.)

Table 89 – sec2_dsk_cb_prm_t Structure

Attribute	Type	I/O	Description
pin_required	uint8_t	O	Indicate whether the user is required to enter five-digit pin. 1=required; 0=not required.
dsk	char *	O	Device Specific Key (DSK) of the joining node for user to verify. Note that the first five digits of DSK are missing if pin_required=1. Example of complete DSK: 34028-23669-20938-46346-33746-07431-56821-14553 Example of DSK with first five digits missing: -23669-20938-46346-33746-07431-56821-14553

5.3.2 Secure Inclusion

5.3.2.1 zwnet_add_sec2_accept

Accept or reject newly added node into Security 2 mode. This function should be called only after receiving Security 2 DSK callback during add or replace failed node operation.

Table 90 – zwnet_add_sec2_accept Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
accept	int	I	1=accept; 0=reject.
dsk	char *	I	Complete Device Specific Key (DSK) of the added node if accept=1; else this can be NULL.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.3.2.2 zwnet_add_sec2_grant_key

Grant keys and CSA to the newly added node in Security 2 mode. This function should be called only after receiving Security 2 requested keys callback during add or replace failed node operation. Note that it is possible to not grant any key and yet to accept S2 bootstrapping, i.e. granted_keys=0 and accept_s2=1.

Table 91 – zwnet_add_sec2_grant_key Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle

granted_keys	uint8_t	I	Bit mask of zero or more security 2 granted keys as in SEC_KEY_BITMSK_XXX XXX: Description S2_K0: S2: Class key 0 (Unauthenticated devices) S2_K1: S2: Class key 1 (Authenticated devices) S2_K2: S2: Class key 2 (Access control devices) S0: S0: Legacy security network key
grant_csa	uint8_t	I	Grant client-side authentication (CSA); 1=grant, 0=reject. If joining node didn't request CSA, it is ignored.
accept_s2	uint8_t	I	Acceptance of S2 bootstrapping process; 1=S2 bootstrapping is accepted and MUST continue, 0=S2 bootstrapping is not accepted and MUST be interrupted
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

5.3.3 SmartStart Provisioning

5.3.3.1 zwnet_pl_add

Add a provisioning list entry.

Table 92 – zwnet_pl_add Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
dsk	char *	I	Device Specific Key (DSK). The format of the DSK must be eight groups of five digits separated by '-' as shown in the example: 34028-23669-20938-46346-33746-07431-56821-14553
info	pl_info_t *	I	Buffer to store additional information of the device (optional). Information type PL_INFO_TYPE_NW_STS must not be used in this function.
info_cnt	uint8_t	I	Number of additional information stored in "info".
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 93 – pl_info_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Information type, PL_INFO_TYPE_XXX XXX: Description PROD_TYPE: Product type PROD_ID: Product ID INC_INTV: Smart Start inclusion request interval used by the node UUID16: UUID assigned to the node NAME: Device name LOC: Device location INCL_STS: Inclusion status of the provisioning list entry

			S2_GRNT	S2 keys to be granted
			BOOT_MODE	Bootstrapping mode
			NW_STS	Network status of the provisioning list entry
info	union	O	Provisioning list entry information. The union of the following:	
name	char [63]	O	For type PL_INFO_TYPE_NAME; device name string in UTF-8 encoding which must be compliant with the following rules: • The combined "name" and "loc" strings MUST NOT be longer than 62 bytes (excluding NUL terminating char) • The escape sequence "\." (backslash followed by dot) must be used for encoding the dot character '.' • MUST NOT contain the underscore character '_' • MUST NOT end with the dash character '-' • MUST be case insensitive	
loc	char [63]	O	For type PL_INFO_TYPE_LOC; device location string in UTF-8 encoding with same restrictions as "name" field.	
prod_type	pl_prod_type_t	O	For type PL_INFO_TYPE_PROD_TYPE; product type.	
prod_id	pl_prod_id_t	O	For type PL_INFO_TYPE_PROD_ID; product ID.	
interval	uint8_t	O	For type PL_INFO_TYPE_INC_INTV; Smart Start inclusion request interval in unit of 128 seconds. This field must have value ranging from 5 to 99.	
uuid	pl_uuid_t	O	For type PL_INFO_TYPE_UUID16; UUID assigned to the node as defined by IETF RFC 4122.	
nw_sts	pl_nw_sts_t	O	For type PL_INFO_TYPE_NW_STS; network status of the provisioning list entry.	
incl_sts	uint8_t	O	For type PL_INFO_TYPE_INCL_STS; inclusion status of the provisioning list entry, PL_INCL_STS_XXX.	
			XXX	Description
			PENDING	The node will be included in the network when it issues SmartStart inclusion requests.
			PASSIVE	The node is in the Provisioning List, but it has been decided by the supporting or controlling node that the node is unlikely to issue SmartStart inclusion requests in the near future. SmartStart Inclusion requests will be ignored by the Z/IP Gateway. All entries with this status MUST be updated to the “Pending” status (PL_INCL_STS_PENDING) when a Provisioning List Iteration Get Command is received.
			IGNORED	SmartStart inclusion requests sent by the node in the Provisioning List entry will be ignored until the status is changed again by a Z/IP Client or controlling node.

grnt_keys	uint8_t	O	For type PL_INFO_TYPE_S2_GRNT; S2 keys to be granted, see SEC_KEY_BITMSK_XXX.	
boot_mode	uint8_t	O	For type PL_INFO_TYPE_BOOT_MODE; Bootstrapping mode, as in PL_BOOT_MODE_XXX.	
			XXX	Description
			S2	The node MUST manually be set to Learn Mode and follow the S2 bootstrapping instructions (if any).
			SMART_STRT	The node will be included and S2 bootstrapped automatically using the Smart Start functionality.

Table 94 – pl_prod_type_t Structure

Attribute	Type	I/O	Description
generic_cls	uint8_t	O	Generic device class
specific_cls	uint8_t	O	Specific device class
icon_type	uint16_t	O	Installer icon type

Table 95 – pl_prod_id_t Structure

Attribute	Type	I/O	Description
manf_id	uint16_t	O	Manufacturer ID
prod_type	uint16_t	O	Product type
prod_id	uint16_t	O	Product ID
app_ver	uint8_t	O	Application version
app_sub_ver	uint8_t	O	Application sub version

Table 96 – pl_uuid_t Structure

Attribute	Type	I/O	Description	
pres_fmt	uint16_t	O	Presentation format, as in UUID_PRES_XXX.	
			XXX	Description
			32HEX	32 hex digits, no delimiters
			16ASCII	16 ASCII chars, no delimiters
			SN_32HEX	"sn:" followed by 32 hex digits, no delimiters
			SN_16ASCII	"sn:" followed by 16 ASCII chars, no delimiters
			UUID_32HEX	"UUID:" followed by 32 hex digits, no delimiters
			UUID_16ASCII	"UUID:" followed by 16 ASCII chars, no delimiters
			RFC4122	RFC4122 compliant presentation (e.g., “58D5E212-165B-4CA0-909B-C86B9CEE0111”)
uuid	uint8_t[16]	O	UUID assigned to the node as defined by IETF RFC 4122.	

Table 97 – pl_nw_sts_t Structure

Attribute	Type	I/O	Description	
node_id	uint8_t	O	NodeID that has been assigned to the Provisioning List entry during network inclusion. 0 indicates that the NodeID is not assigned.	
status	uint8_t	O	network status of the Provisioning List entry, as in PL_NW_STS_XXX.	
			XXX	Description
			NOT_INCL	The node in the Provisioning List is not currently included (added) in the network.
			ADDED	The node in the Provisioning List is included in the network and is functional.
			FAILED	The node in the Provisioning List has been included in the Z-Wave network but is now marked as failing.

5.3.3.2 zwnet_pl_get

Get a provisioning list entry information through callback.

Table 98 – zwnet_pl_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
dsk	char *	I	Device Specific Key (DSK). The format of the DSK must be eight groups of five digits separated by '-' as shown in the example: 34028-23669-20938-46346-33746-07431-56821-14553
cb	pl_info_fn	I	Callback to report provisioning entry information.
usr_ctx	void *	I	User context used in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 99 – pl_info_fn Parameters

Attribute	Type	I/O	Description
usr_ctx	void *	I	User context
lst_ent	pl_lst_ent_t *	I	Provisioning list entry

Table 100 – pl_lst_ent_t Structure

Attribute	Type	I/O	Description
dsk	char [48]	O	Device Specific Key (DSK). The format of the DSK must be eight groups of five digits separated by '-' as shown in the example: 34028-23669-20938-46346-33746-07431-56821-14553
info	pl_info_t[10]	O	Information of the device.
info_cnt	uint8_t	O	Number of information stored in "info". If zero, the dsk is not found in the provisioning list.

5.3.3.3 zwnet_pl_del

Delete a provisioning list entry.

Table 101 – zwnet_pl_del Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
dsk	char *	I	Device Specific Key (DSK). The format of the DSK must be eight groups of five digits separated by '-' as shown in the example: 34028-23669-20938-46346-33746-07431-56821-14553
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.3.3.4 zwnet_pl_list_get

Get all provisioning entries through callback.

Table 102 – zwnet_pl_list_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
cb	pl_list_fn	I	Callback to report provisioning list entries.
usr_ctx	void *	I	User context used in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 103 – pl_list_fn Parameters

Attribute	Type	I/O	Description
usr_ctx	void *	I	User context.
pl_list	pl_lst_ent_t *	I	Provisioning list.
ent_cnt	uint8_t	I	Number of entries in provisioning list 'pl_list'.
lst_cmplt	int	I	Flag to indicate whether all the entries in 'pl_list' represent the whole list in ZIPGW. 1=all the entries have been retrieved from ZIPGW; 0=some of the entries were not retrieved due to error.

5.3.3.5 zwnet_pl_list_del

Delete all provisioning list entries.

Table 104 – zwnet_pl_list_del Parameters

Attribute	Type	I/O	Description
Net	zwnet_t *	I	Network handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.4 Network Management

5.4.1 zwnet_initiate

Indicates willingness to add/remove itself to/from the network or become the new primary. This action is paired with a corresponding zwnet_add or zwnet_migrate from another controller. Upon completion, it will rebuild its internal network structure if it has been added into a new network.

Table 105 – zwnet_initiate Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
cb	get_dsk_fn	I	Callback to report ZIPGW DSK when joining another S2 capable Z-Wave network. Note that the callback will not be invoked if the ZIPGW is not S2 capable or this API is invoked to leave a Z-Wave network.
usr_ctx	void *	I	User context used in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.4.2 zwnet_fail

Called by the inclusion controller to replace/remove a failed node in the network. When replaced, information on the node will be refreshed.

Table 106 – zwnet_fail Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
nodeid	uint8_t	I	Failed node ID. This parameter is ignored when rplc_on_behalf = 1.
replace	uint8_t	I	Operation 0 – Remove. 1 – Replace with a new initiating node.
sec2_param	sec2_add_prm_t *	I	Optional parameters for replacing node with Security 2 protocol. This parameter is ignored when removing failed node (i.e. replace = 0).
rplc_on_behalf	int	I	Flag to indicate enter into "replace failed node" on-behalf (rob) mode; 1=enter into rob mode, 0=normal replace failed node. This parameter is ignored when parameter replace = 0.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.4.3 zwnet_update

This function updates network topology, routing table, and internal network data structures. As it may take several iterations to complete if there are nodes that are out of direct range, calling of this function should be scheduled at a time of low network activity.

5.4.4 zwnet_abort

Abort current operation. The user should be aware that aborting certain network operations, such as add, replace, or joining/leaving a Z-Wave network depends on the stage at which the operation is aborted and hence the operation may not be undone completely. For example, once the callback function, zwnet_notify_fn, delivers the status of ADD_NODE_PROTOCOL_DONE, RP_NODE_PROTOCOL_DONE, or INI_PROTOCOL_DONE (for joining a network), the device has already been added to the Z-Wave network, and aborting the operation at this moment still results in the node being added. To undo the operation completely, the user needs to execute the remove node operation manually. Similarly, for leaving a Z-Wave network, if the callback function, zwnet_notify_fn, delivers the status of INI_PROTOCOL_DONE, the Z/IP gateway attached controller has already been removed from the Z-Wave network, and aborting the operation at this moment is considered too late. The user has to manually execute the API zwnet_initiate command to rejoin the Z-Wave network.

5.5 Network Attributes and Traversal

5.5.1 zwnet_get_desc

Get the read-only network descriptor for node enumeration and other purposes.

Table 107 – zwnet_get_desc Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	network handle from zwnet_init
return	zwnetd_t *	O	network descriptor

Table 108 – zwnetd_t Structure

Attribute	Type	I/O	Description	
id	uint32_t	O	Z-Wave Home ID	
ctl_id	uint8_t	O	Z/IP controller node ID	
ctl_role	uint8_t	O	Z/IP controller role. A bit-mask of ZWNET_CTLR_ROLE_XXX :	
			ZWNET_CTLR_ROLE_PROXY	Support Network Management Proxy
			ZWNET_CTLR_ROLE_INCL	Support Network Management Inclusion
			ZWNET_CTLR_ROLE_PRI	Support Network Management Primary
ctl_cap	uint8_t	O	Z/IP controller capabilities. A bit-mask of ZWNET_CTLR_CAP_XXX:	
			ZWNET_CTLR_CAP_S2	Support Security 2 protocol
			ZWNET_CTLR_CAP_INC_ON_BEHALF	Support inclusion on-behalf
			ZWNET_CTLR_CAP_SMART_START	Support Smart Start
			ZWNET_CTLR_CAP_IMA	Support IMA (Z-Wave Network Installation and maintenance)
			ZWNET_CTLR_CAP_MULTICAST	Support multicast addressing

			ZWNET_CTLR_CAP_IDENTIFY	Capable to identify itself (e.g. blinking a LED)
ctl_zw_role	uint8_t	O	Z/IP controller Z-Wave role, ZW_ROLE_XXX:	
			ZW_ROLE_UNKNOWN	Unknown
			ZW_ROLE_SIS	SIS
			ZW_ROLE_INCLUSION	Inclusion controller
			ZW_ROLE_PRIMARY	Primary controller
			ZW_ROLE_SECONDARY	Secondary controller
user	void *	O	User context which was passed to zwnet_init	
plt_ctx	void *	O	Platform context for printing of output text messages	

5.5.2 zwnet_version

Get the home controller API version, subversion and patch version.

Table 109 – zwnet_version Parameters

Attribute	Type	I/O	Description
ver	uint8_t *	O	Version
subver	uint8_t *	O	Sub version
patch_ver	uint8_t *	O	Patch version

5.5.3 zwnet_get_node

Get the first node (local controller) in the network.

Table 110 – zwnet_get_node Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle
noded	zwnoded_t *	O	Node handle
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

5.5.4 zwnet_get_node_ by_id

Get node with specified node ID in the network.

Table 111 – zwnet_get_node_by_id Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
nodeid	uint8_t	I	Node ID.
noded	zwnoded_t *	O	Node descriptor (Can be NULL, if the purpose is to verify the existence of a node).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.5.5 zwnet_get_ep_ by_id

Get endpoint with specified node and endpoint ID in the network.

Table 112 – zwnet_get_ep_by_id Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle
nodeid	uint8_t	I	Node ID
epid	uint8_t	I	Endpoint ID
epd	zwepd_t *	O	Endpoint descriptor (Can be NULL, if the purpose is to verify the existence of an endpoint).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.5.6 zwnet_get_if_ by_id

Get interface with specified node, endpoint and interface id in the network.

Table 113 – zwnet_get_if_by_id Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle
nodeid	uint8_t	I	Node ID.
epid	uint8_t	I	Endpoint ID.
cls	uint16_t	I	Interface ID (CC).
ifd	zwifd_t *	O	Interface descriptor (Can be NULL, if the purpose is to verify the existence of an interface).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.5.7 zwnet_all_node_sts_get

Get all node status.

Table 114 – zwnet_all_node_sts_get Parameters

Attribute	Type	I/O	Description	
net	zwnet_t *	I	Network handle	
node_sts_buf	uint8_t *	O	Buffer (min. size of 233 bytes) to store all the node status. Individual node status (ZWNET_NODE_STS_XXX) can be accessed using the node ID as index to the buffer.	
			ZWNET_NODE_STS_SLEEP	Node is sleeping.
			ZWNET_NODE_STS_DOWN	Node is down.
			ZWNET_NODE_STS_UP	Node is alive.

5.5.8 zwnet_node_sts_get

Get node status.

Table 115 – zwnet_node_sts_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
node_id	uint8_t	I	Node ID (ranging from 1 to 232).
return	uint8_t	O	Node status (ZWNET_NODE_STS_XXX).

5.6 Advanced Network APIs

These are APIs that are not required for normal operation but are provided for special purposes.

5.6.1 zwnet_migrate

This API is called by the primary controller to transfer its primary controller role to another initiating controller. The initiating controller must not be currently part of this network if it runs on older firmware, and a new node will be created with similar effect as with zwnet_add. With current firmware, the initiating controller can be a secondary controller in the network; in this case, no new node is created. Upon completion, the controller invoking this function will become the secondary controller. It will then get node information from the new primary controller and resolve the primary controller node ID into the IPv6 address.

5.6.2 zwnet_initiate_classic

This API is provided mainly for backward compatibility with older Z-Wave controllers that support only classic learn mode. For description and usage, please refer to the previous section, zwnet_initiate.

Table 116 – zwnet_initiate_classic Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
cb	get_dsk_fn	I	Callback to report ZIPGW DSK when joining another S2 capable Z-Wave network. Note that the callback will not be invoked if the ZIPGW is not S2 capable or this API is invoked to leave a Z-Wave network.
usr_ctx	void *	I	User context used in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.6.3 zwnet_health_chk

Start network health check on all but sleeping nodes. This is part of the Installation and Maintenance (IMA) utility function. It is recommended that the network health check be run during the period when there are few or no network activities running as the process may take a long time to complete.

5.6.4 zwnet_identify

Instruct the controller to identify itself (for example, by blinking an LED in a specific manner). This API is used to support Z-Wave Plus v2 node (including Z/IP gateway controller node), which must have a

visible LED to be used for an Identify function. If the node is itself a light source, e.g., a light bulb, this may be used in place of a dedicated LED.

5.6.5 zwnet_get_user

Get the user context associated with the network handle.

Table 117 – zwnet_get_user Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init.
return	void *	O	User context that was passed as parameter when calling zwnet_init.

5.6.6 zwnet_send_nif

Send node information frame (NIF) to a node or broadcast to the network.

Table 118 – zwnet_send_nif Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
noded	zwnoded_t *	I	Handle of the destination node that will receive the node information frame. NULL for broadcast of NIF.
broadcast	uint8_t	I	Broadcast flag. 1= broadcast; 0= single cast.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

5.6.7 zwnet_poll_rm

Remove a polling request.

Table 119 – zwnet_poll_rm Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
handle	uint16_t	I	Handle of the polling request to remove.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.6.8 zwnet_poll_rm_mul

Remove multiple polling requests.

Table 120 – zwnet_poll_rm_mul Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
usr_token	uint32_t	I	User token of the polling request to remove.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.6.9 zwnet_pref_set

Store network preference into persistent storage.

Table 121 – zwnet_pref_set Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle
buf	void *	I	Buffer that contains the network preference
buf_size	uint16_t	I	Buffer size in bytes
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

5.6.10 zwnet_pref_get

Retrieve network preference from persistent storage.

Table 122 – zwnet_pref_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
buf	void **	O	Return buffer that contains the network preference.
buf_size	uint16_t *	O	Buffer size in bytes.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX **Note**: Caller MUST free the return buffer "buf" if the call is successful.

5.6.11 zwnet_client_pref_set

Store client preference into persistent storage.

Table 123 – zwnet_client_pref_set Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
client_id	uint32_t	I	User-defined client ID to identify the preference.
buf	void *	I	Buffer that contains the client preference.
buf_size	uint16_t	I	Buffer size in bytes.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

5.6.12 zwnet_client_pref_get

Retrieve client preference from persistent storage.

Table 124 – zwnet_client_pref_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
client_id	uint32_t	I	User-defined client ID to identify the preference.
buf	void **	O	Return buffer that contains the client preference.
buf_size	uint16_t *	O	Buffer size in bytes.

return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX. **Note**: Caller MUST free the return buffer "buf" if the call is successful.

5.6.13 zwnet_sec2_get_dsk

Get ZIPGW Security 2 DSK.

Table 125 – zwnet_sec2_get_dsk Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
cb	get_dsk_fn	I	Callback to report ZIPGW DSK.
usr_ctx	void *	I	User context used in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 126 – get_dsk_fn Parameters

Attribute	Type	I/O	Description
usr_ctx	void *	I	User context
dsk	char *	I	ZIPGW DSK string

5.7 Network Utilities APIs

These are APIs that provide IPv4/6 network utilities unrelated to Z-Wave network to ease programmers who are unfamiliar to the IPv4/6 network C APIs.

5.7.1 zwnet_ip_aton

Utility to convert IPv4 / IPv6 address string to numeric equivalent.

Table 127 – zwnet_ip_aton Parameters

Attribute	Type	I/O	Description
addr_str	char *	I	NULL terminated IPv4 / IPv6 address string.
addr_buf	uint8_t *	O	Buffer that should be at least 16-bytes long to store the result.
ipv4	int *	O	Flag to indicate the converted numeric IP address is IPv4 or IPv6. 1=IPv4; 0=IPv6.
return	int	O	Zero on success; non-zero on failure.

5.7.2 zwnet_ip_ntoa

Utility to convert IPv4 / IPv6 address in numerical form to string equivalent.

Table 128 – zwnet_ip_ntoa Parameters

Attribute	Type	I/O	Description

addr	uint8_t *	I	IPv4 / IPv6 address in numerical form.
addr_str	char *	O	Buffer to store the converted address string.
addr_str_len	int	I	The size of the addr_str in bytes.
ipv4	int	I	Flag to indicate the addr parameter is IPv4 or IPv6. 1=IPv4; 0=IPv6.
return	int	O	Zero on success, non-zero on failure.

5.7.3 zwnet_local_addr_get

Utility to get local address that is reachable by destination host.

Table 129 – zwnet_local_addr_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init.
dest_ip	uint8_t*	I	Destination host address either IPv4 or IPv6 according to use_ipv4 flag
local_ip	uint8_t*	O	Buffer of 16-byte to store the local address (either IPv4 or IPv6 according to use_ipv4 flag).
use_ipv4	int	I	Flag to indicate IP address type. 1= IPv4; 0= IPv6.
return	int	O	Zero on success, non-zero on failure.

5.7.4 zwnet_listen_port_get

Utility to get local Z/IP listening port number.

Table 130 – zwnet_listen_port_get Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle from zwnet_init
return	uint16_t	O	The Z/IP client listening port number.

6 Node API

This corresponds to a Z-Wave device (node) and is coupled with Manufacturer Specific CC and Node Naming and Location CC information. It also contains APIs for sleep command queuing and command recording.

6.1 zwnoded_t

This is the descriptor used to access Z-Wave nodes. Only name and location fields are writeable.

Table 131 – zwnoded_t Structure

Attribute	Type	I/O	Description
nodeid	uint8_t	O	Z-Wave Node ID
propty	uint8_t	O	Properties of the node (bit-mask): NODE_PROPTY_XXX.
Vid	uint16_t	O	Z-Wave Vendor ID.
type	uint16_t	O	Vendor type.
Pid	uint16_t	O	Z-Wave Product ID.
Net	zwnet_t *	O	Network handle.
dev_id	dev_id_t	O	Device ID.
proto_ver	uint16_t	O	Z-Wave Protocol Version.
app_ver	uint16_t	O	Application Version.
lib_type	uint8_t	O	Z-Wave Library Type.
category	uint8_t	O	Device category, DEV_XXX.
sensor	uint8_t	O	Flag to indicate whether the node is a sensor (FLIRS).
sleep_cap	uint8_t	O	Flag to indicate the node is capable to sleep (i.e. non-listening and support Wake up CC).
listen	uint8_t	O	Flag to indicate the node is always listening.
zwsw_ver_cnt	uint8_t	O	Number of valid Z-Wave software version types stored in the zwsw_ver array.
zwsw_ver	zwsw_ver_t[]	O	Z-Wave software version information.
s2_keys_valid	uint8_t	O	Flag to indicate whether s2_grnt_keys is valid.
s2_grnt_keys	uint8_t	O	Security 2 granted keys (bit-mask), see SEC_KEY_BITMSK_XXX NOTE: This is valid only s2_keys_valid = 1.
s2_dsk	char[]	O	S2 DSK. If s2_dsk[0] == '\0', the DSK is unavailable for this node.

Table 132 – NODE_PROPTY_XXX

Node property bit-mask	Description
NODE_PROPTY_SECURE_CAP_S0	Node capable of being included securely S0.
NODE_PROPTY_SECURE_CAP_S2	Node capable of being included securely S2.
NODE_PROPTY_ADD_SECURE	Node is included securely.
NODE_PROPTY_ADD_INSECURE	Node is included insecurely.

NODE_PROPTY_IDENTIFY_CAP	Node is capable of identifying itself (e.g. blinking LED three times).

Table 133 – dev_id_t Structure

Attribute	Type	I/O	Description	
type	uint8_t	O	Device ID type: DEV_ID_TYPE_XXX, where:	
			DEV_ID_TYPE_OEM	OEM factory default
			DEV_ID_TYPE_SN	Serial number
			DEV_ID_TYPE_RANDOM	Pseudo random number
format	uint8_t	O	Device ID data format: DEV_ID_FMT_XXX, where:	
			DEV_ID_FMT_UTF	UTF-8
			DEV_ID_FMT_BIN	Binary and MUST be displayed as hexadecimal, e.g. h'30313233A1
len	uint8_t	O	Device ID length	
dev_id	uint8_t[MAX_DEV_ID_LEN + 1]	O	Device ID	

Table 134 – zwsw_ver_t Structure

Attribute	Type	I/O	Description	
type	uint8_t	O	Version type: VER_TYPE_XXX, where	
			VER_TYPE_SDK	SDK version.
			VER_TYPE_APPL_FRW_API	Application framework API version.
			VER_TYPE_HOST_INTF	Host interface version; the version of the API exposed to a host CPU.
			VER_TYPE_ZWAVE_PROTO	Z-Wave protocol version.
			VER_TYPE_APPL	Application software version.
major	uint8_t	O	Major version.	
minor	uint8_t	O	Minor version.	
patch	uint8_t	O	Patch version.	
build	uint16_t	O	Build number. Zero if unused.	

6.2 zwnode_get_net

Get handle to the network containing this node.

Table 135 – zwnode_get_net Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node handle

return	zwnet_t *	O	Network handle

6.3 zwnode_get_next

Get next node in network. The first, i.e., local controller node may be obtained from zwnet_get_node.

Table 136 – zwnode_get_next Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node handle.
next	zwnoded_t *	O	Next node handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

6.4 zwnode_get_ep

Get first endpoint in the node.

Table 137 – zwnode_get_ep Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node handle.
epd	zwepd_t *	O	Endpoint handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

6.5 zwnode_update

Updates node status and information.

Table 138 – zwnode_update Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

6.6 zwnode_identify

Instruct the node to identify itself. This API is to support Z-Wave Plus v2 node, which must have a visible LED to be used for an Identify function. If the node is itself a light source, e.g., a light bulb, this may be used in place of a dedicated LED.

Table 139 – zwnode_identify Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node handle
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

6.7 zwnode_get_ext_ver

Get extended version information.

Table 140 – zwnode_get_ext_ver Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node descriptor.
return	ext_ver_t *	O	Extended version information if the node supports it; else returns NULL. Note: Caller has to free the returned extended version information.

Table 141 – ext_ver_t Structure

Attribute	Type	I/O	Description
hw_ver	uint8_t	I	Hardware version.
fw_cnt	uint8_t	I	Number of firmwares in the device, excluding the Z-Wave firmware.
fw_ver	uint16_t[]	I	Firmware versions place holder as indicated in fw_cnt.

6.8 Advanced Node APIs

These are APIs that should not be used for normal operation but are provided for debugging purposes.

6.8.1 zwnode_mul_cmd_ctl_set

Turn on/off multi command encapsulation capability. By default, if a node supports Multi-command CC, the multi command encapsulation capability is turned on automatically.

Table 142 – zwnode_mul_cmd_ctl_set Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node descriptor.
on	uint8_t	I	1=turn on multi command encapsulation; 0=turn off and flush (send) the commands in the buffer.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

6.8.2 zwnode_mul_cmd_ctl_get

Get the current state of multi command encapsulation.

Table 143 – zwnode_mul_cmd_ctl_get Parameters

Attribute	Type	I/O	Description
noded	zwnoded_t *	I	Node descriptor.
on	uint8_t *	O	state: 1=on; 0=off.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

7 Endpoint API

This corresponds to Z-Wave channels or instances. The APIs here are mostly accessor functions.

7.1 zwepd_t

This is the read-only descriptor used to access endpoints.

Table 144 – zwepd_t Structure

Attribute	Type	I/O	Description
generic	uint8_t	O	Z-Wave Generic device class
specific	uint8_t	O	Z-Wave Specific device class
epid	uint8_t	O	Endpoint number
nodeid	uint8_t	O	Node ID
aggr_ep_cnt	uint8_t	O	Total number of end point members that are represented by this aggregated end point. Zero means this endpoint is NOT an aggregated end point
aggr_members	uint8_t[126]	O	Buffer to store the end point members that are represented by this aggregated end point
net	zwnet_t *	O	Network handle
name	char [33]	O	User configured name string of the endpoint
loc	char [33]	O	User configured location string of the endpoint
zwplus_info	zwplus_info_t	O	Z-Wave Plus information CC

Table 145 – zwplus_info_t structure

Attribute	Type	I/O	Description
zwplus_ver	uint8_t	O	Version. Zero indicates this node is non-Z-Wave Plus.
node_type	uint8_t	O	Node type.
role_type	uint8_t	O	Role type.
instr_icon	uint16_t	O	Installer icon type.
usr_icon	uint16_t	O	User icon type.

7.2 zwep_get_node

Get containing node handle.

Table 146 – zwep_get_node Parameters

Attribute	Type	I/O	Description
epd	zwepd_t *	I	Endpojnt handle
noded	zwnoded_t *	O	Node handle
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

7.3 zwep_get_next

Get next endpoint in node.

Table 147 – zwep_get_next Parameters

Attribute	Type	I/O	Description
epd	zwepd_t *	I	Endpojnt handle.
nxt_epd	zwepd_t *	O	Next endpoint handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

7.4 zwep_get_if

Get first interface in endpoint.

Table 148 – zwep_get_if Parameters

Attribute	Type	I/O	Description
epd	zwepd_t *	I	Endpojnt handle.
Ifd	zwifd_t *	O	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Note: *Caller must free ifd->data if ifd->data_cnt is greater than zero.*

7.5 zwep_nameloc_set

Set endpoint name and location string for device regardless of whether the endpoint has Node Naming And Location interface. If it does, the strings will be sent to the interface physically for storage in the device.

Table 149 – zwep_nameloc_set Parameters

Attribute	Type	I/O	Description
epd	zwepd_t *	I	Endpojnt handle.
nameloc	zw_nameloc_t *	I	NULL terminated name and location strings.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 150 – zw_nameloc_t Structure

Attribute	Type	I/O	Description
name	char[33]	I	Node name
loc	char[33]	I	Node location

8 Interface API

This corresponds to Z-Wave CCs coupled with the corresponding Version CC information. Some of the CCs, such as security CC, multi-channel CC, and multi-command CC, are handled transparently. Basic CC is added automatically to all endpoints even if it is not listed in the endpoint's "multichannel capability report". Individual CCs are categorized into Network, Management, Transport and Application CCs. All Transport CCs are handled by ZIPGW.

8.1 zwifd_t

This is the read-only descriptor used to access interfaces.

Table 151 – zwifd_t Structure

Attribute	Type	I/O	Description	
cls	uint8_t	O	CC.	
ver	uint8_t	O	CC version. Can be upgraded by device database.	
real_ver	uint8_t	O	Real Version of the CC that the device supports.	
propty	uint8_t	O	Properties of the interface (bit-mask): IF_PROPTY_XXX where:	
			IF_PROPTY_SECURE	Interface can be accessed securely bit-mask.
			IF_PROPTY_UNSECURE	Interface can be accessed insecurely bit-mask.
			IF_PROPTY_HIDDEN	Interface is hidden. NOTE: This property is for internal use only; user application MUST NOT use this.
			IF_PROPTY_HIDDEN_POLL	Interface is hidden but device polling is enabled. NOTE: This property is for internal use only; user application MUST NOT use this.
			IF_PROPTY_ALRM_EVT_CLR	Interface is capable to sent event clear notification. NOTE: This bit-mask is only valid for Alarm/Notification CC.
			IF_PROPTY_BSNSR_EVT_CLR	Interface is capable of sending event clear. NOTE: This bit-mask is only valid for binary sensor CC.
			IF_PROPTY_ALRM_SNSR_EVT_CLR	Interface is capable of sending event clear. NOTE: This bit-mask is only valid for the Alarm Sensor CC.
epid	uint8_t	O	Endpoint number.	
nodeid	uint8_t	O	Node ID.	
net	zwnet_t *	O	Network handle.	

8.2 zwif_get_ep

Get containing endpoint.

Table 152 – zwif_get_ep Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
epd	zwepd_t *	O	Endpoint handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

8.3 zwif_get_next

Get next interface in endpoint.

Table 153 – zwif_get_next Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
nxt_ifd	zwifd_t *	O	Next interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Note: *Caller must free nxt_ifd->data if nxt_ifd->data_cnt is greater than zero.*

8.4 zwif_exec

Execute a command on an interface (intended for automation).

Table 154 – zwif_exec Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cmd	uint8_t *	I	Command packet.
len	int	I	Length of cmd in bytes.
cb	tx_cmplt_cb_t	I	Completion callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

8.5 zwif_xxx_poll

Polling variant of zwif_xxx_get API. This is meant for the interfaces that follow to perform a continuous poll instead of a single get.

Table 155 – zwif_xxx_poll Parameters

Attribute	Type	I/O	Description
…	…	I	Parameters of zwif_xxx.
poll_req	zwpoll_req_t *	I/O	Poll request.

return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 156 – zwpoll_req_t Structure

Attribute	Type	I/O	Description
usr_token	uint32_t	I	User-defined token to facilitate deletion of multiple polling requests.
interval	uint16_t	I	Polling interval in seconds; zero = the smallest possible interval.
poll_cnt	uint16_t	I	Number of times to poll; zero = unlimited times (i.e. repetitive polling).
cmplt_cb	zwpoll_cmplt_fn	I	Polling completion callback. NULL if callback is not required.
usr_param	void *	I	User parameter of polling completion callback.
handle	uint16_t	O	Handle if the request is accepted into the polling queue. The handle can be used to facilitate deletion of the polling request.

Table 157 – zwpoll_cmplt_fn Parameters

Attribute	Type	I/O	Description
net	zwnet_t *	I	Network handle.
handle	uint16_t	I	Handle of the polling request.
usr_token	uint32_t	I	User-defined token to facilitate deletion of multiple polling requests.
usr_param	void *	I	User parameter.

9 Management CCs Based Interfaces

Manufacturer Specific, Node Naming and Location, Z-Wave Plus Info, and Version CCs are handled in the Node & Endpoint APIs.

9.1 Group Interface API

This corresponds to the Association & Multichannel Association CC. Lifelines are automatically setup and other associations can be preconfigured via the Device Database, rather than by using these APIs.

9.1.1 zwif_group_sup_get

Get information on the maximum number of groupings the given node supports through report callback function.

Table 158 – zwif_group_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
cb	zwrep_group_sup_fn	I	Maximum number of groupings report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 159 – zwrep_group_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
max_grp	uint8_t	I	Maximum number of groups.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

9.1.2 zwif_group_actv_get

Get information on the current active group from a node through report callback function.

Table 160 – zwif_group_actv_get Parameters

Attribute	Type	I/O	Description
ifd	zwif_t *	I	Group interface handle.
cb	zwrep_group_actv_fn	I	Current active group report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 161 – zwrep_group_actv_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle

group	uint8_t	I	Current active group

9.1.3 zwif_group_get

Get group members information on specified group through zwrep_group_fn report callback function.

Table 162 – zwif_group_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
group	uint8_t	I	Group ID.
cb	zwrep_group_fn	I	Endpoint list report callback function.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 163 – zwrep_group_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
group	uint8_t	I	Group identifier.
max_cnt	uint8_t	I	Maximum number of members the grouping identifier above supports.
cnt	uint8_t	I	Number of end points in this report.
grp_member	grp_member_t *	I	An array of cnt group members in the grouping. Note that the group members may contain non-existence node/endpoint.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 164 – grp_member_t structure

Attribute	Type	I/O	Description
node_id	uint8_t	I	Node ID
ep_id	uint8_t	I	Endpoint id. Value of 255 denotes node association; other values denote endpoint association.

9.1.4 zwif_group_add

Add endpoints to this group and its containing node's return routes. It is recommended to add all endpoints with one call to ensure return routes are set up correctly.

Table 165 – zwif_group_add Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
group	uint8_t	I	Group ID.
ep	zwepd_t *	I	An array of recipient endpoints to be added into the grouping.
cnt	uint8_t	I	Number of endpoints in the array "ep".

cb	zwif_grp_rr_fn	I	Completion callback function (optional). NULL if callback is not required.
user_prm	void *	I	User parameter for the completion callback function (optional). NULL if not in use.
return	int	O	Zero on success and completion callback will be invoked upon completion of return route assignments; positive return code (>0, e.g. ZW_ERR_QUEUED) will be returned if the command is queued and NO completion callback will be invoked; else negative return code (ZW_ERR_XXX) will be returned on error.

Table 166 – zwif_grp_rr_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group interface handle.
status	uint8_t	I	status, RRA_XXX
user_prm	void *	I	User parameter

Table 167 – RRA_XXX

Return route assignment status	Description
RRA_OK	Successfully assigned return routes
RRA_TX_NO_ACK	No acknowledgement of transmission is received before timeout
RRA_TX_FAIL	Not possible to transmit data because the Z-Wave network is busy
RRA_FAIL	Operation failed

9.1.5 zwif_group_del

Remove endpoints from this group.

Table 168 – zwif_group_del Parameters

Attribute	Type	I/O	Description
ifd	zwif_t *	I	Group interface handle.
group	uint8_t	I	Group ID.
grp_member	grp_member_t *	I	An array of cnt members to be removed from the grouping. Node association should be indicated by ep_id=255.
cnt	uint8_t	I	The number of members in the array grp_member. If cnt is zero, all the members in the group may be removed.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.1.6 zwif_group_info_get

Get detailed group information.

Table 169 – zwif_group_info_get Parameters

Attribute	Type	I/O	Description
ifd	zwif_t *	I	Group interface handle.
grp_info	if_grp_info_dat_t **	O	Grouping information if success; NULL on failure.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 170 – if_grp_info_dat_t Structure

Attribute	Type	I/O	Description
group_cnt	uint16_t	I	Number of supported groups.
valid_grp_cnt	uint16_t	I	Number of valid group information in grp_info[].
dynamic	uint8_t	I	Flag to indicate the group information is dynamic. 1=dynamic; 0=static.
grp_info	zw_grp_info_t *[]	I	Place holder for storing pointers to group information.

Table 171 – zw_grp_info_t Structure

Attribute	Type	I/O	Description
grp_num	uint8_t	I	Group number
cmd_ent_cnt	uint8_t	I	Number of entries in command list (cmd_lst)
profile	uint16_t	I	Profile
evt_code	uint16_t	I	Event code
name	char [43]	I	NULL terminated group name string in UTF-8
cmd_lst	grp_cmd_ent_t []	I	Place holder for command list

Table 172 – grp_cmd_ent_t Structure

Attribute	Type	I/O	Description
cls	uint16_t	I	CC
cmd	uint8_t	I	Command

9.1.7 zwif_group_info_free

Free group information.

Table 173 – zwif_group_info_free Parameters

Attribute	Type	I/O	Description
grp_info	if_grp_info_dat_t *	I	Grouping information returned by zwif_group_info_get()

9.2 Group Command Interface API

This corresponds to the Z-Wave association command configuration CC used to specify commands in groups within a node.

9.2.1 zwif_group_cmd_sup_get

Get information on command records supporting capabilities through report callback function.

Table 174 – zwif_group_cmd_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group command configuration interface handle.
cb	zwrep_grp_cmd_sup_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 175 – zwrep_grp_cmd_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group command configuration interface handle.
cmd_cap	zwgrp_cmd_cap_t *	I	Command records supporting capabilities.

Table 176 – zwgrp_cmd_cap_t structure

Attribute	Type	I/O	Description
configurable	uint8_t	O	1=command record is configurable, 0=not configurable type.
config_type	uint8_t	O	Configuration type: 1=value type (only support Basic Set command). 0=command type (support any command).
max_len	uint8_t	O	Maximum command length that can be set.
free_rec	uint16_t	O	Number of free command records that can be set.
total_rec	uint16_t	O	Total number of command records supported.

9.2.2 zwif_group_cmd_get

Get command record for a node within a given grouping identifier through report callback function.

Table 177 – zwif_group_cmd_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group command configuration interface handle.
group	uint8_t	I	Grouping identifier.
nodeid	uint8_t	I	Node ID of the node within the grouping specified.
cb	zwrep_grp_cmd_fn	I	Report callback function.
Return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 178 – zwrep_grp_cmd_fn Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Group command configuration interface handle.
group	uint8_t	I	Grouping identifier.
nodeid	uint8_t	I	Node ID of the node within the grouping specified.
cmd_buf	uint8_t *	I	Command and parameters.
len	uint8_t	I	Length of cmd_buf.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.2.3 zwif_group_cmd_set

Specify which command should be sent to a node within a given group.

Table 179 – zwif_group_cmd_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Group command configuration interface handle.
group	uint8_t	I	Grouping identifier.
node	uint8_t	I	Node within the grouping specified that should receive the command specified in cmd_buf.
cmd_buf	uint8_t *	I	Command and parameters.
len	uint8_t	I	Length of cmd_buf.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.3 Battery Interface API

This corresponds to the Z-Wave battery CC.

9.3.1 zwif_battery_rpt_set

Setup a battery report callback function. Note that the report battery level 0x00..0x64 indicates the battery percentage level from 0 to 100%. Level 0xFF indicates a low-battery warning.

Table 180 – zwif_battery_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Battery interface handle.
rpt_cb	zwrep_batt_lvl_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 181 – zwrep_batt_lvl_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Battery interface handle.
data	zwbatt_dat_t *	I	Battery Level and status.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 182 – zwbatt_dat_t Structure

Attribute	Type	I/O	Description
level	uint8_t	O	Battery level 0x00..0x64 MUST indicate the battery percentage level from 0 to 100%. Level 0xFF MUST indicate a low-battery warning.
sts_valid	uint8_t	O	Indicate whether the following battery status fields are valid. 1=valid; 0=invalid.
Battery status			
rechargeable	uint8_t	O	Flag to indicate if the battery is rechargeable. 1=rechargeable; 0=non-rechargeable
charge_sts	uint8_t	O	Charging status if battery is rechargeable, BATT_STS_XXX
bkup_batt	uint8_t	O	Flag to indicate if the battery is utilized for back-up purposes of a main powered connected device. 1=backup; 0=battery is used for primary means of power
overheat	uint8_t	O	Flag to indicate if overheating is detected. 1=overheating; 0=operating within the normal temperature range
low_fluid	uint8_t	O	Flag to indicate if the battery fluid is low and should be refilled. 1=low; 0=normal
rechg_req	uint8_t	O	Status to indicate if the battery needs to be recharged (for rechargeable battery) or replaced (for non-rechargeable battery), BATT_RECHG_XXX
disconnect	uint8_t	O	Flag to indicate if the battery is currently disconnected or removed from the node. 1=disconnected and the node is running on an alternative power source; 0=connected

Table 183 – BATT_STS_XXX

Battery charging status	Description
BATT_STS_DISCHARGE	Discharging
BATT_STS_CHARGE	Charging
BATT_STS_MAINTAIN	Maintaining

Table 184 – BATT_RECHG_XXX

Battery recharge or replace request	Description
BATT_RECHG_NONE	No recharge or replace battery is needed
BATT_RECHG_SOON	Recharge or replace battery soon
BATT_RECHG_NOW	Recharge or replace battery now

9.3.2 zwif_battery_get

Get battery level through report callback function.

Table 185 – zwif_battery_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Battery interface handle.

flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.3.3 zwif_battery_health_rpt_set

Set up a battery health report callback function.

Table 186 – zwif_battery_health_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_batt_health_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 187 – zwrep_batt_health_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
data	zwbatt_health_t *	I	Battery health status
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 188 – zwbatt_health_t Structure

Attribute	Type	I/O	Description
max_cap	uint8_t	O	The percentage indicating the maximum capacity of the battery. Values in the range 0x00..0x64 indicate the maximum capacity of the battery in the percentage level from 0 to 100%. Value of 0xFF indicates the maximum capacity of the battery is unknown
precision	uint8_t	O	Decimal places of the battery temperature value. E.g. the decimal value 1025 with precision 2 is equal to 10.25
unit	uint8_t	O	Unit used in battery temperature. 0=Celsius; other values are reserved
size	uint8_t	O	Battery temperature data size: 0, 1, 2 or 4 bytes. Value 0 indicates the battery temperature is unknown
data	uint8_t [4]	O	Battery temperature data (a signed number) with the first byte is the most significant byte

9.3.4 zwif_battery_health_get

Get battery health report through report callback.

Table 189 – zwif_battery_health_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.

return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.4 Time and Date Interface API

This corresponds to the Z-Wave Time CC.

9.4.1 zwif_time_rpt_set

Set up a current local time report callback function.

Table 190 – zwif_time_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_time_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 191 – zwrep_time_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
hour	uint8_t	I	Hour (in 24 hours format).
minute	uint8_t	I	Minute.
second	uint8_t	I	Second.
rtc_fail	int	I	Flag to indicate if RTC oscillator has been stopped and hence the advertised time might be inaccurate. 1=stopped; 0=running or node does not support this feature.

9.4.2 zwif_time_get

Get current local time through report callback.

Table 192 – zwif_time_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.4.3 zwif_date_rpt_set

Set up a current date report callback function.

Table 193 – zwif_date_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_date_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 194 – zwrep_date_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle
year	uint16_t	I	Year (example: 2018)
month	uint8_t	I	Month
day	uint8_t	I	Day

9.4.4 zwif_date_get

Get current date adjusted according to the local time zone and Daylight Saving Time through report callback.

Table 195 – zwif_date_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.4.5 zwif_tz_dst_rpt_set

Set up a time zone and Daylight Saving Time information report callback function.

Table 196 – zwif_tz_dst_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_tz_dst_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 197 – zwrep_tz_dst_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tz_info	tmzone_info_t *	I	Time zone info.
dst_info	dst_info_t *	I	Daylight Saving Time info.

Table 198 – tmzone_info_t Structure

Attribute	Type	I/O	Description
sign	int	O	Sign (plus or minus) to apply to the hour and minute fields. 0 = Plus sign (positive offset from UTC); 1 = Minus sign (negative offset from UTC).
hour	uint8_t	O	Hour offset.
minute	uint8_t	O	Minute offset.

Table 199 – dst_info_t Structure

Attribute	Type	I/O	Description

sign	int	O	Sign (plus or minus) for the minute_offset field to apply to the current time while in the Daylight Saving Time. 0 = Plus sign (positive offset from current time); 1 = Minus sign (negative offset from current time).
minute_offset	uint8_t	O	Minute offset.
month_start	uint8_t	O	Month of the year when Daylight Saving Time starts. Range 1..12 (representing respectively January...December).
day_start	uint8_t	O	Day of the month when DST starts. Range 1..31.
hour_start	uint8_t	O	Hour of the day when DST starts. Range 0..23.
month_end	uint8_t	O	Month of the year when DST ends. Range 1..12 (representing respectively January...December).
day_end	uint8_t	O	Day of the month when DST ends. Range 1..31.
hour_end	uint8_t	O	Hour of the day when DST ends. Range 0..23.

9.4.6 zwif_tz_dst_get

Get time zone and Daylight Saving Time information through report callback.

Table 200 – zwif_tz_dst_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.5 Firmware Update Interface API

This corresponds to the Firmware Update Meta Data CC.

9.5.1 zwif_fw_info_get

Get firmware information through report callback.

Table 201 – zwif_fw_info_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
cb	zwrep_fw_info_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 202 – zwrep_fw_info_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
fw_info	zwfw_info_t *	I	firmware information.

Table 203 – zwfw_info_t structure

Attribute	Type	I/O	Description
vid	uint16_t	I	Vendor/Manufacturer ID.
zw_fw_id	uint16_t	I	Z-Wave firmware ID.
chksum	uint16_t	I	CRC-CCITT checksum of Z-Wave firmware.
max_frag_sz	uint16_t	I	Maximum meta data fragment size for firmware update.
fixed_frag_sz	uint8_t	I	Flag to indicate whether the max_frag_sz is fixed, i.e. firmware update request MUST use the given size. 1=fixed size; 0=variable size.
upgrade_flg	uint8_t	I	Firmware upgradable flag. 0= Firmware is not upgradable; 0xFF= Firmware is upgradable.
hw_ver_valid	uint8_t	I	Flag to indicate whether the hw_ver is valid. 1=valid; 0=invalid.
hw_ver	uint8_t	I	Hardware version.
func_normally	uint8_t	I	To indicate whether other CCs function normally during firmware update image transfer. 2=function normally; 1=certain CCs will not function; 0=this information is unavailable
activation	uint8_t	I	To indicate whether node supports the subsequent activation of firmware after downloading. 2=support; 1=not support; 0=this information is unavailable
other_fw_cnt	uint8_t	I	Number of other firmware IDs.
other_fw_id	uint16_t *	I	Pointer to array of other firmware IDs with the count given by other_fw_cnt.

9.5.2 zwif_fw_updt_req

Request firmware update. Caller should call zwif_fw_info_get() first before calling this function.

Table 204 – zwif_fw_updt_req Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
req	zwfw_updt_req_t *	I	Firmware update request.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 205 – zwfw_updt_req_t Structure

Attribute	Type	I/O	Description
vid	uint16_t	I	Vendor/Manufacturer ID the firmware is intended for.
fw_id	uint16_t	I	Firmware ID the firmware is intended for.
hw_ver	uint8_t	I	Hardware version the firmware is intended for; zero if inapplicable.
fw_tgt	uint8_t	I	Firmware target to update. 0= Z-Wave firmware, 0x01 to 0xFF for firmware target returned by zwif_fw_info_get(). For example, to update firmware target in other_fw_id[0], use fw_tgt=1.

delay	uint8_t	I	If supported, ask the target device to delay firmware update until "activation" command is received. 1=delay, user needs to call zwif_fw_updt_actv() to activate the firmware; 0=no delay, i.e. activate the firmware immediately after download.
fw_file	const char *	I	Firmware file path. For Intel hex file format, the extension must be ".hex", ".ihex", ".otz" or ".ota", other extension will be treated as binary file.
sts_cb	zwrep_fw_updt_sts_fn	I	Firmware update request status callback function. See ZW_FW_UPDT_ERR_XXX.
cmplt_cb	zwrep_fw_updt_cmplt_fn	I	Firmware update request completion status callback function.
restart_cb	zwrep_fw_tgt_restart_fn	I	Optional: Firmware update target restart completion callback function. Can be NULL if callback is not required. Note that the callback function works only if the cmplt_cb() invoked with parameter wait_tm_valid=1.

Table 206 – zwrep_fw_updt_sts_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
status	uint8_t	I	Firmware update status, ZW_FW_UPDT_ERR_XXX.

Table 207 – ZW_FW_UPDT_ERR_XXX

Firmware update request status	Description
ZW_FW_UPDT_ERR_INVALID	Invalid combination of vendor ID and firmware ID.
ZW_FW_UPDT_ERR_AUTHEN	Need out-of-band authentication event to enable firmware update.
ZW_FW_UPDT_ERR_FRAG_SZ	The requested Fragment Size exceeds the Max Fragment Size.
ZW_FW_UPDT_ERR_UPGRD	This firmware target is not upgradable.
ZW_FW_UPDT_ERR_HW_VER	Invalid hardware version.
ZW_FW_UPDT_ERR_TRANF_IN_PROG	Another firmware image is current being transferred
ZW_FW_UPDT_ERR_LOW_BATT	Insufficient battery level to complete the firmware update operation
ZW_FW_UPDT_ERR_VALID	Valid combination of vendor ID and firmware ID; the device will start to request firmware data from the requester.

Table 208 – zwrep_fw_updt_cmplt_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
status	uint8_t	I	Firmware update completion status, ZW_FW_UPDT_CMPLT_XXX.

wait_tm	uint16_t	I	Time (in seconds) that is needed before the receiving node becomes available again for communication after the transfer of an image. This parameter is valid only if wait_tm_valid=1.
wait_tm_valid	int	I	Flag to indicate the wait_tm parameter is valid.

Table 209 – ZW_FW_UPDT_CMPLT_XXX

Firmware update completion status	Description
ZW_FW_UPDT_CMPLT_ERR_CHKS	Checksum error in requested firmware.
ZW_FW_UPDT_CMPLT_ERR_DOWNLOAD	Download of the requested firmware failed.
ZW_FW_UPDT_CMPLT_ERR_VID	Vendor/Manufacturer ID mismatched.
ZW_FW_UPDT_CMPLT_ERR_FW_ID	Firmware ID mismatched.
ZW_FW_UPDT_CMPLT_ERR_FW_TGT	Firmware target mismatched.
ZW_FW_UPDT_CMPLT_ERR_FILE_HDR	Invalid file header information.
ZW_FW_UPDT_CMPLT_ERR_FILE_H_FMT	Invalid file header format.
ZW_FW_UPDT_CMPLT_ERR_MEM	Out of memory.
ZW_FW_UPDT_CMPLT_ERR_HW_VER	Hardware version mismatched.
ZW_FW_UPDT_CMPLT_ERR_BATT_LOW	Battery level is low, firmware update was not initiated
ZW_FW_UPDT_CMPLT_ERR_BATT_UNKNW	Battery level is unknown, firmware update was not initiated
ZW_FW_UPDT_CMPLT_OK_WAIT	Image downloaded, waiting for activation command.
ZW_FW_UPDT_CMPLT_OK_NO_RESTART	New image was successfully stored in temporary non-volatile memory. The device does not restart itself.
ZW_FW_UPDT_CMPLT_OK_RESTART	New image was successfully stored in temporary non-volatile memory. The device will now start storing the new image in primary non-volatile memory dedicated to executable code. Then the device will restart itself.

Table 210 – zwrep_fw_tgt_restart_fn Parameters

Attribute	Type	I/O	Description
node	zwnoded_t *	I	Node.
status	uint8_t	I	Firmware update target restart status, ZW_FW_UPDT_RESTART_XXX.

Table 211 – ZW_FW_UPDT_RESTART_XXX

Firmware update completion status	Description
ZW_FW_UPDT_RESTART_OK	Restart o.k.
ZW_FW_UPDT_RESTART_FAILED	Restart failed due to no response from the target.

9.5.3 zwif_fw_updt_actv

Initiate the programming of a previously transferred firmware image.

Table 212 – zwif_fw_updt_actv Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
cb	zwrep_fw_actv_fn	I	Firmware update status callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 213 – zwrep_fw_actv_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
sts	zwfw_actv_sts_t *	I	Firmwares update status information

Table 214 – zwfw_actv_sts_t Structure

Attribute	Type	I/O	Description
vid	uint16_t	I	Vendor/Manufacturer ID.
fw_id	uint16_t	I	Firmware ID.
chksum	uint16_t	I	CRC-CCITT checksum of the firmware
fw_tgt	uint8_t	I	Firmware target to update. 0= Z-Wave firmware, 0x01 to 0xFF for other firmware
status	uint8_t	I	Firmware update status, ZW_FW_ACTV_STS_XXX
hw_ver	uint8_t	I	Hardware version. Value of zero indicates the hardware version is unavailable

Table 215 – ZW_FW_ACTV_STS_XXX

Firmware update activation status	Description
ZW_FW_ACTV_STS_INVALID	Invalid combination of vendor id, firmware id and hardware version or firmware target
ZW_FW_ACTV_STS_ERROR	Error activating the firmware. Original firmware image has been restored
ZW_FW_ACTV_STS_OK	Firmware update completed successfully

9.5.4 zwif_fw_downld_req

Request for firmware download from a device into a file for backup purposes. Caller should call zwif_fw_info_get() first before calling this function.

Table 216 – zwif_fw_downld_req Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
req	zwfw_downld_req_t *	I	Firmware download request.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 217 – zwfw_downld_req_t Structure

Attribute	Type	I/O	Description
vid	uint16_t	I	Vendor/Manufacturer ID the firmware is intended for.
fw_id	uint16_t	I	Firmware ID the firmware is intended for.
fw_tgt	uint8_t	I	Firmware target to download. 0= Z-Wave firmware, 0x01 to 0xFF for firmware target returned by zwif_fw_info_get(). For example, to download firmware target in **other_fw_id[0]**, use fw_tgt=1
hw_ver	uint8_t	I	Hardware version the firmware is intended for.
fw_file	const char *	I	Firmware file path for storing the downloaded binary data.
sts_cb	zwfw_downld_sts_fn	I	Firmware download request status callback function.
cmplt_cb	zwfw_downld_cmplt_fn	I	Firmware download completion status callback function. This function will only be invoked if download request status callback sts_cb() reported ZW_FW_DL_RQ_OK

Table 218 – zwfw_downld_sts_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
status	uint8_t	I	Firmware download request status, ZW_FW_DL_RQ_XXX.

Table 219 – ZW_FW_DL_RQ_XXX

Firmware download request status	Description
ZW_FW_DL_RQ_ERR_INVALID_ID	Error. Invalid combination of vendor/manufacturer id and firmware id
ZW_FW_DL_RQ_ERR_AUTHEN	Error. Device expected an authentication event to enable firmware update
ZW_FW_DL_RQ_ERR_FRAG_SZ	Error. The requested Fragment Size exceeds the Max Fragment Size
ZW_FW_DL_RQ_ERR_DOWNLOAD	Error. This firmware target is not downloadable
ZW_FW_DL_RQ_ERR_HW_VER	Error. Invalid hardware version
ZW_FW_DL_RQ_OK	OK. The receiving node can initiate the firmware download of the target

Table 220 – zwfw_downld_cmplt_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Firmware update interface handle.
status	uint8_t	I	Firmware download completion status, ZW_FW_DL_CMPLT_XXX

Table 221 – ZW_FW_DL_CMPLT_XXX

Firmware download completion status	Description
ZW_FW_DL_CMPLT_ERR_CHKSM	Checksum error in downloaded firmware
ZW_FW_DL_CMPLT_ERR_DOWNLOAD	Download of the firmware failed
ZW_FW_DL_CMPLT_ERR_MEM	Out of memory
ZW_FW_DL_CMPLT_ERR_REQ_STATUS	Firmware download request status failed, downloading not started
ZW_FW_DL_CMPLT_ERR_REQ_TMOUT	Firmware download request timeout, downloading not started
ZW_FW_DL_CMPLT_ERR_FILE_WR	Writing firmware file failed
ZW_FW_DL_CMPLT_ERR_TOO_LARGE	The firmware size is too large
ZW_FW_DL_CMPLT_ERR_OTHER	Other error
ZW_FW_DL_CMPLT_OK	OK. Firmware downloaded and saved to file

9.6 Indicator Interface API

This corresponds to the Indicator CC.

9.6.1 zwif_ind_rpt_set

Setup an indicator report callback function.

Table 222 – zwif_ind_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
rpt_cb	zwrep_ind_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 223 – zwrep_ind_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
val_v1	uint8_t	I	Value of version 1 indicator. This value must be IGNORED if "ind_data" is not NULL. The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can take values from 1 to 99 (0x01 - 0x63).
ind_data	zwind_data_t *	I	Indicator data. NULL if the report contains no indicator object.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 224 – zwind_data_t Structure

Attribute	Type	I/O	Description
ind_id	uint8_t	O	Indicator ID, ZWIND_ID_XXX.
propty_val_cnt	uint8_t	O	Number of property-values in the array pointed by "propty_val".
propty_val	zwind_propty_val_t *	O	Array of property-values.

Table 225 – zwind_propty_val_t Structure

Attribute	Type	I/O	Description
propty_id	uint8_t	O	Property ID, ZWIND_PPTY_ID_XXX.
val	uint8_t	O	Value of the property.

Table 226 – ZWIND_ID_XXX

Indicator ID	Description
ZWIND_ID_ARMED	Use to indicate that alarm is armed (RECOMMENDED color: Red).
ZWIND_ID_NOT_ARMED	Use to indicate that alarm is disarmed (RECOMMENDED color: Green).
ZWIND_ID_READY	Use to indicate that device is ready (RECOMMENDED color: Green).
ZWIND_ID_FAULT	Use to indicate a general error (RECOMMENDED color: Red).
ZWIND_ID_BUSY	Use to indicate that device is temporarily busy (RECOMMENDED color: Yellow).
ZWIND_ID_ENTER_ID	Use to signal that device is waiting for ID (RECOMMENDED color: Yellow).
ZWIND_ID_ENTER_PIN	Use to signal that device is waiting for PIN code (RECOMMENDED color: Yellow).
ZWIND_ID_COD_ACCPT	Use to indicate OK e.g. the entered code is accepted (RECOMMENDED color: Green).
ZWIND_ID_COD_NOT_ACCPT	Use to indicate NOT OK e.g. the entered code is NOT accepted (RECOMMENDED color: Red)
ZWIND_ID_ARMED_STAY	Use to indicate that the alarm is armed and users are staying in the home
ZWIND_ID_ARMED_AWAY	Use to indicate that the alarm is armed and users are away from the home
ZWIND_ID_ALARM	Use to indicate that the alarm is triggered and active with no reason specified
ZWIND_ID_ALARM_BURGLAR	Use to indicate that the alarm is triggered and active due to a Burglar event
ZWIND_ID_ALARM_FIRE	Use to indicate that the alarm is triggered and active due to a Fire Alarm event
ZWIND_ID_ALARM_CO	Use to indicate that the alarm is triggered and active due to Carbon Monoxide event

ZWIND_ID_BYPASS_CHALLGE	Use to indicate that the device expects a bypass challenge code
ZWIND_ID_ENTRY_DELAY	Use to indicate that the alarm is about to be activated unless disarmed by a code
ZWIND_ID_EXIT_DELAY	Use to indicate that the alarm will be active after the exit delay
ZWIND_ID_ZONE1_ARMED	Use to indicate that alarm Zone 1 is armed (RECOMMENDED color: Red).
ZWIND_ID_ZONE2_ARMED	Use to indicate that alarm Zone 2 is armed (RECOMMENDED color: Red).
ZWIND_ID_ZONE3_ARMED	Use to indicate that alarm Zone 3 is armed (RECOMMENDED color: Red).
ZWIND_ID_ZONE4_ARMED	Use to indicate that alarm Zone 4 is armed (RECOMMENDED color: Red).
ZWIND_ID_ZONE5_ARMED	Use to indicate that alarm Zone 5 is armed (RECOMMENDED color: Red).
ZWIND_ID_ZONE6_ARMED	Use to indicate that alarm Zone 6 is armed (RECOMMENDED color: Red).
ZWIND_ID_LCD_BKLIGHT	Use to turn on LCD backlight, e.g. to shortly draw attention when alarm is activated from another entry control keypad.
ZWIND_ID_BTN_BKLIGHT_LETTER	Use to indicate that buttons are ready for user input (letter).
ZWIND_ID_BTN_BKLIGHT_DIGIT	Use to indicate that buttons are ready for user input (digit).
ZWIND_ID_BTN_BKLIGHT_CMD	Use to indicate that buttons are ready for user input (command).
ZWIND_ID_BTN_1	Use to draw attention to Button 1(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_2	Use to draw attention to Button 2(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_3	Use to draw attention to Button 3(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_4	Use to draw attention to Button 4(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_5	Use to draw attention to Button 5(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_6	Use to draw attention to Button 6(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_7	Use to draw attention to Button 7(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_8	Use to draw attention to Button 8(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_9	Use to draw attention to Button 9(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_10	Use to draw attention to Button 10(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_BTN_11	Use to draw attention to Button 11(RECOMMENDED for button backlight or LED next to button).

ZWIND_ID_BTN_12	Use to draw attention to Button 12(RECOMMENDED for button backlight or LED next to button).
ZWIND_ID_IDENTIFY	This indicator MUST be a visible LED used to identify the node.
ZWIND_ID_BUZZER	Use to draw attention or provide user feedback.

Table 227 – ZWIND_PPTY_ID_XXX

Indicator property ID	Description
ZWIND_PPTY_ID_MLEVEL	Indicating a specific level. Example: Light level, sound level. Value: 0x00=OFF; 0x01..0x63 indicates lowest non-zero level to 100%; 0xFF=restore the most recent (non-zero) level.
ZWIND_PPTY_ID_BIN	Turning the indicator On or Off. Value: 0x00=OFF; 0x01..0x63 indicates ON; 0xFF=ON.
ZWIND_PPTY_ID_TGGL_PERD	Toggling period. This property is used to set the duration in tenth of seconds of an On/Off period. Value: 0x00..0xFF ranging from 0 to 25.5 seconds. If this property ID is specified, ZWIND_PPTY_ID_TGGL_CYCLE MUST also be specified.
ZWIND_PPTY_ID_TGGL_CYCLE	Toggling cycle. This property is used to set the number of On/Off periods to run. Value: 0x00..0xFE ranging from 0 to 254 times. Note that user must not set this property to value of 3 which is reserved for indicator ZWIND_ID_IDENTIFY.
ZWIND_PPTY_ID_TGGL_ON_TIME	On time within a period. This property is used to set the length of the On time in tenth of seconds during an On/Off period. It allows asymetic On/Off periods. Value: 0x00=symmetric On/Off period (On time equal to Off time); 0x01..0xFF = ON time ranging from 0.1 to 25.5 seconds.
ZWIND_PPTY_ID_TMOUT_MIN	This property is used to set a timeout to the indicator. The indicator will return to OFF / Mute after this time. Values in the range 0x00..0xFF MUST represent 0. . 255 minutes This value MUST be ignored if Toggling is defined (0x03 and 0x04 property IDs defined). If ALL property IDs within the Timeout Property Group are set to 0x00, the indicator will not time out automatically.
ZWIND_PPTY_ID_TMOUT_SEC	This property is used to set a timeout to the indicator. The indicator will return to OFF / Mute after this time. Values in the range 0x00..0x3B MUST represent 0. . 59 seconds
ZWIND_PPTY_ID_TMOUT_100TH_S	This property is used to set a timeout to the indicator. The indicator will return to OFF / Mute after this time. Values in the range 0x00..0x63 MUST represent 0.00 to 0.99 seconds
ZWIND_PPTY_ID_MLEVEL_SOUND	This property is used to set the volume of a indicator. Value 0x00 MUST represent Off/Mute in volume Values 0x01 .. 0x64 MUST represent 1 .. 100% Value 0xFF MUST restore the most recent (non-zero) level

ZWIND_PPTY_ID_LOW_PWR	ADVERTISE ONLY: This property MUST NOT be used in a controlling Command (SET). This property MAY be used by a supporting node to advertise that the indicator can continue working even if it is in sleep mode.

9.6.2 zwif_ind_get

Get indicator value through report callback.

Table 228 – zwif_ind_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
ind_id	uint8_t	I	Indicator ID, ZWIND_ID_XXX. Zero if unused (i.e. version 1 indicator).
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.6.3 zwif_ind_set

Set indicators value.

Table 229 – zwif_ind_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
val_v1	uint8_t	I	Value of version 1 indicator. This value will be IGNORED if "ind_cnt" is non-zero. The value can be either 0x00 (off/disable) or 0xFF (on/enable). Furthermore it can take values from 1 to 99 (0x01 - 0x63).
ind_cnt	uint8_t	I	Number of indicator data in the array "ind_data".
ind_data	zwind_data_t *	I	Array of indicator data. Each indicator data must use different indicator ID.
return	int	O	ZW_ERR_NONE on success; ZW_ERR_TOO_LARGE if indicator data (ind_data) is too large to fit into the transport layer payload; else ZW_ERR_XXX.

9.6.4 zwif_ind_sup_get

Get all supported indicators information through report callback.

Table 230 – zwif_ind_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
cb	zwrep_ind_sup_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 231 – zwrep_ind_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
ind_cnt	uint8_t	I	number of supported indicators in the array "ind".
ind	zwind_sup_t *	I	array of supported indicators.
valid	int	I	validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 232 – zwind_sup_t Structure

Attribute	Type	I/O	Description
ind_id	uint8_t	O	Indicator ID, ZWIND_ID_XXX.
propty_cnt	uint8_t	O	Number of property ID in the buffer "propty".
propty	uint8_t *	O	Buffer to store property ID.

9.6.5 zwif_ind_sup_cache_get

Get all supported indicators information from cache. Caller must call zwif_ind_sup_free() to free the supported indicators if this call is successful.

Table 233 – zwif_ind_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
ind_cnt	uint8_t *	O	Supported indicator counts.
ind	zwind_sup_t **	O	Array of supported indicators.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.6.6 zwif_ind_sup_free

Free supported indicators buffer.

Table 234 – zwif_ind_sup_free Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Indicator interface handle.
ind_cnt	uint8_t	I	Supported indicator counts.
ind	zwind_sup_t *	I	Array of supported indicators.

9.7 Wakeup Interface API

This corresponds to the Wakeup CC, which implies a sleep-capable node. This interface should not be used when ZIPGW mailbox is used.

9.7.1 zwif_wakeup_get

Get wakeup information through report callback function. The wakeup information includes minimum, maximum, default and current wake-up interval, the node that will receive wakeup notification, and the wakeup interval step size.

Table 235 – zwif_wakeup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Wakeup interface handle.
cb	zwrep_wakeup_fn	I	Report callback function.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 236 – zwrep_wakeup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Wakeup interface handle.
cap	zwif_wakeup_t *	I	Capabilities report, NULL for wake up notification.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 237 – zwif_wakeup_t Structure

Attribute	Type	I/O	Description
min	uint32_t	O	Minimum in seconds
max	uint32_t	O	Maximum in seconds
def	uint32_t	O	Default in seconds
interval	uint32_t	O	Steps between min and max
cur	uint32_t	O	Current setting in seconds
node	zwnoded_t	O	Node to send wakeup notification

9.7.2 zwif_wakeup_set

Set wakeup interval and node to notify on wakeup.

Table 238 – zwif_wakeup_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Wake up interface handle.
secs	uint32_t	I	Interval in seconds (24 bit).
node	zwnoded_t *	I	Node to notify.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

9.8 Status Interface API

This corresponds to the Application Status CC.

9.8.1 zwif_appl_busy_rpt_set

Setup an application busy status report callback function.

Table 239 – zwif_appl_busy_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Status interface handle.
rpt_cb	zwrep_appl_busy_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 240 – zwrep_appl_busy_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Status interface handle.
status	uint8_t	I	busy status (ZW_BSY_STS_XXX).
wait_tm	uint8_t	I	wait time in seconds; only valid if status = ZW_BSY_STS_TRY_WAIT.

Table 241 – ZW_BSY_STS_XXX

Application Busy Status	Description
ZW_BSY_STS_TRY	Try again later.
ZW_BSY_STS_TRY_WAIT	Try again in “Wait Time” seconds.
ZW_BSY_STS_Q	Request queued, will be executed later.

9.8.2 zwif_appl_reject_rpt_set

Setup an application rejected request report callback function.

Table 242 – zwif_appl_reject_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Status interface handle.
rpt_cb	zwrep_appl_reject_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 243 – zwrep_appl_reject_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Status interface handle.
status	uint8_t	I	application rejected request status (ZW_RJ_STS_XXX).

Table 244 – ZW_RJ_STS_XXX

Application Rejected Status	Description
ZW_RJ_STS_REJECT	Supported command rejected by the application in the receiving node.

10 Network CC Based Interfaces

Inclusion Controller, Network Management, NOP, Node Provisioning, ZIP and ZIP_ND CCs are handled by the Network APIs. Mailbox CC is used automatically by the Interface APIs when ZIPGW Mailbox is in use. Although the Power Level CC is exposed via an interface, it is typically only used via the Network Health API.

10.1 Z/IP Gateway Interface API

This corresponds to the Z/IP Gateway CC.

10.1.1 zwif_gw_mode_set

Configure ZIPGW operating mode.

Table 245 – zwif_gw_mode_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle.
mode	uint8_t	I	Operating mode; ZW_GW_XXX. ZW_GW_STAND_ALONE = Stand alone mode as ZIPGW; ZW_GW_PORTAL = Portal mode with its configuration pushed from a portal server.
portal_profile	zwgw_portal_prof_t *	I	portal profile; must be valid if mode is ZW_GW_PORTAL.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 246 – zwgw_portal_prof_t structure

Attribute	Type	I/O	Description
port	uint16_t	I	Port number that the portal is listening on
addr6	uint8_t[16]	I	Full IPv6 address with no compression. The address SHOULD be in the ULA IPv6 prefix or in a globally routable IPv6 prefix. The address MAY be an IPv4-mapped IPv6 address. The field MUST NOT carry a link-local IPv6 address. The IPv6 address MAY be specified as ::/128 (all zeros), i.e. the unspecified address. If setting the IPv6 address field to the unspecified IPv6 address, the portal_name field MUST be set to a DNS-resolvable FQDN.
portal_name	uint8_t[64]	I	Optional portal name field MUST be formatted as a UTF-8 based FQDN string such as “example.com” without null terminated character.
name_len	uint8_t	I	Portal_name length. Valid value: 0 to 63.

10.1.2 zwif_gw_mode_get

Get ZIPGW operating mode through report callback.

Table 247 – zwif_gw_mode_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle.
cb	zwrep_gw_mode_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 248 – zwrep_gw_mode_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle.
status	uint8_t	I	0=success; 1=failure due to either timeout or the gateway is locked with its parameter hidden; 2=transmit error.
mode	uint8_t	I	Operating mode; ZW_GW_XXX. ZW_GW_STAND_ALONE = Stand alone mode as ZIPGW; ZW_GW_PORTAL = Portal mode with its configuration pushed from a portal server.
portal_profile	zwgw_portal_prof_t *	I	Portal profile; only valid if mode is ZW_GW_PORTAL. If it is NULL, no profile is stored in the ZIPGW.

10.1.3 zwif_gw_cfg_lock

Lock ZIPGW configuration parameters. Once the ZIPGW has been locked, it MUST NOT be possible to unlock the device unless :

- A factory default reset unlocks the ZIPGW and reverts all settings to default
- An unlock command received via an authenticated secure connection with the portal

Table 249 – zwif_gw_mode_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle.
lock	uint8_t	I	Lock configuration parameters. 1= lock; 0= unlock.
show	uint8_t	I	Control whether to allow configuration parameters to be read back. 1= allow read back; 0= disallow.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

10.1.4 zwif_gw_unsolicit_set

Configure ZIPGW unsolicited message destination.

Table 250 – zwif_gw_unsolicit_set Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	ZIPGW interface handle.
dst_ip	uint8_t *	I	unsolicited destination IPv6 address.
dst_port	uint16_t	I	unsolicited destination port .
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

10.1.5 zwif_gw_unsolicit_get

Get ZIPGW unsolicited message destination through report callback.

Table 251 – zwif_gw_unsolicit_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle.
cb	zwrep_gw_unsolicit_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 252 – zwrep_gw_unsolicit_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	ZIPGW interface handle
dst_ip	uint8_t *	I	Unsolicited destination IPv6 address
dst_port	uint16_t	I	Unsolicited destination port

10.2 Z/IP Portal Interface API

This corresponds to the Z/IP portal CC.

10.2.1 zwif_gw_cfg_set

Set ZIPGW portal mode configuration.

Table 253 – zwif_gw_cfg_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Z/IP portal interface handle.
cfg	zwportal_cfg_t *	I	Configuration parameters.
cb	zwrep_cfg_sts_fn	I	Configuration status callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 254 – zwportal_cfg_t Structure

Attribute	Type	I/O	Description
lan_ipv6_addr	uint8_t[16]	I	LAN IPv6 address of ZIPGW. May be all-zeroes IPv6 address for auto-configuration.
dflt_gw	uint8_t[16]	I	Default IPv6 gateway.

pan_prefix	uint8_t[16]	I	PAN interface prefix with /64 prefix length. May be all-zeroes IPv6 address for auto-configuration.
portal_ipv6_prefix	uint8_t[16]	I	Portal IPv6 address. The ZIPGW must route all IP traffic for the portal ipv6 prefix into the secure TLS connection.
lan_ipv6_prefix_len	uint8_t	I	LAN IPv6 address prefix length.
portal_ipv6_prefix_len	uint8_t	I	Portal IPv6 prefix length.

Table 255 – zwrep_cfg_sts_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Z/IP portal interface handle.
sts	uint8_t	I	Configuration status. 0xFF = o.k.; 0x01 = invalid configuration block.

10.2.2 zwif_gw_cfg_get

Get ZIPGW portal mode configuration through report callback.

Table 256 – zwif_gw_mode_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Z/IP portal interface handle.
cb	zwrep_gw_cfg_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 257 – zwrep_gw_cfg_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Z/IP portal interface handle.
cfg	zwportal_cfg_t *	I	Configuration parameters.

10.3 Power Level Interface API

This corresponds to the Z-Wave Power Level CC. This interface is meant to be used in an installation or test situation.

10.3.1 zwif_power_level_rpt_set

Set up a power level report callback function.

Table 258 – zwif_power_level_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.

rpt_cb	zwrep_power_level_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 259 – zwrep_power_level_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
lvl	uint8_t	I	Current power level indicator value in effect on the node. Ranges from 0 to 9. 0=normal power; 1= -1dbm; 2= -2dbm, etc.
timeout	uint8_t	I	Time out value, in seconds.

10.3.2 zwif_power_level_get

Get the power level indicator value in use by the node through report callback function.

Table 260 – zwif_power_level_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

10.3.3 zwif_power_level_set

Set the power level which should be used by the node when transmitting RF.

Table 261 – zwif_power_level_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
lvl	uint8_t	I	Power level. Ranges from 0 to 9. 0=normal power; 1= -1dbm; 2= -2dbm, etc.
timeout	uint8_t	I	Time out value (in seconds) ranges from 1-255 before resetting to normal power level.

10.3.4 zwif_power_level_test_rpt_set

Setup power level test report callback function

Table 262 – zwif_power_level_test_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
rpt_cb	zwrep_power_level_test_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 263 – zwrep_power_level_test_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.

node_id	uint8_t	I	Test node ID. If node ID is 0, it means no test has been made and the rest of the parameters should be ignored.
status	uint8_t	I	Status of the test operation. POWERLEVEL_TEST_XXX.
frame_cnt	uint8_t	I	Number of frame count which has been acknowledged by the node as specified by node_id.

Table 264 – POWERLEVEL_TEST_XXX

Power level test status	Description
POWERLEVEL_TEST_FAILED	No test frame transmissions has been acknowledged.
POWERLEVEL_TEST_SUCCES	At least 1 test frame transmission has been acknowledged.
POWERLEVEL_TEST_INPROGRESS	Test is still in progress.

10.3.5 zwif_power_level_test_get

Get the result of power level test through report callback.

Table 265 – zwif_power_level_test_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

10.3.6 zwif_power_level_test_set

Set the power level which should be used by the node when transmitting RF.

Table 266 – zwif_power_level_test_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Power level interface handle.
noded	zwnoded_t *	I	Node descriptor of the test node which should receive the transmitted test frames.
lvl	uint8_t	I	Power level. Ranges from 0 to 9. 0=normal power; 1= -1dbm; 2= -2dbm, etc.
frame_cnt	uint16_t	I	Test frame count to be carried out. (1-65535).

11 Application CC Based Interfaces

11.1 Basic Interface API

This corresponds to the Z-Wave Basic CC, allowing 8 bit get/set operations. As a node need not advertise this in its node information frame, this API will always be available for each endpoint.

11.1.1 zwif_basic_rpt_set

Set up a basic report callback function.

Table 267 – zwif_basic_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Basic interface handle.
rpt_cb	zwrep_basic_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 268 – zwrep_basic_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
val	zwbasic_t *	I	Basic data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected

Table 269 – zwbasic_t Structure

Attribute	Type	I/O	Description
curr_val	uint8_t	I	Current value: 0 = off. 0x01~0x63 = percentage (%). 0xFE = Unknown. 0xFF = On.
tgt_val	uint8_t	I	Target value with same interpretation as curr_val.
dur	uint8_t	I	Duration: 0 = already at the target. 0x01~0x7F = seconds. 0x80~0xFD = 1~126 minutes. 0xFE = Unknown duration. 0xFF = reserved.

11.1.2 zwif_basic_get

Get the value of an interface that is mapped to the basic interface through report callback function.

Table 270 – zwif_basic_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Basic interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.1.3 zwif_basic_set

Set the value of an interface that is mapped to the basic interface.

Table 271 – zwif_basic_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Basic interface handle
v	uint8_t	I	Value (the range of value is device specific).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.2 Switch Interface API

This corresponds to the Z-Wave Binary Switch CC.

11.2.1 zwif_switch_rpt_set

Set up the report callback.

Table 272 – zwif_switch_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Switch interface handle.
cb	zwrep_switch_fn	I	Switch report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 273 – zwrep_switch_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Switch interface handle.
val	zwswitch_t *	I	Switch data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a switch state change is detected.

Table 274 – zwswitch_t Structure

Attribute	Type	I/O	Description
curr_val	uint8_t	I	Current value: 0 – off 0xFE – Unknown 0xFF – on.
tgt_val	uint8_t	I	Target value with same interpretation as curr_val.
dur	uint8_t	I	Duration: 0 = already at the target. 0x01~0x7F = seconds. 0x80~0xFD = 1~126 minutes. 0xFE = Unknown duration. 0xFF = reserved.

11.2.2 zwif_switch_get

Solicit a report of the current state of the switch.

Table 275 – zwif_switch_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Switch interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 276 – ZWIF_GET_BMSK_XXX

Interface “get” API flag bitmask	Description
ZWIF_GET_BMSK_CACHE	Get cached data.
ZWIF_GET_BMSK_LIVE	Get live report data by invoking the zwif_XXX_get command.

11.2.3 zwif_switch_set

Turn switch on or off.

Table 277 – zwif_switch_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Switch interface handle.
on	uint8_t	I	0 – off. Non-zero – on.
dur	uint8_t	I	Duration that the transition should take. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127 seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default duration.

cb	zw_postset_fn	I	Optional post-set polling callback function. NULL if no callback required.
usr_param	void *	I	Optional user-defined parameter passed in callback
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.2.4 zwif_switch_mset

Turn on or off switches using multicast addressing if available.

Table 278 – zwif_switch_mset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of switch interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_switch_set() with dur=0, cb=NULL and usr_param=NULL
on	uint8_t	I	0 – off. Non-zero – on.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 279 – zw_postset_fn Parameters

Attribute	Type	I/O	Description	
net	zwnet_t *	I	Network handle.	
node_id	uint8_t	I	Node ID.	
ep_id	uint8_t	I	Endpoint ID	
cls_id	uint16_t	I	CC (interface) ID.	
usr_param	void *	I	User-defined parameter.	
reason	int	I	Reason for the callback, as in ZWPSET_REASON_XXX:	
			XXX	Description
			TGT_HIT	Set target was hit.
			TIMEOUT	Timeout, set target was not hit.
			UNSUPPORTED	Post-set polling is unsupported for this interface.
			DEVICE_RMV	Device removed.
			BUSY	Device is busy. Try again the zwif_xxx_set() function call later.
			SEND_FAILED	Failed to send command.
			FAILED	Failed to hit the target for some reason.

11.3 Level Interface API

This corresponds to the Z-Wave Multi-Level Switch CC.

11.3.1 zwif_level_rpt_set

Set up a report callback and trigger switch type information callback through zwrep_level_fn.

Table 280 – zwif_level_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
cb	zwrep_level_fn	I	Level report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 281 – zwrep_level_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
val	zwlevel_dat_t *	I	Multilevel switch data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a level change is detected.

Table 282 – zwlevel_dat_t structure

Attribute	Type	I/O	Description
curr_val	uint8_t	I	Current value: 0 = off 0x01~0x63 = percentage (%) 0xFE = Unknown 0xFF = On.
tgt_val	uint8_t	I	Target value with same interpretation as curr_val
dur	uint8_t	I	Duration: 0 = already at the target 0x01~0x7F = seconds 0x80~0xFD = 1~126 minutes 0xFE = Unknown duration 0xFF = reserved

11.3.2 zwif_level_sup_get

Get a switch type report through report callback. Older versions of target nodes may not support this function.

Table 283 – zwif_level_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle
cb	zwrep_lvl_sup_fn	I	Callback function to receive the supported switch type report
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 284 – zwrep_lvl_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
pri_type	uint8_t	I	Primary switch type. Possible types: 0 – not supported 1 – off/on 2 – down/up 3 – close/open 4 – counterclockwise/clockwise 5 – left/right 6 – reverse/forward 7 – push/pull
sec_type	uint8_t	I	Secondary switch type. Possible types same as those for primary switch.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.3.3 zwif_level_sup_cache_get

Get the supported switch types from cache.

Table 285 – zwif_level_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
pri_type	uint8_t *	O	Primary switch type, SW_TYPE_XXX.
sec_type	uint8_t *	O	Secondary switch type , SW_TYPE_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.4 zwif_level_get

Solicit a report of the current level.

Table 286 – zwif_level_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.5 zwif_level_set

Set level and other parameters. Older versions of target nodes may not support all parameters.

Table 287 – zwif_level_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle
v	uint8_t	I	Level 0: OFF

			255: previous state 1-99: %
dur	uint8_t	I	Dimming duration 0: instant 1-127: in seconds 128-254: 1 to 127 minutes 255: default factory rate
cb	zw_postset_fn	I	Optional post-set polling callback function. NULL if no callback required.
usr_param	void *	I	Optional user-defined parameter passed in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.6 zwif_level_mset

Set multilevel switches using multicast addressing if available.

Table 288 – zwif_level_mset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_level_set() with dur=0, cb=NULL and usr_param=NULL
v	uint8_t	I	Level 0: OFF 255: previous state 1-99: %
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.7 zwif_level_start

Start a level change.

Table 289 – zwif_level_start Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
level_ctrl	zwlevel_t *	I	Level parameters.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 290 – zwlevel_t structure

Attribute	Type	I/O	Description
pri_dir	uint8_t	I	Primary switch direction. 0 – up 1 – down 3 – no change
pri_level	uint8_t	I	Primary switch start level

pri_ignore_lvl	uint8_t	I	Ignore primary switch start level. 0 – use primary switch start level 1 - ignore start level
sec_dir	uint8_t	I	Secondary switch direction, same definition as primary
sec_step	uint8_t	I	Secondary switch step size: 0-99 - % 255 – factory default
dur	uint8_t	I	Dimming duration in seconds which is the interval it takes to dim from level 0 to 99.

11.3.8 zwif_level_mstart

Start modifying levels using multicast addressing if available.

Table 291 – zwif_level_mstart Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_level_start()
level_ctrl	zwlevel_t *	I	Level parameters.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.9 zwif_level_stop

Stop a level change.

Table 292 – zwif_level_stop Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Level interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.3.10 zwif_level_mstop

Stop modifying levels using multicast addressing if available.

Table 293 – zwif_level_mstop Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_level_stop()
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.4 Color Switch Interface API

This corresponds to the Z-Wave Color Switch CC.

11.4.1 zwif_color_sw_rpt_set

Set up a color switch report callback function

Table 294 – zwif_color_sw_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_color_sw_get_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 295 – zwrep_color_sw_get_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
data	zwcolor_t *	I	Color component data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected

Table 296 – zwcolor_t Structure

Attribute	Type	I/O	Description
id	uint8_t	O	Color component ID, COL_SW_COMP_ID_XXX
value	uint8_t	O	Value or current value
target_val	uint8_t	O	Target value
dur	uint8_t	O	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved.

Table 297 – COL_SW_COMP_ID_XXX

Color Switch Component ID	Description
COL_SW_COMP_ID_WARM_WHITE	Warm white
COL_SW_COMP_ID_COLD_WHITE	Cold white
COL_SW_COMP_ID_RED	Red
COL_SW_COMP_ID_GREEN	Green
COL_SW_COMP_ID_BLUE	Blue
COL_SW_COMP_ID_AMBER	Amber (for 6 channels color mixing)
COL_SW_COMP_ID_CYAN	Cyan (for 6 channels color mixing)
COL_SW_COMP_ID_PURPLE	Purple (for 6 channels color mixing)
COL_SW_COMP_ID_INDEX	Indexed color

11.4.2 zwif_color_sw_get

Get a color component value through report callback.

Table 298 – zwif_color_sw_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id	uint8_t	O	color component ID, COL_SW_COMP_ID_XXX.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.4.3 zwif_color_sw_set

Set color component values.

Table 299 – zwif_color_sw_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cnt	uint8_t	I	Color component count (maximum 31).
id	uint8_t *	I	Pointer to first element of the color component ID (see COL_SW_COMP_ID_XXX) array.
val	uint8_t *	I	Pointer to first element of the color component value array.
dur	uint8_t	I	Level change duration. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127 seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default rate.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.4.4 zwif_color_sw_start

Start increasing/decreasing color switch level.

Table 300 – zwif_color_sw_start Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
color_ctrl	zwcol_ctl_t *	I	Level control of color switch.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 301 – zwcol_ctl_t structure

Attribute	Type	I/O	Description
dir	uint8_t	O	Color switch direction, COL_SW_LVL_XXX
id	uint8_t	O	Color component ID, COL_SW_COMP_ID_XXX
use_start_lvl	uint8_t	O	Flag to indicate whether to start changing level from user supplied start_level. 1=use user supplied start level; 0=start from current level
start_level	uint8_t	O	Color switch start level (only valid if use_start_lvl=1)
dur	uint8_t	O	Level change duration

Table 302 – COL_SW_LVL_XXX

Color switch direction	Description
COL_SW_LVL_UP	Increase level
COL_SW_LVL_DOWN	Decrease level

11.4.5 zwif_color_sw_stop

Stop changing color switch level.

Table 303 – zwif_color_sw_stop Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id	uint8_t	O	Color component ID, COL_SW_COMP_ID_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.4.6 zwif_color_sw_sup_get

Get supported barrier color components through report callback.

Table 304 – zwif_color_sw_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cb	zwrep_color_sw_sup_fn	I	Callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 305 – zwrep_color_sw_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
comp_cnt	uint8_t	I	Number of color components in comp_id array.
comp_id	uint8_t *	I	Color component ID (see COL_SW_COMP_ID_XXX) array.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.4.7 zwif_color_sw_sup_cache_get

Get supported color components from cache.

Table 306 – zwif_color_sw_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
comp_cnt	uint8_t *	O	Number of color components.

comp_id	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported color component ID (see COL_SW_COMP_ID_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5 Window Covering Interface API

This corresponds to the Z-Wave Window Covering CC. It is used to control the amount of light through window.

11.5.1 zwif_wincvr_rpt_set

Set up a window covering parameter status report callback function

Table 307 – zwif_wincvr_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_wincvr_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 308 – zwrep_wincvr_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
val	wincvr_dat_t *	I	Window covering parameter status data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected

Table 309 – wincvr_dat_t Structure

Attribute	Type	I/O	Description
id	uint8_t	O	Parameter id WIN_COVER_ID_XXX
curr_val	uint8_t	O	Current value with range 0~99. 0=Closed; 99=Open
tgt_val	uint8_t	O	Target value with the same interpretation as curr_val
dur	uint8_t	O	Duration needed to reach the target value: 0 = already at the target; 0x01~0x7F = seconds in 1-second resolution; 0x80~0xFD = 1~126 minutes in 1-minute resolution; 0xFE = Unknown duration; 0xFF = reserved

Table 310 – WIN_COVER_ID_XXX

Window covering parameter ID	Description

WIN_COVER_ID_OUT_L	Outbound edge towards the left. Level changed up: opening; level changed down: closing
WIN_COVER_ID_OUT_L_POS	Outbound edge towards the left with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_OUT_R	Outbound edge towards the right. Level changed up: opening; level changed down: closing
WIN_COVER_ID_OUT_R_POS	Outbound edge towards the right with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_L	Inbound edge towards the left. Level changed up: opening; level changed down: closing
WIN_COVER_ID_IM_L_POS	Inbound edge towards the left with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_R	Inbound edge towards the right. Level changed up: opening; level changed down: closing
WIN_COVER_ID_IM_R_POS	Inbound edge towards the right with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_RL	Inbound edges controlled horizontally. Level changed up: opening; level changed down: closing
WIN_COVER_ID_IM_RL_POS	Inbound edges controlled horizontally with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_V_SLAT_ANG	Vertical slats angle (Right/Left movement). Level changed up: closing to the left inside; level changed down: closing to the right inside
WIN_COVER_ID_V_SLAT_ANG_POS	Vertical slats angle (Right/Left movement) with position value range 0~99. 0=Closed (to the right inside); 50=Open; 99=Closed (to the left inside)
WIN_COVER_ID_OUT_B	Outbound edge towards the bottom. Level changed up: opening; level changed down: closing
WIN_COVER_ID_OUT_B_POS	Outbound edge towards the bottom with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_OUT_T	Outbound edge towards the top. Level changed up: opening; level changed down: closing
WIN_COVER_ID_OUT_T_POS	Outbound edge towards the top with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_B	Inbound edge towards the bottom. Level changed up: opening; level changed down: closing
WIN_COVER_ID_IN_B_POS	Inbound edge towards the bottom with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_T	Inbound edge towards the top. Level changed up: opening; level changed down: closing
WIN_COVER_ID_IN_T_POS	Inbound edge towards the top with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_IN_TB	Inbound edges controlled vertically. Level changed up: opening; level changed down: closing

WIN_COVER_ID_IM_TB_POS	Inbound edges controlled vertically with position value range 0~99. 0=Closed; 99=Open
WIN_COVER_ID_H_SLAT_ANG	Horizontal slats angle (Up/Down movement). Level changed up: closing down inside; level changed down: closing up inside
WIN_COVER_ID_H_SLAT_ANG_POS	Horizontal slats angle (Up/Down movement) with position value range 0~99. 0=Closed (up inside); 50=Open; 99=Closed (down inside)

11.5.2 zwif_wincvr_get

Get window covering parameter status through report callback.

Table 311 – zwif_wincvr_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id	uint8_t	O	Parameter id with position control (see WIN_COVER_ID_XXX_POS)
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.3 zwif_wincvr_set

Set the position of the window covering.

Table 312 – zwif_wincvr_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cnt	uint8_t	I	Parameter count (maximum 31)
id	uint8_t *	I	Parameter ids array of those with position control (see WIN_COVER_ID_XXX_POS)
val	uint8_t *	I	Parameter value array with the values correspond to the parameter ids in "id" array
dur	uint8_t	I	Time that the transition should take from the current value to the new target value. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127 seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default rate.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.4 zwif_wincvr_mset

Set the position of the window covering using multicast addressing if available.

Table 313 – zwif_wincvr_mset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_wincvr_set()
cnt	uint8_t	I	Parameter count (maximum 31)
id	uint8_t *	I	Parameter ids array of those with position control (see WIN_COVER_ID_XXX_POS)
val	uint8_t *	I	Parameter value array with the values correspond to the parameter ids in "id" array
dur	uint8_t	I	Time that the transition should take from the current value to the new target value. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127 seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default rate.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.5 zwif_wincvr_start

Start transitioning window covering to a new level.

Table 314 – zwif_wincvr_start Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id	uint8_t	O	Parameter id (see WIN_COVER_ID_XXX)
dir	uint8_t	O	Direction of the level change. 0 = increase level; 1 = decrease level
dur	uint8_t	O	Duration from the minimum value to the maximum value which dictates the level change rate. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127 seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default rate.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.6 zwif_wincvr_mstart

Start transitioning window covering to a new level using multicast addressing if available.

Table 315 – zwif_wincvr_mstart Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_wincvr_start()
id	uint8_t	O	Parameter id (see WIN_COVER_ID_XXX)
dir	uint8_t	O	Direction of the level change. 0 = increase level; 1 = decrease level
dur	uint8_t	O	Duration from the minimum value to the maximum value which dictates the level change rate. 0=instantly; 0x01 to 0x7F = 1 second (0x01) to 127

			seconds (0x7F); 0x80 to 0xFE = 1 minute (0x80) to 127 minutes (0xFE); 0xFF = factory default rate.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.7 zwif_wincvr_stop

Stop an ongoing transition started by zwif_wincvr_start API.

Table 316 – zwif_wincvr_stop Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id	uint8_t	O	Parameter id (see WIN_COVER_ID_XXX)
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.8 zwif_wincvr_mstop

Stop an ongoing transition started by zwif_wincvr_start API using multicast addressing if available.

Table 317 – zwif_wincvr_mstop Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_wincvr_stop()
id	uint8_t	O	Parameter id (see WIN_COVER_ID_XXX)
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.5.9 zwif_wincvr_sup_get

Get window covering supported parameter IDs through report callback.

Table 318 – zwif_wincvr_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cb	zwrep_wincvr_sup_fn	I	Callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 319 – zwrep_wincvr_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id_len	uint8_t	I	Size of parameter IDs buffer.
id	uint8_t *	I	Buffer to store supported parameter IDs (WIN_COVER_ID_XXX)
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.5.10 zwif_wincvr_sup_cache_get

Get window covering supported parameter IDs from cache.

Table 320 – zwif_wincvr_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
id_cnt	uint8_t *	O	Supported parameter IDs count
id	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported parameter IDs (WIN_COVER_ID_XXX)
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6 Barrier Operator Interface API

This corresponds to the Z-Wave Barrier Operator CC.

11.6.1 zwif_barrier_rpt_set

Set up a barrier operator state report callback function.

Table 321 – zwif_barrier_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_barrier_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 322 – zwrep_barrier_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle
state	uint8_t	I	Barrier operator state, ZW_BAR_STA_XXX. It may also be in a range from 1 to 99 (%) open.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 323 – ZW_BAR_STA_XXX

Barrier operator state	Description
ZW_BAR_STA_CLOSED	The barrier is in the closed position.
ZW_BAR_STA_CLOSING	The barrier is closing; current position is unknown.
ZW_BAR_STA_STOPPED	The barrier is stopped; current position is unknown.
ZW_BAR_STA_OPENING	The barrier is opening; current position is unknown.
ZW_BAR_STA_OPEN	The barrier is in the open position.

11.6.2 zwif_barrier_get

Get barrier operator state through report callback.

Table 324 – zwif_barrier_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6.3 zwif_barrier_set

Set barrier operator state.

Table 325 – zwif_barrier_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tgt_state	uint8_t	I	target state 0: close, 0xFF:open
cb	zw_postset_fn	I	Optional post-set polling callback function. NULL if no callback required.
usr_param	void *	I	Optional user-defined parameter passed in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6.4 zwif_barrier_mset

Set barrier operator state using multicast addressing if available.

Table 326 – zwif_barrier_mset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_barrier_set()with cb=NULL and usr_param=NULL
tgt_state	uint8_t	I	target state 0: close, 0xFF:open
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6.5 zwif_barrier_notif_sup_get

Get supported barrier operator notification subsystem report through report callback.

Table 327 – zwif_barrier_notif_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cb	zwrep_ barrier_notif_sup_fn	I	callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 328 – zwrep_barrier_notif_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
type_len	uint8_t	I	Size of barrier operator notification subsystem type buffer.
type	uint8_t *	I	Buffer to store supported barrier operator notification subsystem types (ZW_BAR_NOTIF_TYP_XXX).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 329 – ZW_BAR_NOTIF_TYP_XXX

Barrier operator notification subsystem type	Description
ZW_BAR_NOTIF_TYP_UNSUPPORTED	Not supported.
ZW_BAR_NOTIF_TYP_AUDIBLE	Audible Notification subsystem (e.g. siren).
ZW_BAR_NOTIF_TYP_VISUAL	Visual Notification subsystem (e.g. flashing light).

11.6.6 zwif_barrier_notif_sup_cache_get

Get supported barrier operator notification subsystem from cache.

Table 330 – zwif_barrier_notif_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
type_cnt	uint8_t *	O	supported barrier operator notification subsystem types counts.
sup_type	uint8_t *	O	caller supplied buffer of size 255 bytes to store barrier operator notification subsystem types (ZW_BAR_NOTIF_TYP_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6.7 zwif_barrier_notif_rpt_set

Set up a barrier operator notification subsystem configuration report callback function.

Table 331 – zwif_barrier_notif_rpt_set Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_barrier_subsys_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 332 – zwrep_barrier_subsys_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
subsys_type	uint8_t	I	Barrier operator subsystem type, ZW_BAR_NOTIF_TYP_XXX.
subsys_sta	uint8_t	I	Barrier operator subsystem state: 0=off; 0xFF=on.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	state number that is incremented by one whenever a cache change is detected.

11.6.8 zwif_barrier_notif_cfg_get

Get barrier operator notification subsystem configuration through report callback.

Table 333 – zwif_barrier_notif_cfg_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
type	uint8_t	I	Notification subsystem type ZW_BAR_NOTIF_TYP_XXX; except ZW_BAR_NOTIF_TYP_UNSUPPORTED.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.6.9 zwif_barrier_notif_cfg_set

Enable/disable barrier operator notification subsystem.

Table 334 – zwif_barrier_notif_cfg_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
type	uint8_t	I	Notification subsystem type ZW_BAR_NOTIF_TYP_XXX; except ZW_BAR_NOTIF_TYP_UNSUPPORTED.
state	uint8_t	I	State : 0=disable; 0xFF=enable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.7 Sound Switch Interface API

This corresponds to the Z-Wave Sound Switch CC.

11.7.1 zwif_snd_switch_rpt_set

Set up a sound switch state report callback function.

Table 335 – zwif_snd_switch_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tone_config_rpt_cb	zwrep_snd_switch_config_fn	I	Tone config Report callback function.
tone_play_rpt_cb	zwrep_snd_switch_tone_play_fn	I	Tone play Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 336 – zwrep_snd_switch_config_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
volume	uint8_t	I	Device current volume in percentage. In the range of 0..100.
def_tone_id	uint8_t	I	Current configured default tone.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 337 – zwrep_snd_switch_tone_play_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tone_id	uint8_t	I	Tone ID of the tone which is currently being played. 0 indicates no tone is playing.
tone_vol	uint8_t	I	Tone volume. 0 = use the configured current volume set by the zwif_snd_switch_config_set(). Value 1 to 100 indicate the actual volume setting from respectively 1% to 100%. 255 = use most recent non-zero volume setting if the current volume is muted (0). If the current configured volume is not muted (> 0), this value (255) indicates to use the configured current volume set by the zwif_snd_switch_config_set().
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

11.7.2 zwif_snd_switch_config_get

Get sound switch configuration setting through report callback.

Table 338 – zwif_snd_switch_config_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.7.3 zwif_snd_switch_config_set

Set sound switch configuration.

Table 339 – zwif_snd_switch_config_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
volume	uint8_t	I	Volume setting. 0 means mute volume. 1..100 indicates actual volume from 1% to 100%. 255 means unmute volume, or modify default tone only without modifying volume setting.
def_tone_id	uint8_t	I	Default tone ID to set. 0 means no change on the default tone.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.7.4 zwif_snd_switch_tone_play_get

Get sound switch currently played tone through report callback.

Table 340 – zwif_snd_switch_tone_play_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.7.5 zwif_snd_switch_tone_play_set

Play/Stop tone on sound switch device.

Table 341 – zwif_snd_switch_tone_play_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tone_id	uint8_t	I	Tone ID of which to play. 0 means stop playing any tone. 255 means play the default tone.
tone_vol	uint8_t	I	Tone volume if the device supports it; else this parameter will be ignored. 0 = use the configured current volume set by the zwif_snd_switch_config_set(). Value 1 to 100 indicate the actual volume setting from respectively 1% to 100%.

			255 = use most recent non-zero volume setting if the current volume is muted (0). If the current configured volume is not muted (> 0), this value (255) indicates to use the configured current volume set by the zwif_snd_switch_config_set().
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.7.6 zwif_snd_switch_tone_info_get

Get tone information for a specific tone ID or all the tone information through report callback.

Table 342 – zwif_snd_switch_tone_info_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tone_id	uint8_t	I	0 indicates to get all the tone information from cache. Non-0 indicates to get a specific tone-id information.
cb	zwrep_snd_switch_tone_info_fn	I	callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 343 – zwrep_snd_switch_tone_info_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
tone_cnt	uint8_t	I	number of tone information stored in ptone_info_list
ptone_info_list	zwsnd_switch_tone_info_t *	I	Array of tone information data
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored

Table 344 – zwsnd_switch_tone_info_t Structure

Attribute	Type	I/O	Description
wtone_info_sz	uint16_t	O	Size of Tone information data
ptone_info	if_snd_switch_tone_info_t *	O	Tone information data

Table 345 – if_snd_switch_tone_info_t Structure

Attribute	Type	I/O	Description
tone_id	uint8_t	O	Tone ID
tone_duration	uint16_t	O	Tone duration
tone_name_len	uint8_t	O	Tone name length in bytes (including NUL terminating character)
tone_name	char [1]	O	Tone name string place holder in UTF-8

11.8 Binary Sensor Interface API

This corresponds to the Z-Wave Binary Sensor CC.

11.8.1 zwif_bsensor_rpt_set

Set up a binary sensor report callback.

Table 346 – zwif_bsensor_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle.
rpt_cb	zwrep_bsensor_fn	I	Binary sensor report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 347 – zwrep_bsensor_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle
state	uint8_t	I	0 – idle 1 – event detected
type	uint8_t	I	Sensor type, ZW_BSENSOR_TYPE_XXX. If type equals to zero, sensor type is unknown.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

11.8.2 zwif_bsensor_get

Solicit a report of the current state of the binary sensor.

Table 348 – zwif_bsensor_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle.
type	uint8_t	I	Preferred sensor type, ZW_BSENSOR_TYPE_XXX. If type equals to zero, the sensor report will return the factory default sensor type.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.8.3 zwif_bsensor_sup_get

Get the supported binary sensor types through report callback.

Table 349 – zwif_bsensor_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle.
cb	zwrep_bsensor_sup_fn	I	Report callback function.

cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 350 – zwrep_bsensor_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle.
type_len	uint8_t	I	Size of sensor type buffer.
type	uint8_t *		buffer to store supported sensor types (ZW_BSENSOR_TYPE_XXX).
valid	int	I	validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.8.4 zwif_bsensor_sup_cache_get

Get the supported binary sensor types from cache.

Table 351 – zwif_bsensor_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Binary sensor interface handle.
snsr_cnt	uint8_t *	O	Supported sensor counts.
sup_snsr	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported sensors (ZW_BSENSOR_TYPE_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.9 Alarm/Notification Interface API

This corresponds to the Z-Wave Alarm/Notification CC.

11.9.1 zwif_alrm_rpt_set

Set up an Alarm/Notification report callback function.

Table 352 – zwif_alrm_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
rpt_cb	zwrep_alrm_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 353 – zwrep_alrm_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
alarm_info	zwalrm_t *	I	Alarm/Notification info.

ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 354 – zwalrm_t Structure

Attribute	Type	I/O	Description	
type	uint8_t	O	Vendor-specific alarm type.	
level	uint8_t	O	Vendor-specific alarm level.	
ex_info	uint8_t	O	Are extended information fields are valid? 1= valid; 0= invalid.	
ex_zensr_nodeid	uint8_t	O	Zensor Net source node ID. 0 if not based on ZensorNet.	
ex_status	uint8_t	O	Z-Wave Alarm/Notification status (ZW_ALRM_STS_XXX).	
ex_type	uint8_t	O	Z-Wave Alarm/Notification type (ZW_ALRM_XXX).	
ex_event	uint8_t	O	Z-Wave Alarm/Notification event (ZW_ALRM_EVT_XXX).	
ex_has_sequence	uint8_t	O	Flag to indicate whether this report has sequence number as stored in "ex_sequence_no". 1=valid; 0 = invalid.	
ex_sequence_no	uint8_t	O	Z-Wave Alarm/Notification sequence number. Only valid if field ex_has_sequence is 1.	
ex_evt_len	uint8_t	O	Z-Wave Alarm/Notification event parameter length. Zero if no parameter.	
ex_evt_type	uint8_t	O	Z-Wave Alarm/Notification event parameter type (ZW_ALRM_PARAM_XXX) :	
			ZW_ALRM_PARAM_LOC	Node location UTF-8 string (NULL terminated)
			ZW_ALRM_PARAM_USRID	User ID
			ZW_ALRM_PARAM_OEM_ERR_CODE	OEM proprietary system failure code
			ZW_ALRM_PARAM_PROPRIETARY	Proprietary event parameters
			ZW_ALRM_PARAM_EVENT_ID	Event ID which is no more active
			ZW_ALRM_PARAM_UNKNOWN	Unknown Alarm/Notification event parameters. It could be from a higher version of CC, or the device violates the spec and send Parameters when Spec does not define
ex_evt_prm	uint8_t[1]	O	Z-Wave Alarm/Notification event parameter place holder	

Table 355 – ZW_ALRM_STS_XXX

Z-Wave alarm status	Description

ZW_ALRM_STS_DEACTIVATED	Unsolicited Alarm/Notification report is deactivated (push mode) or report message carries valid notification information (pull mode).
ZW_ALRM_STS_ACTIVATED	Unsolicited Alarm/Notification report is activated (push mode) .
ZW_ALRM_STS_NO_PEND_NOTICE	Report message does not carry valid notification information. The queue is empty (pull mode).

Table 356 – ZW_ALRM_XXX

Z-Wave Alarm/Notification type	Description
ZW_ALRM_SMOKE	Smoke alarm
ZW_ALRM_CO	Carbon monoxide alarm
ZW_ALRM_CO2	Carbon dioxide alarm
ZW_ALRM_HEAT	Heat alarm
ZW_ALRM_WATER	Water alarm
ZW_ALRM_LOCK	Lock access control alarm
ZW_ALRM_BURGLAR	Burglar alarm or home security
ZW_ALRM_POWER	Power management alarm
ZW_ALRM_SYSTEM	System alarm
ZW_ALRM_EMERGENCY	Emergency alarm
ZW_ALRM_CLOCK	Alarm clock
ZW_ALRM_APPLIANCE	Home appliance alarm
ZW_ALRM_HEALTH	Home health alarm
ZW_ALRM_SIREN	Siren alarm
ZW_ALRM_WATER_VLV	Water Valve alarm
ZW_ALRM_WEATHER	Weather alarm
ZW_ALRM_IRRIGATION	Irrigation alarm
ZW_ALRM_GAS	Gas alarm
ZW_ALRM_PEST_CTL	Pest control
ZW_ALRM_LIGHT_SNSR	Light sensor
ZW_ALRM_WATER_QM	Water quality monitoring
ZW_ALRM_HOME_MNTR	Home monitoring
ZW_ALRM_REQ_PEND	Request pending notification
ZW_ALRM_FIRST	Used by the zwif_alrm_get() to retrieve the first alarm detection from the supported list

Table 357 – ZW_ALRM_EVT_XXX

Z-Wave Alarm/ Notification type	Z-Wave alarm/Notification event	Description
All	ZW_ALRM_EVT_UNKNOWN	Unknown event.

	ZW_ALRM_EVT_INACTIVE_CLEAR	Event inactive (push mode) / Previous Events cleared (pull mode)
Smoke alarm	ZW_ALRM_EVT_SMOKE_L	Smoke detected with location
	ZW_ALRM_EVT_SMOKE	Smoke detected
	ZW_ALRM_EVT_SMOKE_TEST	Smoke alarm test
	ZW_ALRM_EVT_SMOKE_REPLA	Replacement required
	ZW_ALRM_EVT_SMOKE_REPLA_EOL	Replacement required, End-of-life
	ZW_ALRM_EVT_SMOKE_MAINTNC_PLAN	Maintenance required, planned periodic inspection
	ZW_ALRM_EVT_SMOKE_MAINTNC_DUST	Maintenance required, dust in device
Carbon monoxide alarm	ZW_ALRM_EVT_CO_L	Carbon monoxide detected with location
	ZW_ALRM_EVT_CO	Carbon monoxide detected
	ZW_ALRM_EVT_CO_TEST	Carbon monoxide test. Event parameter value: 1=OK, 2=Failed
	ZW_ALRM_EVT_CO_REPLA	Replacement required
	ZW_ALRM_EVT_CO_REPLA_EOL	Replacement required, End-of-life
	ZW_ALRM_EVT_CO_SILENCED	Alarm silenced
	ZW_ALRM_EVT_CO_MAINTNC_PLAN	Maintenance required, planned periodic inspection
Carbon dioxide alarm	ZW_ALRM_EVT_CO2_L	Carbon dioxide detected with location
	ZW_ALRM_EVT_CO2	Carbon dioxide detected
	ZW_ALRM_EVT_CO2_TEST	Carbon dioxide test. Event parameter value: 1=OK, 2=Failed
	ZW_ALRM_EVT_CO2_REPLA	Replacement required
	ZW_ALRM_EVT_CO2_REPLA_EOL	Replacement required, End-of-life
	ZW_ALRM_EVT_CO2_SILENCED	Alarm silenced
	ZW_ALRM_EVT_CO2_MAINTNC_PLAN	Maintenance required, planned periodic inspection
Heat alarm	ZW_ALRM_EVT_OVERHEAT_L	Overheat detected with location info
	ZW_ALRM_EVT_OVERHEAT	Overheat detected
	ZW_ALRM_EVT_TEMP_RISE_L	Rapid temperature rise detected with location
	ZW_ALRM_EVT_TEMP_RISE	Rapid temperature rise detected
	ZW_ALRM_EVT_UNDRHEAT_L	Underheat detected with location
	ZW_ALRM_EVT_UNDRHEAT	Underheat detected

	ZW_ALRM_EVT_HT_TEST	Heat alarm test
	ZW_ALRM_EVT_HT_REPLA_EOL	Replacement required, End-of-life
	ZW_ALRM_EVT_HT_SILENCED	Alarm silenced
	ZW_ALRM_EVT_HT_MAINTNC_DUST	Maintenance required, dust in device
	ZW_ALRM_EVT_HT_MAINTNC_PLAN	Maintenance required, planned periodic inspection
	ZW_ALRM_EVT_RAPID_TEMP_FALL_L	Rapid temperature fall with location info
	ZW_ALRM_EVT_RAPID_TEMP_FALL	Rapid temperature fall
Water alarm	ZW_ALRM_EVT_LEAK_L	Water leak detected with location
	ZW_ALRM_EVT_LEAK	Water leak detected
	ZW_ALRM_EVT_LVL_L	Water level dropped detected with location
	ZW_ALRM_EVT_LVL	Water level dropped detected
	ZW_ALRM_EVT_REPLACE_WATER_FILTER	Replace Water Filter
	ZW_ALRM_EVT_WATER_FLOW	Water flow. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold, 4=max
	ZW_ALRM_EVT_WATER_PRESSURE	Water pressure. Event parameter meaning same as those for water flow.
	ZW_ALRM_EVT_WATER_TEMP	Water temperature. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold
	ZW_ALRM_EVT_WATER_LEVEL	Water level. Event parameter meaning same as those for water temperature.
	ZW_ALRM_EVT_SUMP_PUMP_ACTV	Sump pump active
	ZW_ALRM_EVT_SUMP_PUMP_FAIL	Sump pump failure
Lock access control alarm	ZW_ALRM_EVT_MANUAL_LCK	Manual lock operation
	ZW_ALRM_EVT_MANUAL_ULCK	Manual unlock operation
	ZW_ALRM_EVT_RF_LCK	RF lock operation
	ZW_ALRM_EVT_RF_ULCK	RF unlock operation
	ZW_ALRM_EVT_KEYPAD_LCK	Keypad lock operation with user identifier
	ZW_ALRM_EVT_KEYPAD_ULCK	Keypad unlock operation with user identifier
	ZW_ALRM_EVT_MANUAL_NOT_FUL_LCK	Manual not fully locked operation

	ZW_ALRM_EVT_RF_NOT_FUL_LCK	RF not fully locked operation
	ZW_ALRM_EVT_AUTO_LCK	Auto lock locked operation
	ZW_ALRM_EVT_AUTO_NOT_FUL_OPER	Auto lock not fully operation
	ZW_ALRM_EVT_LCK_JAMMED	Lock jammed
	ZW_ALRM_EVT_ALL_CODE_DEL	All user codes deleted
	ZW_ALRM_EVT_1_CODE_DEL	Single user code deleted
	ZW_ALRM_EVT_CODE_ADDED	New user code added
	ZW_ALRM_EVT_CODE_DUP	New user code not added due to duplicate code
	ZW_ALRM_EVT_KEYPAD_DISABLED	Keypad temporary disabled
	ZW_ALRM_EVT_KEYPAD_BUSY	Keypad busy
	ZW_ALRM_EVT_NEW_PROG_CODE	New program code entered - unique code for lock configuration
	ZW_ALRM_EVT_USR_CODE_LIMIT	Manually enter user access code exceeds code limit
	ZW_ALRM_EVT_RF_ULCK_INVLD_CODE	Unlock by RF with invalid user code
	ZW_ALRM_EVT_RF_LCK_INVLD_CODE	Locked by RF with invalid user code
	ZW_ALRM_EVT_WINDOW_DOOR_OPEN	Window/door is open
	ZW_ALRM_EVT_WINDOW_DOOR_CLOSED	Window/door is closed
	ZW_ALRM_EVT_WIN_DR_HDL_OPEN	Window/door handle is open
	ZW_ALRM_EVT_WIN_DR_HDL_CLOSED	Window/door handle is closed
	ZW_ALRM_EVT_USR_CODE_VIA_KEYPAD	User Code entered via keypad (with event param: ZW_ALRM_PARAM_USRID)
Burglar alarm or home security	ZW_ALRM_EVT_INTRUSION_L	Intrusion detected with location
	ZW_ALRM_EVT_INTRUSION	Intrusion detected
	ZW_ALRM_EVT_TMPR_COVER	Tampering, product covering removed
	ZW_ALRM_EVT_TMPR_CODE	Tampering, Invalid Code
	ZW_ALRM_EVT_GLASS_L	Glass breakage detected with location
	ZW_ALRM_EVT_GLASS	Glass breakage detected
	ZW_ALRM_EVT_MOTION_DET_L	Motion detected with location info
	ZW_ALRM_EVT_MOTION_DET	Motion detected with unknown location info
	ZW_ALRM_EVT_TMPR_MOVED	Tampering, product moved

	ZW_ALRM_EVT_IMPACT	Impact detected. This event indicates that the node has detected an excessive amount of pressure or that an impact has occurred on the product itself
	ZW_ALRM_EVT_MAGNET_FIELD	Magnetic field interference detected. This state is used to indicate that magnetic field disturbance has been detected and the product functionality may not work reliably
Power management alarm	ZW_ALRM_EVT_POWER	Power has been applied
	ZW_ALRM_EVT_AC_OFF	AC mains disconnected
	ZW_ALRM_EVT_AC_ON	AC mains re-connected
	ZW_ALRM_EVT_SURGE	Surge Detection
	ZW_ALRM_EVT_VOLT_DROP	Voltage Drop/Drift detected
	ZW_ALRM_EVT_OVER_CURRENT	Over-current detected
	ZW_ALRM_EVT_OVER_VOLT	Over-voltage detected
	ZW_ALRM_EVT_OVER_LOAD	Over-load detected
	ZW_ALRM_EVT_LOAD_ERR	Load error
	ZW_ALRM_EVT_REPLACE_BATT_SOON	Replace battery soon
	ZW_ALRM_EVT_REPLACE_BATT_NOW	Replace battery now
	ZW_ALRM_EVT_BATT_CHARGING	Battery is charging
	ZW_ALRM_EVT_BATT_CHARGED	Battery is fully charged
	ZW_ALRM_EVT_CHARGE_BATT_SOON	Charge battery soon
	ZW_ALRM_EVT_CHARGE_BATT_NOW	Charge battery now
	ZW_ALRM_EVT_BKUP_BATT_LOW	Back-up battery is low
	ZW_ALRM_EVT_BATT_FLUID_LOW	Battery fluid is low
	ZW_ALRM_EVT_BKUP_BATT_DISCONN	Back-up battery disconnected
System alarm	ZW_ALRM_EVT_HW	System hardware failure
	ZW_ALRM_EVT_SW	System software failure
	ZW_ALRM_EVT_HW_OEM_CODE	System hardware failure with OEM proprietary failure code
	ZW_ALRM_EVT_SW_OEM_CODE	System software failure with OEM proprietary failure code
	ZW_ALRM_EVT_HEARTBEAT	Heartbeat. The Heartbeat event may be issued by a device to advertise that the device is still alive or to notify its presence
	ZW_ALRM_EVT_TMPR_COVER_RMV	Tampering, product cover removed
	ZW_ALRM_EVT_EMGCY_SHUTOFF	Emergency shutoff
	ZW_ALRM_EVT_DIGITAL_HI	Digital input high state
	ZW_ALRM_EVT_DIGITAL_LOW	Digital input low state

	ZW_ALRM_EVT_DIGITAL_OPEN	Digital input open. This state represents a generic digital input that is left open (not connected to anything)
Emergency alarm	ZW_ALRM_EVT_POLICE	Contact police
	ZW_ALRM_EVT_FIRE	Contact fire service
	ZW_ALRM_EVT_MEDICAL	Contact medical service
	ZW_ALRM_EVT_PANIC	Panic alert. This event is used to indicate that a panic situation or emergency occured
Alarm clock	ZW_ALRM_EVT_WKUP	Wake up
	ZW_ALRM_EVT_TIMER_ENDED	Timer ended
	ZW_ALRM_EVT_TIME_REMAINING	Time remaining. Event parameter value (3 bytes): byte-0 unit = hours, byte-1 unit = minutes, byte-2 unit = seconds
Barrier	ZW_ALRM_EVT_BARRIER_INIT	Barrier performing Initialization process. Event parameter value: 0=Completed, 0xFF=In progress
	ZW_ALRM_EVT_BARRIER_OP_FORCE_EXCEED	Barrier operation (Open / Close) force has been exceeded.
	ZW_ALRM_EVT_BARRIER_MOTOR_TIME_EXCEED	Barrier motor has exceeded manufacturer's operational time limit. Event parameter value: 0 to 0x7F = 0 to 127 seconds; 0x80 to 0xFE = 1 to 127 minutes
	ZW_ALRM_EVT_BARRIER_MECHANIC_EXCEED	Barrier operation has exceeded physical mechanical limits. (For example: barrier has opened past the open limit)
	ZW_ALRM_EVT_BARRIER_OP_FAILED	Barrier unable to perform requested operation due to UL requirements.
	ZW_ALRM_EVT_BARRIER_OP_DISABLED	Barrier Unattended operation has been disabled per UL requirements
	ZW_ALRM_EVT_BARRIER_MALFUNC	Barrier failed to perform Requested operation, device malfunction
	ZW_ALRM_EVT_BARRIER_VACAT_MODE	Barrier Vacation Mode. Event parameter value: 0=disabled, 0xFF=enabled
	ZW_ALRM_EVT_BARRIER_BEAM_OBST	Barrier Safety Beam Obstacle. Event parameter value: 0=no obstruction, 0xFF=obstruction

	ZW_ALRM_EVT_BARRIER_SNR_ERR	Barrier Sensor Not Detected / Supervisory Error. Event parameter value: 0=sensor not defined, 1 to 0xFF=sensor ID
	ZW_ALRM_EVT_BARRIER_LOW_BATT	Barrier Sensor Low Battery Warning. Event parameter value: 0=sensor not defined, 1 to 0xFF=sensor ID
	ZW_ALRM_EVT_BARRIER_SHORT	Barrier detected short in Wall Station wires
	ZW_ALRM_EVT_BARRIER_NON_Z-WAVE	Barrier associated with non-Z-Wave remote control
Appliance	ZW_ALRM_EVT_PROG_STARTED	Program started
	ZW_ALRM_EVT_PROG_IN_PROGRESS	Program in progress
	ZW_ALRM_EVT_PROG_END	Program completed
	ZW_ALRM_EVT_REPLACE_FILTER	Replace main filter
	ZW_ALRM_EVT_SET_TEMP_ERR	Failure to set target temperature
	ZW_ALRM_EVT_SUPPLYING_WATER	Supplying water
	ZW_ALRM_EVT_WATER_SUPPLY_ERR	Water supply failure
	ZW_ALRM_EVT_BOILING	Boiling
	ZW_ALRM_EVT_BOILING_ERR	Boiling failure
	ZW_ALRM_EVT_WASHING	Washing
	ZW_ALRM_EVT_WASHING_ERR	Washing failure
	ZW_ALRM_EVT_RINSING	Rinsing
	ZW_ALRM_EVT_RINSING_ERR	Rinsing failure
	ZW_ALRM_EVT_DRAINING	Draining
	ZW_ALRM_EVT_DRAINING_ERR	Draining failure
	ZW_ALRM_EVT_SPINNING	Spinning
	ZW_ALRM_EVT_SPINNING_ERR	Spinning failure
	ZW_ALRM_EVT_DRYING	Drying
	ZW_ALRM_EVT_DRYING_ERR	Drying failure
	ZW_ALRM_EVT_FAN_ERR	Fan failure
	ZW_ALRM_EVT_COMPRESSOR_ERR	Compressor failure
Home Health	ZW_ALRM_EVT_LEAVING_BED	Leaving Bed
	ZW_ALRM_EVT_SITTING_ON_BED	Sitting on bed
	ZW_ALRM_EVT_LYING_ON_BED	Lying on bed
	ZW_ALRM_EVT_POSTURE_CHANGED	Posture changed
	ZW_ALRM_EVT_SITTING_ON_BED_EDGE	Sitting on edge of bed
	ZW_ALRM_EVT_VOLATILE_ORGANIC_COMPOUND_LVL	Volatile Organic Compound level. Event parameter value (pollution level): 1=clean, 2=Slightly polluted, 3=Moderately polluted, 4=Highly polluted

	ZW_ALRM_EVT_SLEEP_APNEA	Sleep apnea detected. Event parameter value (breath level): 1=low breath, 2=No breath at all
	ZW_ALRM_EVT_SLEEP_STAGE_0	Sleep stage 0 detected: Dreaming/REM
	ZW_ALRM_EVT_SLEEP_STAGE_1	Sleep stage 1 detected: Light sleep, non-REM 1
	ZW_ALRM_EVT_SLEEP_STAGE_2	Sleep stage 2 detected: Medium sleep, non-REM 2
	ZW_ALRM_EVT_SLEEP_STAGE_3	Sleep stage 3 detected: Deep sleep, non-REM 3
Siren	ZW_ALRM_EVT_SIREN_ACT	Siren Active
Water Valve	ZW_ALRM_EVT_VALVE_OP	Valve Operation. Event parameter value: 0=closed, 1=open
	ZW_ALRM_EVT_MSTR_VALVE_OP	Master Valve Operation. Event parameter value: 0=closed, 1=open
	ZW_ALRM_EVT_SHORT_CCT	Valve Short Circuit
	ZW_ALRM_EVT_MSTR_SHORT_CCT	Master Valve Short Circuit
	ZW_ALRM_EVT_CUR_ALRM	Valve Current Alarm. Event parameter value: 1=no data, 2=below low threshold, 3=above high threshold, 4=max
	ZW_ALRM_EVT_MSTR_CUR_ALRM	Master Valve Current Alarm. Event parameter value same as Valve Current Alarm
Weather	ZW_ALRM_EVT_RAIN	Rain
	ZW_ALRM_EVT_MOIST	Moisture
	ZW_ALRM_EVT_FREEZE	Freeze
Irrigation	ZW_ALRM_EVT_SCHED_START	Schedule Started. Event parameter value is schedule ID
	ZW_ALRM_EVT_SCHED_FIN	Schedule Finished. Event parameter value is schedule ID
	ZW_ALRM_EVT_VLV_TBL_RUN_START	Valve Table Run Started. Event parameter value is valve table ID
	ZW_ALRM_EVT_VLV_TBL_RUN_FIN	Valve Table Run Finished. Event parameter value is valve table ID
	ZW_ALRM_EVT_DEV_UNCONFIG	Device is not Configured
Gas	ZW_ALRM_EVT_COMBUST_GAS_DET_L	Combustible Gas detected with location info
	ZW_ALRM_EVT_COMBUST_GAS_DET	Combustible Gas detected with unknown location info

	ZW_ALRM_EVT_TOXIC_GAS_DET_L	Toxic Gas detected with location info
	ZW_ALRM_EVT_TOXIC_GAS_DET	Toxic Gas detected with unknown location info
	ZW_ALRM_EVT_GAS_ALRM_TEST	Gas Alarm Test
	ZW_ALRM_EVT_GAS_ALRM_REPLACE	Replacement Required
Pest Control	ZW_ALRM_EVT_TRAP_ARMED_L	Trap armed with location info
	ZW_ALRM_EVT_TRAP_ARMED	Trap armed
	ZW_ALRM_EVT_TRAP_REARM_REQ_L	Trap re-arm required with location info
	ZW_ALRM_EVT_TRAP_REARM_REQ	Trap re-arm required
	ZW_ALRM_EVT_PEST_DET_L	Pest detected with location info
	ZW_ALRM_EVT_PEST_DET	Pest detected
	ZW_ALRM_EVT_PEST_EXTERMINATED_L	Pest exterminated with location info
	ZW_ALRM_EVT_PEST_EXTERMINATED	Pest exterminated
Light Sensor	ZW_ALRM_EVT_LIGHT_DET	Light detected
	ZW_ALRM_EVT_COLOR_TRANS_DET	Light color transition detected
Water Quality Monitoring	ZW_ALRM_EVT_CHLORINE_ALRM	Chlorine alarm. Event parameter value: 1=Below low threshold, 2=Above high threshold
	ZW_ALRM_EVT_ACIDITY	Acidity (pH). Event parameter value: 1=Below low threshold, 2=Above high threshold, 3=Decreasing pH, 4=Increasing pH
	ZW_ALRM_EVT_OXIDATION_ALRM	Water Oxidation alarm. Event parameter value: 1=Below low threshold, 2=Above high threshold
	ZW_ALRM_EVT_CHLORINE_EMPTY	Chlorine empty
	ZW_ALRM_EVT_ACIDITY_EMPTY	Acidity empty
	ZW_ALRM_EVT_WATERFLOW_MEAS_SHORTAGE	Waterflow measuring station shortage detected
	ZW_ALRM_EVT_WATERFLOW_CLR_WTR_SHORTAGE	Waterflow clear water shortage detected
	ZW_ALRM_EVT_DISINFECT_ERR	Disinfection system error detected. Event parameter value (bit-mask): bit 0~3: System 1~4 disorder detected, bit 4~7: System 1~4 salt shortage
	ZW_ALRM_EVT_FILTER_CLEANING	Filter cleaning ongoing. Event parameter value: 1~255= Filter number

	ZW_ALRM_EVT_HEATING	Heating operation ongoing
	ZW_ALRM_EVT_FILTER_PUMP	Filter pump operation ongoing
	ZW_ALRM_EVT_FRESHWATER	Freshwater operation ongoing
	ZW_ALRM_EVT_DRY_PROTECT	Dry protection operation active
	ZW_ALRM_EVT_WATER_TANK_EMPTY	Water tank is empty
	ZW_ALRM_EVT_WATER_TANK_UNKNOWN	Water tank level is unknown
	ZW_ALRM_EVT_WATER_TANK_FULL	Water tank is full
	ZW_ALRM_EVT_COLLECTIVE_DISORDER	Collective disorder
Home Monitoring	ZW_ALRM_EVT_HOME_OCCUPIED_L	Home occupied with location info
	ZW_ALRM_EVT_HOME_OCCUPIED	Home occupied

11.9.2 zwif_alrm_get

Get the state of the Alarm/Notification device (push mode) or the pending notification (pull mode) through report callback function.

Table 358 – zwif_alrm_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle
vtype	int16_t	I	Vendor specific alarm type. Zero if this field is not used ; -1 to indicate "don't care" for cache get.
ztype	uint8_t	I	Z-Wave alarm type (ZW_ALRM_XXX). Zero if this field is not used; 0xFF = select a supported Notification Type (push mode), or retrieve the first alarm detection (pull mode).
evt	uint8_t	I	Event corresponding to Z-Wave alarm/Notification type (ZW_ALRM_EVT_XXX). Zero if this field is not used. This parameter is valid for push mode only
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

11.9.3 zwif_alrm_set

Set the state of the specified Z-Wave alarm type (push mode) or clear a persistent notification (pull mode).

Table 359 – zwif_alrm_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm interface handle.
ztype	uint8_t	I	Z-Wave alarm type (ZW_ALRM_XXX).
sts	uint8_t	I	Z-Wave alarm status. For alarm operating in push mode: 0= disable unsolicited report ; 0xFF= enable. For pull mode: 0=clear a persistent notification.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.9.4 zwif_alrm_sup_get

Get the supported alarm/notification types through report callback function.

Table 360 – zwif_alrm_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
cb	zwrep_alrm_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 361 – zwrep_alrm_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
have_vtype	uint8_t	I	Flag to indicate whether vendor specific alarm type supported. 1=supported; else 0=unsupported.
ztype_len	uint8_t	I	Size of ztype buffer.
ztype	uint8_t *	I	Buffer to store supported Z-Wave alarm/Notification types (ZW_ALRM_XXX).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.9.5 zwif_alrm_sup_cache_get

Get the supported alarm types from cache.

Table 362 – zwif_alrm_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm interface handle.
have_vtype	uint8_t *	O	Lag to indicate whether vendor specific alarm type supported. 1=supported; else 0=unsupported.
ztype_cnt	uint8_t *	O	Number of supported Z-Wave alarm types.
ztype_buf	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported Z-Wave alarm types (ZW_ALRM_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.9.6 zwif_alrm_sup_evt_get

Get the supported events of a specified alarm/notification type through report callback function.

Table 363 – zwif_alrm_sup_evt_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
ztype	uint8_t	I	Z-Wave alarm/Notification type (ZW_ALRM_XXX).

cb	zwrep_alrm_evt_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 364 – zwrep_alrm_evt_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
ztype	uint8_t	I	Z-Wave alarm/Notification type (ZW_ALRM_XXX).
evt_len	uint8_t	I	Size of evt buffer.
evt	uint8_t *	I	Buffer to store supported event of the alarm type specified in ztype.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.9.7 zwif_alrm_sup_evt_cache_get

Get the supported events of a specified alarm type from cache.

Table 365 – zwif_alrm_sup_evt_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm interface handle.
ztype	uint8_t	I	Z-Wave alarm type (ZW_ALRM_XXX).
evt_cnt	uint8_t *	O	Number of supported events.
evt_buf	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported events.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.9.8 zwif_alrm_vtype_sup_get

Get the supported vendor specific alarm types from device DB.

Table 366 – zwif_alrm_vtype_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm/Notification interface handle.
rec_head	if_rec_alarm_match_t**	O	Head of the alarm/Notification record link-list.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.10 Alarm Sensor Interface API

This corresponds to the Z-Wave Alarm Sensor CC.

11.10.1 zwif_alrm_snsr_rpt_set

Set up an alarm sensor report callback function.

Table 367 – zwif_alrm_snsr_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
rpt_cb	zwrep_alrm_snsr_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 368 – zwrep_alrm_snsr_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm sensor interface handle.
data	zw_alrm_snsr_t *	I	Alarm sensor report data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 369 – zw_alrm_snsr_t Structure

Attribute	Type	I/O	Description
src_node_id	uint8_t	O	Source node ID with the alarm condition. Not valid in Zensor Net.
type	uint8_t	O	Alarm sensor types. See ZW_ALRM_SNSR_TYPE_XXX
state	uint8_t	O	Sensor state: 0=no alarm; 0xFF=alarm; 1-99: alarm severity in percentage.
activetime	uint16_t	O	Alarm active time since last received report in seconds. Zero means this field must be ignored.

Table 370 – ZW_ALRM_SNSR_TYPE_XXX

Alarm Sensor Type	Description
ZW_ALRM_SNSR_TYPE_GP	General Purpose Alarm.
ZW_ALRM_SNSR_TYPE_SMOKE	Smoke Alarm.
ZW_ALRM_SNSR_TYPE_CO	CO Alarm.
ZW_ALRM_SNSR_TYPE_CO2	CO2 Alarm.
ZW_ALRM_SNSR_TYPE_HEAT	Heat Alarm.
ZW_ALRM_SNSR_TYPE_WATER_LEAK	Water Leak Alarm.
ZW_ALRM_SNSR_TYPE_1ST_SUP	Return first Alarm on supported list.

11.10.2 zwif_alrm_snsr_get

Get alarm sensor state through report callback.

Table 371 – zwif_alrm_snsr_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm sensor interface handle.
type	uint8_t	O	Preferred alarm sensor type. ZW_ALRM_SNSR_TYPE_XXX. If type equals to ZW_ALRM_SNSR_TYPE_1ST_SUP, the alarm sensor report will return the first supported sensor type report.

flag	int	O	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.10.3 zwif_alrm_snsr_sup_get

Get the supported alarm sensor types through report callback.

Table 372 – zwif_alrm_snsr_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm sensor interface handle.
cb	zwrep_alrm_snsr_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 373 – zwrep_alrm_snsr_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm sensor interface handle.
type_len	uint8_t	I	Size of alarm sensor type buffer.
type	uint8_t *	I	Buffer to store supported alarm sensor types (ZW_ALRM_SNSR_TYPE_XXX).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.10.4 zwif_alrm_snsr_sup_cache_get

Get the supported alarm sensor types from cache.

Table 374 – zwif_alrm_snsr_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Alarm sensor interface handle.
sup_snsr	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported sensors (ZW_ALRM_SNSR_TYPE_XXX).
snsr_cnt	uint8_t *	O	Supported sensor counts.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.11 Sensor Interface API

This corresponds to the Z-Wave Multi-Level Sensor CC.

11.11.1 zwif_sensor_rpt_set

Set up a multilevel sensor report callback.

Table 375 – zwif_sensor_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
cb	zwrep_sensor_fn	I	Sensor report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 376 – zwrep_sensor_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
data	zwsensor_t *	I	Sensor data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 377 – zwsensor_t Structure

Attribute	Type	I/O	Description
type	uint8_t	I	Sensory type ZW_SENSOR_TYPE_XXX.
precision	uint8_t	I	Number of decimal places.
unit	uint8_t	I	ZW_SENSOR_UNIT_XXX_YYY.
size	uint8_t	I	Number of bytes: 1, 2 or 4.
data	uint8_t[4]	I	Value at sensor (a signed number) with the first byte is the most significant byte.

Table 378 – ZW_SENSOR_TYPE_XXX, ZW_SENSOR_UNIT_XXX_YYY

Type	XXX	YYY	Units
Temperature	TEMP	CEL, FAHR	Celsius, Fahrenheit
General purpose	GP	CENT, NOM	Percentage, Nominal
Luminance	LUMIN	CENT, LX	Percentage, Lux
Power	POWER	W, BTUH	Watts, British thermal unit per hour
Humidity	HUMID	CENT, ABS	Percentage, g/m3
Velocity	VELO	MPS, MPH	Meters per second, miles per hour
Wind Direction	DIR	DEG	Degrees: 0-none, 360-north
Atmosphere	ATM	KPA, INS	Kilopascal, Inches of mercury
Barometer	BARO	KPA, INS	Kilopascal, Inches of mercury
Solar radiation	SLR	WM2	Watts per square meter
Dew point	DEW	CEL, FAHR	Celsius, Fahrenheit
Rain rate	RAIN	MMH, INH	Millimeters or inches per hour
Tide level	TIDE	M, FT	Meter, feet
Weight	WGT	KG, LBS	Kilograms, pounds
Voltage	VOLT	V, MV	Volts, mill volts
Current	AMP	A, MA	Amperes, mill amperes
Carbon dioxide	CO2	PPM	Parts per million
Air flow	AIR	M3H, F3M	Cubic meters per hour, cubic feet per minute
Tank	TANK	L, M3, GL	Liters, cubic meters, US gallons

Distance	DIST	M, CM, FT	Meters, centimeters, feet
Angle position	AGL	CENT, N, S	Percentage, degrees from north/south pole
Rotation	ROT	RPM, HZ	Revolutions per minute, Hertz
Water temperature	WTR_TEMP	CEL, FAHR	Celsius, Fahrenheit
Soil temperature	SOIL_TEMP	CEL, FAHR	Celsius, Fahrenheit
Seismic intensity	SEIS_INT	MERC, EMCRO, LIEDU, SHNDO	Mercalli, European Macroseismic, Liedu, Shindo
Seismic magnitude	SEIS_MAG	LOCAL, MOM, SWAVE, BWAVE	ML, MW, MS, MB
Ultraviolet	UV	INDEX	UV index
Electrical resistivity	ELEC_RES	OHMM	ohm metre
Electrical conductivity	ELEC_COND	SIEM	siemens per metre
Loudness	LOUDNESS	ABS, A_WT	dB, dBA
Moisture	MOIST	PERC, VOL_WTR, IMPD, WTR_ACT	Percentage, Volume water content (m3/m3), Impedance (k ohm), Water activity (aw)
Frequency	FREQ	HZ, kHZ	Hz - Must be used until 4.294967295 GHz, kHz- Must be used until 4.294967295 GHz
Time	TIME	SEC	Seconds
Target temperature	TGT_TEMP	CEL, FAHR	Celsius (C), Fahrenheit (F)
Particulate matter 2.5	PM_2_5	MM3, UGM3	mole per cubic meter (mol/m^3), absolute microgram/cubic meter (ug/m^3)
Formaldehyde CH2O-level	F_CH2O	MM3	mole per cubic meter (mol/m^3)
Radon Concentration	RAD_CONT	BQM3, PCIL	Becquerel/cubic meter (bq/m^3), picocuries/liter (pCi/L)
Methane Density CH4	METH_DENS	MM3	mole per cubic meter (mol/m^3)
Volatile Organic Compound	VOC	MM3	mole per cubic meter (mol/m^3)
Carbon Monoxide CO-level	CO_LVL	MM3	mole per cubic meter (mol/m^3)
Soil Humidity	SOIL_HUMD	PERC	Percentage value
Soil Reactivity	SOIL_REAC	PH	acidity (pH)

Soil Salinity	SOIL_SAL	MM3	mole per cubic meter (mol/m^3)
Heart Rate	HEART_RT	BPM	beats/minute (Bpm)
Blood Pressure	BLOOD_PRS	SYS, DIA	Systolic mmHg (upper number), Diastolic mmHg (lower number)
Muscle Mass	MUSCLE_MS	KG	kilogram (kg)
Fat Mass	FAT_MS	KG	kilogram (kg)
Bone Mass	BONE_MS	KG	kilogram (kg)
Total Body Water	TBW	KG	kilogram (kg)
Basic Metabolic Rate	BMR	J	joule (J)
Body Mass Index	BMI	IDX	BMI Index
Acceleration, X-axis	ACCEL_X	MS2	(m/s^2)
Acceleration, Y-axis	ACCEL_Y	MS2	(m/s^2)
Acceleration, Z-axis	ACCEL_Z	MS2	(m/s^2)
Smoke Density	SMOKE_DEN	PERC	Percentage value
Water Flow	WATER_FLW	LHR	liter/hour (l/h)
Water Pressure	WATER_PRS	KPA	kilopascal (kPa)
RF Signal Strength	RF_SGN	RSSI, DBM	RSSI (Percentage value), (dBm)
Particulate Matter 10	PM_10	MOLE, UG	Mole per cubic meter (mol/m3), Microgram per cubic meter (µg/m3)
Respiratory rate	RESPI_RATE	BPM	Breaths per minute (bpm)
Relative Modulation level	REL_MOD	PERC	Percentage value (%)
Boiler water temperature	BOILER_WTR_TEMP	TEMP_C	Celsius (C)
Domestic Hot Water (DHW) temperature	DHW_TEMP	TEMP_C	Celsius (C)
Outside temperature	OUTSIDE_TEMP	TEMP_C	Celsius (C)
Exhaust temperature	EXHAUST_TEMP	TEMP_C	Celsius (C)
Water Chlorine level	WATER_CHLOR_LVL	MGL	Milligram per liter (mg/l)
Water acidity	WATER_ACID	PH	Acidity (pH)
Water Oxidation reduction potential	WATER_OXI_RED	MV	MilliVolt (mV)

11.11.2 zwif_sensor_get

Solicit a report of the current state of the multilevel sensor.

Table 379 – zwif_sensor_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
type[1]	uint8_t	I	Preferred sensor type, ZW_SENSOR_TYPE_XXX. If type equals zero, the sensor report will return the factory default sensor type.
unit[2]	uint8_t	I	Preferred sensor unit, ZW_SENSOR_UNIT_XXX. This parameter is ignored if type=0.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Note 1 and 2 : *Preferred sensor type and unit are not guaranteed to be returned in the report callback. It depends on the multilevel sensor CC version number and the whether the device supports them.*

11.11.3 zwif_sensor_sup_get

Get the supported sensor types through report callback function.

Table 380 – zwif_sensor_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
cb	zwrep_sensor_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when the cache is unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 381 – zwrep_sensor_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
type_len	uint8_t	I	Size of sensor type buffer.
type	uint8_t *	I	Buffer to store supported sensor types.(ZW_SENSOR_TYPE_XXX)
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.11.4 zwif_sensor_unit_get

Get the supported sensor units through report callback function.

Table 382 – zwif_sensor_unit_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
sensor_type	uint8_t	I	Sensor type.
cb	zwrep_sensor_unit_fn	I	Report callback function.

cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 383 – zwrep_sensor_unit_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
sensor_type	uint8_t	I	Sensor type, ZW_SENSOR_TYPE_XXX.
sensor_unit_msk	uint8_t	I	Bitmask of units supported for the sensor_type, ZW_SENSOR_UNIT_XXX.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.11.5 zwif_sensor_unit_cache_get

Get supported sensor units from cache.

Table 384 – zwif_sensor_unit_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
sensor_type	uint8_t	I	Sensor type.
unit_cnt	uint8_t *	O	Supported sensor unit count.
sup_unit	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported sensor units (ZW_SENSOR_UNIT_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.11.6 zwif_sensor_sup_cache_get

Get supported sensor types from cache.

Table 385 – zwif_sensor_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Sensor interface handle.
sup_snsr	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported Sensor types (ZW_SENSOR_TYPE_XXX).
snsr_cnt	uint8_t *	O	Supported sensor counts.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 386 – if_sensor_data_t Structure

Attribute	Type	I/O	Description
sensor_type	uint8_t	I	Supported sensor type ZW_SENSOR_TYPE_XXX.
sensor_unit	uint8_t	I	Supported sensor units (bit-mask). See ZW_SENSOR_UNIT_XXX_YYY.

11.12 Central Scene Interface API

This corresponds to the Z-Wave Central Scene CC.

11.12.1 zwif_csc_rpt_set

Set up a central scene notification report callback function.

Table 387 – zwif_csc_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
rpt_cb	zwrep_csc_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 388 – zwrep_csc_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
data	zwcsc_notif_t *	I	Central scene notification data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 389 – zwcsc_notif_t Structure

Attribute	Type	I/O	Description
seqNo	uint8_t	O	Sequence number. Incremented each time a new report is issued by the device.
keyAttr	uint8_t	O	Key attribute. ZW_CSC_KEY_ATTRIB_XXX.
sceneNo	uint8_t	O	Scene Number. Actual scene identifier.
slow_rfsh	uint8_t	O	Slow refresh of "Key Held Down" notification. Non-zero=enable; 0=disable. If **disabled**: A new "Key Held Down" notification MUST be sent by CSC every 200ms until the key is released. If not receiving a new "Key Held Down" notification within 400ms after the most recent "Key Held Down" notification, a "Key Release" notification must be assumed. If **enabled**: A new "Key Held Down" notification MUST be sent by CSC every 55 seconds until the key is released. If not receiving a new "Key Held Down" notification within 60 seconds after the most recent "Key Held Down" notification, a "Key Release" notification must be assumed.

Table 390 – ZW_CSC_KEY_ATTRIB_XXX

Thermostat Setpoint Type	Description
ZW_CSC_KEY_ATTRIB_KEY_PRESSED_1_TIME	A key is pressed and released before time out.
ZW_CSC_KEY_ATTRIB_KEY_RELEASED	A key is released. Termination of a Key Held Down sequence.
ZW_CSC_KEY_ATTRIB_KEY_HELDDOWN	A key is pressed and not released before time out. Event used to signal continuation of key held down operation.
ZW_CSC_KEY_ATTRIB_KEY_PRESSED_2_TIME	A key is pressed twice and no more key presses follow.
ZW_CSC_KEY_ATTRIB_KEY_PRESSED_3_TIME	A key is pressed 3 times and no more key presses follow.
ZW_CSC_KEY_ATTRIB_KEY_PRESSED_4_TIME	A key is pressed 4 times and no more key presses follow.
ZW_CSC_KEY_ATTRIB_KEY_PRESSED_5_TIME	A key is pressed 5 times and no more key presses follow.

11.12.2 zwif_csc_sup_get

Get the maximum number of supported scenes and the key attributes supported per scene through report callback.

Table 391– zwif_csc_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
cb	zwrep_csc_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 392 – zwrep_csc_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
scene_cnt	uint8_t	I	maximum number of supported scenes.
sameKA	uint8_t	I	it indicates if all scenes are supporting the same Key Attributes. 1=same, 0=different.
KA_array_len	uint8_t	I	length/width of key attribute array. For v1 COMMAND_CLASS_CENTRAL_SCENE, KA_array_len will be 0.
KA_array	uint8_t *	I	key attribute array. It contains all the supported key attributes for each scene. When KA_array_len is 0, KA_array should be ignored. If sameKA is 1, KA_array is a one-dimensional array with length indicated by 'KA_array_len'.

			All the scenes support the same set of Key Attribute. For each row of the array, the first element is the number of valid key attributes given in this row. If a particular scene does not support any key attribute, the first element for that row will be 0. If sameKA is 0, KA_array is a two-dimensional array with size KA_array_len x scene_cnt. For each row of the array, the first element is the number of valid key attributes given in this row. If a particular scene does not support any key attribute, the first element for that row will be 0.
slow_rfsh	uint8_t	I	status for slow refresh of Key Held Down notification. Non-zero=enable; 0=disable.
valid	int	I	validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

KA Array returned in the following format:

v1 interface -- sameKA 1, scene_cnt 5, KA_array_len 0, KA Array is NULL

V2 interface or above --

Example 1:

sameKA 1, scene_cnt 5, KA_array_len 4 KA_array: [3] [0] [1] [2]

Example 2:

sameKA 0, scene_cnt 5, KA_array_len 4

KA_array:

[3] [0] [1] [2] // 3 valid KAs in this row

[2] [0] [5] [X] //Only have 2 valid KA in this row

[0] [X] [X] [X] //No valid KA in this row

[3] [0] [3] [5]

[1] [1] [X] [X] //Only have 1 valid KA in this row

11.12.3 zwif_csc_cfg_rpt_set

Setup configuration for scene configuration report callback function.

Table 393– zwif_csc_cfg_rpt_set Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Central scene interface handle.
rpt_cb	zwrep_csc_cfg_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 394 – zwrep_csc_cfg_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
slow_refresh	int	I	Status for slow refresh of Key Held Down notification. Non-zero=enable; 0=disable.
ts	time_t	I	time stamp of when the report is received.

11.12.4 zwif_csc_cfg_get

Get configuration for scene notifications report through report callback.

Table 395– zwif_csc_cfg_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.12.5 zwif_csc_cfg_set

Configure the use of slow refresh.

Table 396– zwif_csc_cfg_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Central scene interface handle.
slow_refresh	int	I	Flag to enable slow refresh of Key Held Down notification. Non-zero=enable; 0=disable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.13 Pulse Meter Interface API

This corresponds to the Z-Wave Pulse Meter CC.

11.13.1 zwif_pulsemeter_rpt_set

Setup a pulse meter report callback function.

Table 397 – zwif_pulsemeter_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Pulse meter interface handle.
rpt_cb	zwrep_pulsemeter_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 398 – zwrep_pulsemeter_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Pulse meter interface handle.
cnt	uint32_t	I	Number of pulses detected in the device.

11.13.2 zwif_pulsemeter_get

Get pulse meter reading through report callback function.

Table 399 – zwif_pulsemeter_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Pulse meter interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.14 Meter Interface API

This corresponds to the Z-Wave Meter CC.

11.14.1 zwif_meter_rpt_set

Set up a meter report callback function.

Table 400 – zwif_meter_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
rpt_cb	zwrep_meter_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 401 – zwrep_meter_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
data	zwmeter_dat_t *	I	Current value and unit of the meter.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 402 – zwmeter_dat_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Meter type ZW_METER_TYPE_XXX.
precision	uint8_t	O	Decimal places of the value.
unit	uint8_t	O	Meter unit ZW_METER_UNIT_XXX.
size	uint8_t	O	Data size: (1,2,or 4 bytes).

data	uint8_t[4]	O	Meter value (a signed number) with the first byte is the most significant byte.	
delta_time	uint16_t	O	Elapsed time in seconds between the 'Meter Value' and the 'Previous Meter Value' measurements. If delta_time = 0, it means no 'Previous Meter Value' measurement.	
prv_data	uint8_t[4]	O	Previous meter data (a signed number) with the first byte is the most significant byte. This field is valid only if delta_time is greater than 0.	
rate_type	uint8_t	O	rate type: ZW_METER_RATE_XXX where:	
			ZW_METER_RATE_IMPORT	Import: Meter Value is a consumed measurement.
			ZW_METER_RATE_EXPORT	Export: Meter Value is a produced measurement.
			ZW_METER_RATE_IMPORT_EXPORT	Both Import and Export. Must not be used in zwif_meter_get().

Table 403 – ZW_METER_TYPE_XXX and ZW_METER_UNIT_XXX

Meter Type	Meter Unit	Unit
ZW_METER_TYPE_ELEC	ZW_METER_UNIT_ELEC_KWH	kWh
	ZW_METER_UNIT_ELEC_KVAH	kVAh
	ZW_METER_UNIT_ELEC_W	W
	ZW_METER_UNIT_ELEC_PULSE	Pulse count
	ZW_METER_UNIT_ELEC_V	V
	ZW_METER_UNIT_ELEC_A	A
	ZW_METER_UNIT_ELEC_PF	Power factor
	ZW_METER_UNIT_ELEC_KVAR	KVar
	ZW_METER_UNIT_ELEC_KVARH	KVarh
ZW_METER_TYPE_GAS	ZW_METER_UNIT_GAS_CM	Cubic meters
	ZW_METER_UNIT_GAS_CF	Cubic feet
	ZW_METER_UNIT_GAS_PULSE	Pulse count
ZW_METER_TYPE_WATER	ZW_METER_UNIT_WATER_CM	Cubic meters
	ZW_METER_UNIT_WATER_CF	Cubic feet
	ZW_METER_UNIT_WATER_GAL	US gallons
	ZW_METER_UNIT_WATER_PULSE	Pulse count
ZW_METER_TYPE_HEAT	ZW_METER_UNIT_HEAT_KWH	kWh
ZW_METER_TYPE_COOL	ZW_METER_UNIT_COOL_KWH	kWh

11.14.2 zwif_meter_get

Get the meter reading through report callback function.

Table 404 – zwif_meter_get Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Meter interface handle
unit	uint16_t	I	Preferred unit (ZW_METER_UNIT_XXX). The report may not return the preferred unit if the device doesn't support it.
rate_type	uint8_t	I	Preferred rate type (ZW_METER_RATE_IMPORT or ZW_METER_RATE_EXPORT or 0 for no preference). The report may not return the preferred rate type if the device doesn't support it.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

11.14.3 zwif_meter_sup_get

Get information on the meter capabilities through report callback function. Older versions of target nodes may not support this function.

Table 405 – zwif_meter_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
cb	zwrep_meter_sup_fn	I	Callback function to receive the meter capabilities report.
cache	int	I	Flag: To get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 406 – zwrep_meter_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
meter_cap	zwmeter_cap_t *	I	Meter capabilities.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 407 – zwmeter_cap_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Meter type ZW_METER_TYPE_XXX
reset_cap	uint8_t	O	Capability to reset all accumulated values stored in the meter device. 1=reset supported. 0=not supported.
rate_type	uint8_t	O	Rate type ZW_METER_RATE_XXX or 0 if the supported rate type information is unavailable.
unit_sup	uint16_t	O	Supported unit bit-mask : ZW_METER_SUP_UNIT_XXX . When the relevant bit is 1, the unit is supported; else it is not supported.

Table 408 – ZW_METER_SUP_UNIT_XXX

Meter Type	Meter Supported Unit Bit-mask	Unit
Electric	ZW_METER_SUP_UNIT_ELEC_KWH	kWh
	ZW_METER_SUP_UNIT_ELEC_KVAH	kVAh
	ZW_METER_SUP_UNIT_ELEC_W	W
	ZW_METER_SUP_UNIT_ELEC_PULSE	Pulse count
	ZW_METER_SUP_UNIT_ELEC_V	V
	ZW_METER_SUP_UNIT_ELEC_A	A
	ZW_METER_SUP_UNIT_ELEC_PF	power factor
	ZW_METER_SUP_UNIT_ELEC_KVAR	kVar
	ZW_METER_SUP_UNIT_ELEC_KVARH	kVarh
Gas	ZW_METER_SUP_UNIT_GAS_CM	Cubic meters
	ZW_METER_SUP_UNIT_GAS_CF	Cubic feet
	ZW_METER_SUP_UNIT_GAS_PULSE	Pulse count
Water	ZW_METER_SUP_UNIT_WATER_CM	Cubic meters
	ZW_METER_SUP_UNIT_WATER_CF	Cubic feet
	ZW_METER_SUP_UNIT_WATER_GAL	U.S. gallons
	ZW_METER_SUP_UNIT_WATER_PULSE	Pulse count

11.14.4 zwif_meter_sup_cache_get

Get information on the meter capabilities from cache.

Table 409 – zwif_meter_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
meter_cap	zwmeter_cap_t *	I	Meter capabilities.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.14.5 zwif_meter_reset

Reset all accumulated values stored in the meter device.

Table 410 – zwif_meter_reset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.14.6 zwif_meter_set_admin

Set meter admin name.

Table 411 – zwif_meter_set_admin Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle
name	char *	I	Admin number

11.14.7 zwif_meter_get_desc

Get meter admin name through report callback function.

Table 412 – zwif_meter_get_desc Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
cb	zwrep_meterd_fn	I	Callback function to receive the meter descriptor report.
cache	int	I	Flag: To get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 413 – zwrep_meterd_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Meter interface handle.
meter	zwmeter_t *	I	Meter descriptor.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 414 – zwmeter_t Structure

Attribute	Type	I/O	Description
caps	uint8_t	O	ZW_METER_CAP_xxx or-mask: ZW_METER_CAP_MON = ID and admin number available. ZW_METER_CAP_CFG = admin number can be set.
id	char[33]	O	ID, used for identification of customer and metering device.
admin	char[33]	O	Admin number used to identify customer.

11.15 Door Lock Interface API

This corresponds to the Z-Wave Door Lock CC.

11.15.1 zwif_dlck_op_rpt_set

Set up a door lock operation report callback function.

Table 415 – zwif_dlck_op_rpt_set Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Door lock interface handle.
rpt_cb	zwrep_dlck_op_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 416 – zwrep_dlck_op_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
op_sts	zwdlck_op_t *	I	Operation status.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 417 – zwdlck_op_t Structure

Attribute	Type	I/O	Description
mode	uint8_t	O	Current door lock mode (ZW_DOOR_XXX).
out_mode	uint8_t	O	Outside door handles mode. It's a 4-bit mask; bit=0 for disabled, bit=1 for enabled. When disabled, the actual handle cannot open the door locally. When enabled, the actual handle can open the door locally.
in_mode	uint8_t	O	Inside door handles mode. It's a 4-bit mask; bit=0 for disabled, bit=1 for enabled.
cond	uint8_t	O	Door condition (i.e. status). See ZW_COND_XXX_MASK.
tmout_min	uint8_t	O	The remaining time in minutes before the door lock will automatically be locked again. Value of 0xFE means timeout is not supported.
tmout_sec	uint8_t	O	The remaining time in seconds before the door lock will automatically be locked again. Value of 0xFE means timeout is not supported.
tgt_mode	uint8_t	O	Target door lock mode (ZW_DOOR_XXX).
dur	uint8_t	O	Duration: 0 = already at the target. 0x01~0x7F = seconds. 0x80~0xFD = 1~126 minutes. 0xFE = Unknown duration. 0xFF = reserved.

Table 418 – ZW_DOOR_XXX

Door lock operation mode	Description
ZW_DOOR_UNSEC	Door unsecured.
ZW_DOOR_UNSEC_TMOUT	Door unsecured with timeout. Fallback to secured mode after timeout has expired.
ZW_DOOR_UNSEC_IN	Door unsecured for inside door handles.
ZW_DOOR_UNSEC_IN_TMOUT	Door unsecured for inside door handles with timeout.
ZW_DOOR_UNSEC_OUT	Door unsecured for outside door handles.
ZW_DOOR_UNSEC_OUT_TMOUT	Door unsecured for outside door handles with timeout.

ZW_DOOR_UNKNOWN	Unknown state. This could happen while in transition to a new mode. Example: Bolt is not fully retracted/engaged.
ZW_DOOR_SEC	Door secured.

Table 419 – ZW_COND_XXX_MASK

Door condition bit-mask	Description
ZW_COND_DOOR_MASK	Door status bit-mask. Bit=0 means open; else closed.
ZW_COND_BOLT_MASK	Bolt status bit-mask. Bit=0 means locked; else unlocked.
ZW_COND_LATCH_MASK	Latch status bit-mask. Bit=0 means open; else closed.

11.15.2 zwif_dlck_op_get

Get the state of the door lock device through report callback function.

Table 420 – zwif_dlck_op_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.15.3 zwif_dlck_op_set

Set door lock operation.

Table 421 – zwif_dlck_op_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
mode	uint8_t	I	Operation mode (ZW_DOOR_XXX).
cb	zw_postset_fn	I	Optional post-set polling callback function. NULL if no callback required.
usr_param	void *	I	Optional user-defined parameter passed in callback.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.15.4 zwif_dlck_op_mset

Set door lock operation using multicast addressing if available

Table 422 – zwif_dlck_op_mset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Array of door lock interface handles.
ifd_cnt	uint8_t	I	Number of interfaces in "ifd" array. If value is 1, it is equivalent to calling zwif_dlck_op_set()with cb=NULL and usr_param=NULL
mode	uint8_t	I	Operation mode (ZW_DOOR_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.15.5 zwif_dlck_cfg_set

Set the configuration of the door lock device.

Table 423 – zwif_dlck_cfg_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
config	zwdlck_cfg_t *	I	Configuration.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 424 – zwdlck_cfg_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Door lock operation type (ZW_DOOR_OP_XXX) : ZW_DOOR_OP_CONST = Constant operation. ZW_DOOR_OP_TIMED = Timed operation.
out_sta	uint8_t	O	Outside door handles state. It's a 4-bit mask; bit=0 for disable, bit=1 for enable. When disabled, the actual handle cannot open the door locally. When enabled, the actual handle can open the door locally.
in_sta	uint8_t	O	Inside door handles state. It is a 4-bit mask; bit=0 for disable, bit=1 for enable.
tmout_min	uint8_t	O	Lock timeout in minutes. Valid value: 0 to 253. Value of 0xFE means timeout is not supported.
tmout_sec	uint8_t	O	Lock timeout in seconds. Valid value: 0 to 59. Value of 0xFE means timeout is not supported.
blk_to_blk	uint8_t	O	Indicate if the block-to-block functionality is enabled. Non-zero means enabled; zero means disabled.
twist_asst	uint8_t	O	Indicate if the twist assist functionality is enabled. Non-zero means enabled; zero means disabled.
auto_rlck_tm	uint16_t	O	Time setting in seconds for auto-relock functionality. Zero means the functionality is disabled.
hold_rel_tm	uint16_t	O	Time setting in seconds for letting the latch retracted after the supporting node's mode has been changed to unsecured. Zero means the functionality is disabled.

11.15.6 zwif_dlck_cfg_get

Get configuration parameter through report callback function.

Table 425 – zwif_dlck_cfg_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
cb	zwrep_dlck_cfg_fn	I	Report callback function.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 426 – zwrep_dlck_cfg_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
config	zwdlck_cfg_t *	I	Configuration.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

11.15.7 zwif_dlck_cap_get

Get supported door lock capabilities through report callback function.

Table 427 – zwif_dlck_cap_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
cb	zwrep_dlck_cap_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 428 – zwrep_dlck_cap_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
cap	zwdlck_cap_t *	I	Door lock capabilities.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 429 – zwdlck_cap_t Structure

Attribute	Type	I/O	Description
op_type_cnt	uint8_t	O	Number of supported door lock operation types in "op_type" buffer.
mode_cnt	uint8_t	O	Number of supported door lock modes in "mode" buffer.
out_hdl	uint8_t	O	Supported outside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if the corresponding handle can be enabled and disabled; else the corresponding handle cannot be enabled or disabled.
in_hdl	uint8_t	O	Supported inside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if the corresponding handle can be enabled and disabled; else the corresponding handle cannot be enabled or disabled.

door_comp_msk	uint8_t	O	supported door lock component bitmask (ZW_COND_XXX) that can be reported in the Door Lock condition field of the Door Lock Operation Report Command.
cap	uint8_t	O	Door lock capabilities bitmask (ZW_DLCK_CAP_XXX). Bit is set to 1 means supported; else unsupported.
op_type	uint8_t *	O	Door lock operation type (ZW_DOOR_OP_XXX) buffer.
Mode	uint8_t *	O	Door lock mode (ZW_DOOR_XXX except ZW_DOOR_UNKNOWN) buffer.

Table 430 – ZW_DLCK_CAP_XXX_MASK

Door lock capabilities bit-mask	Description
ZW_DLCK_CAP_BLK_TO_BLK_MASK	Bit-mask for block-to-block functionality
ZW_DLCK_CAP_TWIST_ASST_MASK	Bit-mask for twist assist functionality
ZW_DLCK_CAP_HOLD_REL_MASK	Bit-mask for hold and release functionality
ZW_DLCK_CAP_AUTO_RLCK_MASK	Bit-mask for auto-relock functionality

11.15.8 zwif_dlck_cap_cache_get

Get supported door lock capabilities from cache. Caller must call zwif_dlck_cap_free() to free the door lock capabilities if this call is successful.

Table 431 – zwif_dlck_cap_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Door lock interface handle.
cap	zwdlck_cap_t **	O	Door lock capabilities.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.15.9 zwif_dlck_cap_free

Free door lock capabilities buffer.

Table 432 – zwif_dlck_cap_free Parameters

Attribute	Type	I/O	Description
cap	zwdlck_cap_t *	I	Door lock capabilities

11.16 Door Lock Logging Interface API

This corresponds to the Z-Wave Door Lock Logging CC. It is used to provide an audit trail in an access control application.

11.16.1 zwif_lcklog_rpt_set

Set up a door lock logging record report callback function.

Table 433 – zwif_lcklog_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rpt_cb	zwrep_lcklog_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 434 – zwrep_lcklog_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
log_rec	zwdlck_log_t *	I	Door lock logging record.

Table 435 – zwdlck_log_t Structure

Attribute	Type	I/O	Description
rec_number	uint8_t	O	Record number
rec_sts	uint8_t	O	Record status. 0=record is empty; 1=record is valid
evt	uint8_t	O	Event, ZWLCK_EVT_XXX
usr_id	uint8_t	O	User id. A valid User Identifier MUST be a value starting from 1 to the maximum number of users supported by the device. 0 if the record does not need to identify a user or User Code is provided in the "usr_code" field
usr_code_len	uint8_t	O	User code string length (excluding the NUL terminated character)
usr_code	uint8_t [12]	O	NUL terminated user code string encoded in ASCII digit (if available)
year*	uint16_t	O	Timestamp: Year (e.g. 2019)
month*	uint8_t	O	Timestamp: Month (1 to 12)
day*	uint8_t	O	Timestamp: Day (1 to 31)
hour*	uint8_t	O	Timestamp: Hour (0 to 23) in local time
minute*	uint8_t	O	Timestamp: Minute (0 to 59) in local time
second*	uint8_t	O	Timestamp: Second (0 to 59) in local time

NOTE: If no RTC (Real Time Clock), all values in Time Stamp record marked with (*) SHOULD be set to 0. Most recent record MUST be stored under Record Number 1.

Table 436 – ZWLCK_EVT_XXX

Door lock logging event	Description
ZWLCK_EVT_LOCK_BY_KPAD_CODE	Lock Command: Keypad access code verified lock command
ZWLCK_EVT_UNLOCK_BY_KPAD_CODE	Unlock Command: Keypad access code verified unlock command
ZWLCK_EVT_LOCK_BY_BUTTON	Lock Command: Keypad lock button pressed
ZWLCK_EVT_UNLOCK_BY_BUTTON	Unlock command: Keypad unlock button pressed
ZWLCK_EVT_LOCK_BY_KPAD_COD_SCHD	Lock Command: Keypad access code out of schedule
ZWLCK_EVT_UNLOCK_BY_KPAD_COD_SCHD	Unlock Command: Keypad access code out of schedule
ZWLCK_EVT_KPAD_WRONG_CODE	Keypad illegal access code entered

ZWLCK_EVT_LOCK_BY_KEY_LATCH	Key or latch operation locked (manual)
ZWLCK_EVT_UNLOCK_BY_KEY_LATCH	Key or latch operation unlocked (manual)
ZWLCK_EVT_AUTO_LOCK	Auto lock operation
ZWLCK_EVT_AUTO_UNLOCK	Auto unlock operation
ZWLCK_EVT_LOCK_BY_ZWAVE_CODE	Lock Command: Z-Wave access code verified
ZWLCK_EVT_UNLOCK_BY_ZWAVE_CODE	Unlock Command: Z-Wave access code verified
ZWLCK_EVT_LOCK_BY_ZWAVE	Lock Command: Z-Wave (no code)
ZWLCK_EVT_UNLOCK_BY_ZWAVE	Unlock Command: Z-Wave (no code)
ZWLCK_EVT_LOCK_BY_ZWAVE_COD_SCHD	Lock Command: Z-Wave access code out of schedule
ZWLCK_EVT_UNLOCK_BY_ZWAVE_COD_SCHD	Unlock Command Z-Wave access code out of schedule
ZWLCK_EVT_ZWAVE_WRONG_CODE	Z-Wave illegal access code entered
ZWLCK_EVT_LOCK_SECURED	Lock secured
ZWLCK_EVT_LOCK_UNSECURED	Lock unsecured
ZWLCK_EVT_USR_CODE_ADDED	User code added
ZWLCK_EVT_USR_CODE_DELETED	User code deleted
ZWLCK_EVT_ALL_USR_CODE_DELETED	All user codes deleted
ZWLCK_EVT_MASTER_CODE_CHANGED	Master code changed
ZWLCK_EVT_USR_CODE_CHANGED	User code changed
ZWLCK_EVT_LOCK_RESET	Lock reset
ZWLCK_EVT_CONFIG_CHANGED	Configuration changed
ZWLCK_EVT_LOW_BATTERY	Low battery
ZWLCK_EVT_BATTERY_INSTALLED	New Battery installed

11.16.2 zwif_lcklog_get

Get door lock logging record through report callback.

Table 437 – zwif_lcklog_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
rec_num	uint8_t	O	Record number. Value 0 is used to get the most recent record entry
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.16.3 zwif_lcklog_sup_get

Get maximum number of supported door lock logging records through report callback.

Table 438 – zwif_lcklog_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cb	zwrep_lcklog_cap_fn	I	Callback function.

cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.

Table 439 – zwrep_lcklog_cap_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
max_rec_cnt	uint8_t	I	Maximum number of door lock logging records.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.16.4 zwif_lcklog_sup_cache_get

Get maximum number of supported door lock logging records from cache.

Table 440 – zwif_lcklog_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
max_rec_cnt	uint8_t *	O	Maximum number of door lock logging records
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17 User Code Interface API

This corresponds to the Z-Wave User Code CC, normally found in door locks.

11.17.1 zwif_usrcod_rpt_set

Set up a user code report callback function.

Table 441 – zwif_usrcod_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
rpt_cb	zwrep_usr_cod_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 442 – zwrep_usr_cod_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
usr_cod	zwusrcod_t *	I	User code and its status.

Table 443 – zwusrcod_t structure

Attribute	Type	I/O	Description

id	uint8_t	O	User identifier. Value of zero is used to set the user code and user status for all supported user identifiers (up to 255) and should be used only with user id status set to ZW_USRCOD_AVAIL.
id_sts	uint8_t	O	User id status, ZW_USRCOD_XXX
code_len	uint8_t	O	User code length. This field is ignored when used to set user code with user ID status equals to ZW_USRCOD_AVAIL (i.e. to erase user code).
code	uint8_t[10]	O	User code; minimum length = 4, maximum length = 10 ASCII digits.

Table 444 – ZW_USRCOD_XXX

User ID status	Description
ZW_USRCOD_AVAIL	Available (not set); i.e., when used to set user code, it means to erase user code.
ZW_USRCOD_UNAVAIL	Occupied /Enabled
ZW_USRCOD_RSVD	Reserved by administrator (the user code is in use but disabled).
ZW_USRCOD_MESSAGING	For messaging purposes only. It is used to relay notifications to a controlling application
ZW_USRCOD_PASSAGE_MODE	Passage mode. It is used to let the door lock permanently grant access
ZW_USRCOD_NO_STS	Status unavailable (The requested user id is invalid). Used in user code report only.

11.17.2 zwif_usrcod_get

Get the specified user code and its status through report callback function.

Table 445 – zwif_usrcod_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
usr_id	uint8_t	I	User identifier.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.3 zwif_usrcod_set

Set the user code of a door lock.

Table 446 – zwif_usrcod_set Parameters

Attribute	Type	I/O	Description
Ifd	zwifd_t *	I	User code interface handle
usr_cod	zwusrcod_t *	I	User code and its status
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.4 zwif_usrcod_sup_get

Get the number of supported users through report callback function.

Table 447 – zwif_usrcod_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
cb	zwrep_usr_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 448 – zwrep_usr_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
max_usr_cnt	uint16_t	I	Number of supported user codes (up to 255 for version 1 and up to 65535 for version 2)
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.17.5 zwif_usrcod_sup_cache_get

Get the number of supported user codes from cache.

Table 449 – zwif_usrcod_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
usr_num	uint16_t *	O	Number of supported user codes.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.6 zwif_usrcod_ext_rpt_set

Set up an extended user code report callback function.

Table 450 – zwif_usrcod_ext_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
rpt_cb	zwrep_usr_cod_ext_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 451 – zwrep_usr_cod_ext_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
usr_cod	zwusrcod_ext_t *	I	Buffer that stores extended user codes.
usr_cod_cnt	uint8_t	I	Number of extended user codes in "usr_cod"
next_usrid	uint16_t	I	Next user identifier having a user id status different than ZW_USRCOD_AVAIL. Zero if no more user code to get.

Table 452 – zwusrcod_ext_t structure

Attribute	Type	I/O	Description
id	uint16_t	O	User identifier Value of zero is used to set the user code and user status for all supported user identifiers and should be used only with user id status set to ZW_USRCOD_AVAIL.
id_sts	uint8_t	O	User id status, ZW_USRCOD_XXX
code_len	uint8_t	O	User code length. This field must be zero when used to set user code with user id status equals to ZW_USRCOD_AVAIL (i.e. to erase user code)
code	uint8_t[10]	O	User code encoded in ASCII representation; minimum length = 4, maximum length = 10

11.17.7 zwif_usrcod_ext_get

Get the specified extended user code and its status through report callback function.

Table 453 – zwif_usrcod_ext_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
usr_id	uint16_t	I	User identifier.
rpt_more	int	I	Flag used to instruct the node to report as many User Codes (with User id status other than ZW_USRCOD_AVAIL) as possible in one report if the node supports "Multiple User Code Report" capability. Non-zero = get more User Codes; 0 = get only the requested user code.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.8 zwif_usrcod_ext_set

Set one or more extended user codes if the device supports "Multiple User Code Set" capability.

Table 454 – zwif_usrcod_ext_set Parameters

Attribute	Type	I/O	Description
Ifd	zwifd_t *	I	User code interface handle
usr_cod	zwusrcod_ext_t *	I	Buffer that stores extended user codes
usr_cod_cnt	uint8_t	I	Number of extended user codes in "usr_cod". If set to 1, it is equivalent to calling zwif_usrcod_set for user code CC version 1 node
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX. ZW_ERR_TOO_LARGE if extended user codes (usr_cod) is too large to fit into the transport layer payload

11.17.9 zwif_usrcod_cap_get

Get supported user code capabilities through report callback.

Table 455 – zwif_usrcod_cap_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
cb	zwrep_usr_cap_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 456 – zwrep_usr_cap_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
cap	zwusrcod_cap_t *	I	User code capabilities
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 457 – zwusrcod_cap_t structure

Attribute	Type	I/O	Description
cap	uint16_t	O	User code capabilities bitmask, ZW_USRCOD_CAP_XXX
id_sts_cnt	uint8_t	O	Number of user id status stored in "id_sts"
kp_mod_cnt	uint8_t	O	Number of keypad modes stored in "kp_mod"
ascii_key_cnt	uint8_t	O	Number of ASCII keys stored in "ascii_key"
id_sts	uint8_t[10]	O	Buffer to store supported user id status, ZW_USRCOD_XXX
kp_mod	uint8_t[10]	O	Buffer to store supported keypad modes, ZW_KEYPAD_MOD_XXX
ascii_key	uint8_t[128]	O	Buffer to store supported ASCII keys

Table 458 – ZW_USRCOD_CAP_XXX Bitmask

User code capabilities bitmask	Description
ZW_USRCOD_CAP_MS_COD	Bitmask to indicate support of master code functionality
ZW_USRCOD_CAP_MS_COD_DEACT	Bitmask to indicate support of master code de-activation
ZW_USRCOD_CAP_CHECKSUM	Bitmask to indicate support of user code checksum functionality
ZW_USRCOD_CAP_MUL_REPORT	Bitmask to indicate support of reporting multiple user codes at once in a single Extended User Code Report Command
ZW_USRCOD_CAP_MUL_SET	Bitmask to indicate support of setting multiple user codes at once in a single Extended User Code Set Command

11.17.10 zwif_usrcod_cap_cache_get

Get supported user code capabilities from cache. Caller must free the allocated memory to the supported user code capabilities if this call is successful.

Table 459 – zwif_usrcod_cap_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
cap	zwusrcod_cap_t **	O	User code capabilities
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.11 zwif_usrcod_kp_mod_rpt_set

Set up a user code keypad mode report callback function.

Table 460 – zwif_usrcod_kp_mod_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
rpt_cb	zwrep_usr_kpmod_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 461 – zwrep_usr_kpmod_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
kp_mod	uint8_t	I	keypad modes, ZW_KEYPAD_MOD_XXX.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 462 – ZW_KEYPAD_MOD_XXX

Keypad mode	Description
ZW_KEYPAD_MOD_NORMAL	Normal mode. This mode MUST be the default mode. Accept the Master Code, if supported
ZW_KEYPAD_MOD_VACATION	Vacation mode. It is used to disable or switch off the User Code functionalities for normal users. However, it accepts the Master Code, if supported
ZW_KEYPAD_MOD_PRIVACY	Privacy mode. It is used to completely disable or switch off the User Code functionalities. Any keypad input is ignored, including Master Code
ZW_KEYPAD_MOD_LCK_OUT	Locked out mode. It is used as a security measure to prevent brute force attacks. This mode is normally activated by the device itself. If the device activates the Locked Out mode, it MUST issue a User Code Keypad Mode Report Command to the Lifeline destination and MUST return to the last active mode automatically after a defined timeout. Any keypad input is ignored, including Master Code.

11.17.12 zwif_usrcod_kp_mod_get

Get user code keypad modes through report callback function.

Table 463 – zwif_usrcod_kp_mod_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.13 zwif_usrcod_kp_mod_set

Set the user code keypad mode.

Table 464 – zwif_usrcod_kp_mod_set Parameters

Attribute	Type	I/O	Description
Ifd	zwifd_t *	I	User code interface handle
kp_mod	uint8_t	I	keypad mode, ZW_KEYPAD_MOD_XXX
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.14 zwif_usrcod_ms_cod_rpt_set

Set up a master code report callback function.

Table 465 – zwif_usrcod_ms_cod_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
rpt_cb	zwrep_ms_cod_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 466 – zwrep_ms_cod_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
ms_cod	uint8_t *	I	Buffer that stores the master code. This can be NULL if code_len is zero.
code_len	uint8_t	I	Master code length. Length of zero is used to indicate the Master Code is deactivated or if the device advertises no support for the Master Code functionality in User Code Capabilities Report Command. Length in the range from 4 to 10 is used to advertise the master code as stored in "ms_cod"

11.17.15 zwif_usrcod_ms_cod_get

Get master code through report callback function.

Table 467 – zwif_usrcod_ms_cod_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.16 zwif_usrcod_ms_cod_set

Set master code.

Table 468 – zwif_usrcod_ms_cod_set Parameters

Attribute	Type	I/O	Description
Ifd	zwifd_t *	I	User code interface handle
ms_cod	uint8_t *	I	Buffer that stores the master code.
code_len	uint8_t	I	Master code length. Length of zero is used to deactivate the master code (if the node supports master code de-activation). Length in the range from 4 to 10 is used to set the master code as stored in "ms_cod"
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.17.17 zwif_usrcod_chksum_rpt_set

Set up a user codes checksum report callback function.

Table 469 – zwif_usrcod_chksum_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
rpt_cb	zwrep_chksum_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 470 – zwrep_chksum_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
chksum	uint16_t	I	Checksum representing all the currently set user codes.

11.17.18 zwif_usrcod_chksum_get

Get checksum representing all the currently set user codes through report callback function.

Table 471 – zwif_usrcod_chksum_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	User code interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.18 Thermostat Fan Mode Interface API

This corresponds to the Z-Wave Thermostat Fan Mode CC.

11.18.1 zwif_thrmo_fan_md_rpt_set

Set up a thermostat fan mode report callback function.

Table 472 – zwif_thrmo_fan_md_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
rpt_cb	zwrep_thrmo_fan_md_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 473 – zwrep_thrmo_fan_md_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
off	uint8_t	I	Fan off mode flag. Non-zero indicates that the fan is fully OFF, “0” indicates that it is possible to change between Fan Modes.
mode	uint8_t	I	Fan operating mode (ZW_THRMO_FAN_MD_XXX)
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 474 – ZW_THRMO_FAN_MD_XXX

Thermostat Fan Mode	Description
ZW_THRMO_FAN_MD_AUTO_LO	Turn the manual fan operation off unless turned on by the furnace or AC. Lower speed is selected in case it is a two-speed fan.
ZW_THRMO_FAN_MD_LOW	Turn the manual fan to low speed.
ZW_THRMO_FAN_MD_AUTO_HI	Turn the manual fan operation off unless turned on by the furnace or AC. High speed is selected in case it is a two-speed fan.
ZW_THRMO_FAN_MD_HI	Turn the manual fan to high speed.
ZW_THRMO_FAN_MD_AUTO_MID	Turn the manual fan operation off unless turned on by the furnace or AC. Medium speed is selected in case it is a three-speed fan.
ZW_THRMO_FAN_MD_MID	Turn the manual fan to medium speed.
ZW_THRMO_FAN_MD_CIR	Turn the manual fan operation off unless turned on by the circulation algorithms.
ZW_THRMO_FAN_MD_HUM_CIR	Turn the manual fan operation off unless turned on by the humidity circulation algorithms.
ZW_THRMO_FAN_MD_LEFT_RIGHT	Turn the manual fan operation off unless turned on by the manufacturer specific "left & right" circulation algorithms.

ZW_THRMO_FAN_MD_UP_DOWN	Turn the manual fan operation off unless turned on by the manufacturer specific "up & down" circulation algorithms.
ZW_THRMO_FAN_MD_QUIET	Turn the manual fan operation off unless turned on by the manufacturer specific "quiet" algorithms.
ZW_THRMO_FAN_MD_EXT_CIR	Turn the manual fan operation off unless turned on by the manufacturer specific "circulation" algorithms. This mode will circulate fresh air from the outside

11.18.2 zwif_thrmo_fan_md_get

Get the thermostat fan operating mode through report callback.

Table 475 – zwif_thrmo_fan_md_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.18.3 zwif_thrmo_fan_md_set

Set the fan mode in the device.

Table 476 – zwif_thrmo_fan_md_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
off	uint8_t	I	Fan off mode flag. Non-zero will switch the fan fully OFF. In order to activate other fan mode this flag must be set to “0”.
mode	uint8_t	I	Fan operating mode (ZW_THRMO_FAN_MD_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.18.4 zwif_thrmo_fan_md_sup_get

Get the supported thermostat fan operating modes through report callback.

Table 477 – zwif_thrmo_fan_md_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
cb	zwrep_thrmo_fan_md_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

Table 478 – zwrep_thrmo_fan_md_sup_fn Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Thermostat fan mode interface handle
off	uint8_t	I	Flag to indicate whether off mode is supported. 1=supported; else 0=unsupported.
mode_len	uint8_t	I	Size of mode buffer.
mode	uint8_t *	I	Buffer to store supported thermostat fan operating modes (ZW_THRMO_FAN_MD_XXX).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.18.5 zwif_thrmo_fan_md_sup_cache_get

Get supported thermostat fan operating modes from cache.

Table 479 – zwif_thrmo_fan_md_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan mode interface handle.
off	uint8_t *	O	Flag to indicate whether off mode is supported.
mode_cnt	uint8_t *	O	Supported thermostat fan operating modes count.
mode	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported thermostat fan operating modes (ZW_THRMO_FAN_MD_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.19 Thermostat Fan State Interface API

This corresponds to the Z-Wave Thermostat Fan State CC.

11.19.1 zwif_thrmo_fan_sta_rpt_set

Set up a thermostat fan operating state report callback function.

Table 480 – zwif_thrmo_fan_sta_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan operating state interface handle.
rpt_cb	zwrep_thrmo_fan_sta_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 481 – zwrep_thrmo_fan_sta_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan operating state interface handle.
state	uint8_t	I	Fan operating state (ZW_THRMO_FAN_STA_XXX).
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 482 – ZW_THRMO_FAN_STA_XXX

Thermostat Fan State	Description
ZW_THRMO_FAN_STA_IDLE	Idle.
ZW_THRMO_FAN_STA_LOW	Running / Running Low - Lower speed is selected in case it is a two-speed fan.
ZW_THRMO_FAN_STA_HI	Running High - High speed is selected in case it is a two-speed fan.
ZW_THRMO_FAN_STA_MEDIUM	Running Medium.
ZW_THRMO_FAN_STA_CIR	Circulation Mode.
ZW_THRMO_FAN_STA_CIR_HUMID	Humidity Circulation Mode.
ZW_THRMO_FAN_STA_CIR_RL	Right-Left Circulation Mode.
ZW_THRMO_FAN_STA_CIR_UP_DN	Up-down Circulation Mode.
ZW_THRMO_FAN_STA_CIR_QUIET	Quiet Circulation Mode.

11.19.2 zwif_thrmo_fan_sta_get

Get the thermostat fan operating state through report callback.

Table 483 – zwif_thrmo_fan_sta_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat fan operating state interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.20 Thermostat Mode Interface API

This corresponds to the Z-Wave Thermostat Mode CC.

11.20.1 zwif_thrmo_md_rpt_set

Setup a thermostat mode report callback function.

Table 484 – zwif_thrmo_md_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
rpt_cb	zwrep_thrmo_md_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 485 – zwrep_thrmo_md_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
mode	uint8_t	I	Mode (ZW_THRMO_MD_XXX).
manf_dat	uint8_t *	I	Manufacturer data when the mode is ZW_THRMO_MD_MANF_SPECIFIC.

manf_dat_cnt	uint8_t	I	Manufacturer data count in bytes. Valid range is from 0 to 7 bytes.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 486 – ZW_THRMO_MD_XXX

Thermostat Mode	Description
ZW_THRMO_MD_OFF	System is off.
ZW_THRMO_MD_HEAT	Only heating will occur.
ZW_THRMO_MD_COOL	Only cooling will occur.
ZW_THRMO_MD_AUTO	Auto.
ZW_THRMO_MD_AUX_HEAT	Auxiliary/Emergency Heat- the thermostat may be put into Aux heat mode simply to use a more efficient secondary heat source when there are no failures of the compressor or heat pump unit itself.
ZW_THRMO_MD_RESUME	The system will resume from last active mode.
ZW_THRMO_MD_FAN	Only cycle fan to circulate air.
ZW_THRMO_MD_FURNACE	Only furnace.
ZW_THRMO_MD_DRY_AIR	The system will cycle cooling in relation to the room and set point temperatures in order to remove moisture from ambient.
ZW_THRMO_MD_MOIST_AIR	Humidification.
ZW_THRMO_MD_AUTO_CHANGEOVER	Heating or cooling will come on according to the auto changeover setpoint.
ZW_THRMO_MD_ENE_SAVE_HEAT	Energy Save Mode Heating will occur.
ZW_THRMO_MD_ENE_SAVE_COOL	Energy Save Mode Cooling will occur.
ZW_THRMO_MD_AWAY	Special Heating Mode, i.e. preventing water from freezing in forced water systems.
ZW_THRMO_MD_FULL_PWR	SPEED UP / FULL POWER heating or cooling mode will be activated when temperature exceeds FULL POWER set point.
ZW_THRMO_MD_MANF_SPECIFIC	Vendor specific thermostat mode.

11.20.2 zwif_thrmo_md_get

Get the thermostat mode through report callback.

Table 487 – zwif_thrmo_md_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.20.3 zwif_thrmo_md_set

Set the thermostat mode in the device.

Table 488 – zwif_thrmo_md_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
mode	uint8_t	I	Thermostat mode (ZW_THRMO_MD_XXX).
manf_dat	uint8_t *	I	Manufacturer data when the mode is ZW_THRMO_MD_MANF_SPECIFIC.
manf_dat_cnt	uint8_t	I	Manufacturer data count in bytes. Valid range is from 0 to 7 bytes.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.20.4 zwif_thrmo_md_sup_get

Get the supported thermostat modes through report callback.

Table 489 – zwif_thrmo_md_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
cb	zwrep_thrmo_md_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 490 – zwrep_thrmo_md_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
mode_len	uint8_t	I	Size of mode buffer.
mode	uint8_t *	I	Buffer to store supported thermostat modes (ZW_THRMO_MD_XXX except ZW_THRMO_MD_MANF_SPECIFIC).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.20.5 zwif_thrmo_md_sup_cache_get

Get the supported thermostat modes from cache.

Table 491 – zwif_thrmo_md_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle

mode_buf	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported thermostat modes (ZW_THRMO_MD_XXX except ZW_THRMO_MD_MANF_SPECIFIC).
mode_cnt	uint8_t *	O	Number of supported modes.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.21 Thermostat Operating State Interface API

This corresponds to the Z-Wave Thermostat Operating State CC.

11.21.1 zwif_thrmo_op_sta_rpt_set

Setup a thermostat operating state report callback function.

Table 492 – zwif_thrmo_op_sta_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
rpt_cb	zwrep_thrmo_op_sta_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 493 – zwrep_thrmo_op_sta_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
state	uint8_t	I	Thermostat operating state (ZW_THRMO_OP_STA_XXX).
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

Table 494 – ZW_THRMO_OP_STA_XXX

Thermostat Fan State	Description
ZW_THRMO_OP_STA_IDLE	Idle
ZW_THRMO_OP_STA_HEAT	Heating
ZW_THRMO_OP_STA_COOL	Cooling
ZW_THRMO_OP_STA_FAN	Fan only
ZW_THRMO_OP_STA_PD_HEAT	Pending Heat
ZW_THRMO_OP_STA_PD_COOL	Pending Cool
ZW_THRMO_OP_STA_VENT	Vent/Economizer
ZW_THRMO_OP_STA_AUX_HEAT	Aux heating
ZW_THRMO_OP_STA_HEAT_2	Second stage heating
ZW_THRMO_OP_STA_COOL_2	Second stage cooling
ZW_THRMO_OP_STA_AUX_HEAT_2	Second stage aux heating
ZW_THRMO_OP_STA_AUX_HEAT_3	Third stage aux heating

11.21.2 zwif_thrmo_op_sta_get

Get the thermostat operating state through report callback.

Table 495 – zwif_thrmo_op_sta_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.21.3 zwif_thrmo_op_sta_log_sup_get

Get the supported thermostat operating state logging.

Table 496 – zwif_thrmo_op_sta_log_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Interface handle.
cb	zwrep_thrmo_op_sta_log_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 497 – zwrep_thrmo_op_sta_log_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
op_sta_len	uint8_t	I	Size of op_sta buffer.
op_sta	uint8_t *	I	Buffer to store supported thermostat operating state logging (ZW_THRMO_OP_STA_XXX).
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.21.4 zwif_thrmo_op_sta_log_sup_cache_get

Get the supported thermostat operating state logging from cache.

Table 498 – zwif_thrmo_op_sta_log_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat mode interface handle.
op_sta_cnt	uint8_t *	O	Supported thermostat operating state logging count.
op_sta	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported thermostat operating state logging (ZW_THRMO_OP_STA_XXX).
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

11.21.5 zwif_thrmo_op_sta_log_rpt_set

Setup a thermostat operating state logging report callback function.

Table 499 – zwif_thrmo_op_sta_log_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
rpt_cb	zwrep_thrmo_op_sta_log_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 500 – zwrep_thrmo_op_sta_log_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
sta_log	zwthrmo_op_sta_log_t *	I	State logging array.
sta_log_cnt	uint8_t	I	Number of state loggings in sta_log.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored..

Table 501 – zwthrmo_op_sta_log_t Structure

Attribute	Type	I/O	Description
tmstamp	time_t	O	Time stamp.
log_type	uint8_t	O	Operating state log type, ZW_THRMO_OP_STA_XXX.
today_hr	uint8_t	O	The number of hours (00 ~ 24) the thermostat has been in the indicated operating state since 12:00 am of the current day.
today_mn	uint8_t	O	The number of minutes (00 ~ 59) the thermostat has been in the indicated operating state since 12:00 am of the current day.
prev_hr	uint8_t	O	The number of hours (00 ~ 24) the thermostat has been in the indicated operating state since 12:00 am of the previous day.
prev_mn	uint8_t	O	The number of minutes (00 ~ 59) the thermostat has been in the indicated operating state since 12:00 am of the previous day.

11.21.6 zwif_thrmo_op_sta_log_get

Get the thermostat operating state logging through report callback.

Table 502 – zwif_thrmo_op_sta_log_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat operating state interface handle.
sta_log_type	uint8_t *	I	Operating state logging type array, ZW_THRMO_OP_STA_XXX, except ZW_THRMO_OP_STA_IDLE.
sta_log_type_cnt	uint8_t	I	Number of operating state logging type in sta_log_type array.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.22 Thermostat Setback Interface API

This corresponds to the Z-Wave Thermostat Setback CC.

11.22.1 zwif_thrmo_setb_rpt_set

Setup a thermostat setback report callback function.

Table 503 – zwif_thrmo_setb_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setback interface handle.
rpt_cb	zwrep_thrmo_setb_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 504 – zwrep_thrmo_setb_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setback interface handle.
type	uint8_t	I	setback type (ZW_THRMO_SETB_TYP_XXX).
state	uint8_t	I	setback state (ZW_THRMO_SETB_STA_XXX).
tenth_degree	int8_t	I	1/10 of a degree. This parameter is valid only if state equals to ZW_THRMO_SETB_STA_SETB.

Table 505 – ZW_THRMO_SETB_TYP_XXX

Thermostat Setback Type	Description
ZW_THRMO_SETB_TYP_NO_OVERRIDE	No override
ZW_THRMO_SETB_TYP_TEMP_OVR	Temporary override
ZW_THRMO_SETB_TYP_PERM_OVR	Permanent override

Table 506 – ZW_THRMO_SETB_STA_XXX

Thermostat Setback State	Description
ZW_THRMO_SETB_STA_SETB	Setback in 1/10 degrees (Kelvin)
ZW_THRMO_SETB_STA_FROST_PROCT	Frost Protection
ZW_THRMO_SETB_STA_ENER_SAVE	Energy Saving Mode
ZW_THRMO_SETB_STA_UNUSED	Unused (for reporting only)

11.22.2 zwif_thrmo_setb_get

Get the thermostat setback type and state through report callback.

Table 507 – zwif_thrmo_setb_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setback interface handle
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX

11.22.3 zwif_thrmo_setb_set

Set the thermostat setback type and state in the device.

Table 508 – zwif_thrmo_setb_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setbackinterface handle.
type	uint8_t	I	Setback type (ZW_THRMO_SETB_TYP_XXX).
state	uint8_t	I	Setback state (ZW_THRMO_SETB_STA_XXX).
tenth_degree	int8_t	I	1/10 of a degree (Kelvin). This parameter is valid if state equals to ZW_THRMO_SETB_STA_SETB. Valid values: -128 to 120 (inclusive) i.e., setback temperature ranges from -12.8 degree K to 12 degree K.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.23 Thermostat Setpoint Interface API

This corresponds to the Z-Wave Thermostat Setpoint CC.

11.23.1 zwif_thrmo_setp_rpt_set

Setup a thermostat setpoint report callback function.

Table 509 – zwif_thrmo_setp_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
rpt_cb	zwrep_thrmo_setp_fn	I	Report callback function.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 510 – zwrep_thrmo_setp_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
data	zwsetp_t *	I	Setpoint data.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
stat_num	uint16_t	I	State number that is incremented by one whenever a cache change is detected.

Table 511 – zwsetp_t structure

Attribute	Type	I/O	Description
type	uint8_t	O	Thermostat setpoint type (ZW_THRMO_SETP_TYP_XXX).
precision	uint8_t	O	Decimal places of the value. The decimal value 1025 with precision 2 is therefore equal to 10.25.

unit	uint8_t	O	Thermostat setpoint unit (ZW_THRMO_SETP_UNIT_XXX).
size	uint8_t	O	Data size: 1,2,or 4 bytes.
data	uint8_t [4]	O	Setpoint data (a signed number) with the first byte as MSB.

Table 512 – ZW_THRMO_SETP_TYP_XXX

Thermostat Setpoint Type	Description
ZW_THRMO_SETP_TYP_HEATING	Heating
ZW_THRMO_SETP_TYP_COOLING	Cooling
ZW_THRMO_SETP_TYP_FURNACE	Furnace
ZW_THRMO_SETP_TYP_DRY	Dry air
ZW_THRMO_SETP_TYP_MOIST	Moist air
ZW_THRMO_SETP_TYP_AUTO_CHANGEOVER	Auto changeover
ZW_THRMO_SETP_TYP_ENE_SAVE_HEAT	Energy Save Heating
ZW_THRMO_SETP_TYP_ENE_SAVE_COOL	Energy Save Cooling
ZW_THRMO_SETP_TYP_AWAY_HEAT	Away heating
ZW_THRMO_SETP_TYP_AWAY_COOL	Away cooling
ZW_THRMO_SETP_TYP_FULL_POWER	Full power

Table 513 – ZW_THRMO_SETP_UNIT_XXX

Thermostat Setpoint Unit	Description
ZW_THRMO_SETP_UNIT_C	Celsius
ZW_THRMO_SETP_UNIT_F	Fahrenheit

11.23.2 zwif_thrmo_setp_get

Get the thermostat setpoint through report callback.

Table 514 – zwif_thrmo_setp_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
type	uint8_t	I	Thermostat setpoint type (ZW_THRMO_SETP_TYP_XXX)..
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.23.3 zwif_thrmo_setp_set

Set the thermostat setpoint in the device.

Table 515 – zwif_thrmo_setp_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
data	zwsetp_t *	I	Setpoint data.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.23.4 zwif_thrmo_setp_sup_get

Get the supported thermostat setpoint types through report callback. For Danfoss Living Connect workaround, the setpoint supported type 0 (Invalid) will be mapped to type 1 (ZW_THRMO_SETP_TYP_HEATING).

Table 516 – zwif_thrmo_setp_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
cb	zwrep_thrmo_setp_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 517 – zwrep_thrmo_setp_sup_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle
type_len	uint8_t	I	size of type buffer
type	uint8_t *	I	buffer to store supported thermostat setpoint type (ZW_THRMO_SETP_TYP_XXX)
valid	int	I	validity of the report. If 1 the report is valid; else this report contains no data and should be ignored

11.23.5 zwif_thrmo_setp_sup_cache_get

Get supported thermostat setpoint types from cache.

Table 518 – zwif_thrmo_setp_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
type_cnt	uint8_t *	O	Supported setpoint types count.
type	uint8_t *	O	Caller supplied buffer of size 255 bytes to store supported thermostat setpoint types (ZW_THRMO_SETP_TYP_XXX).
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.23.6 zwif_thrmo_setp_sup_range_get

Get the supported thermostat temperature range through report callback.

Table 519 – zwif_thrmo_setp_sup_range_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
type	uint8_t	I	Setpoint type, ZW_THRMO_SETP_TYP_XXX.
cb	zwrep_thrmo_setp_range_fn	I	Report callback function.

cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	Int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 520 – zwrep_thrmo_setp_range_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle
type	uint8_t	I	Setpoint type, ZW_THRMO_SETP_TYP_XXX.
temp_range	zwsetp_temp_range_t *	I	Supported temperature ranges array.
temp_range_cnt	uint8_t	I	Temperature ranges count in the "temp_range" array.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.23.7 zwif_thrmo_setp_sup_range_cache_get

Get supported thermostat temperature range from cache.

Table 521 – zwif_thrmo_setp_sup_range_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Thermostat setpoint interface handle.
setp_type	uint8_t	I	Setpoint type, ZW_THRMO_SETP_TYP_XXX.
temp_range	zwsetp_temp_range_t *	I	Caller supplied temperature ranges array with 2 elements.
temp_range_cnt	uint8_t *	O	Temperature ranges count stored in the "temp_range" array.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.24 Configuration Interface API

This corresponds to the Z-Wave Configuration CC for manufacturer-specific device configuration. The semantics of parameters and values are found in the manufacturer user manuals.

11.24.1 zwif_config_rpt_set

Set up a configuration parameter report callback function.

Table 522 – zwif_config_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
rpt_cb	zwrep_config_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 523 – zwrep_config_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
param	zwconfig_t *	I	Parameter value.

Table 524 – zwconfig_t Structure

Attribute	Type	I/O	Description
param_num	uint8_t	I/O	Parameter number.
size	uint8_t	I/O	Data size (1,2,or 4 bytes).
data	uint8_t[4]	I/O	Data (a signed or unsigned number, depending on parameter number data format) with the first byte (i.e. data[0]) is the most significant byte.
use_default	uint8_t	I	Parameter flag (only valid for configuration set command): 1=use default factory setting. 0=use the value specified in data[].

11.24.2 zwif_config_get

Get the value of a configuration parameter through report callback function.

Table 525 – zwif_config_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
param_num	uint8_t	I	Parameter number of the value to get.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.24.3 zwif_config_set

Set the value of a configuration parameter.

Table 526 – zwif_config_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
param	zwconfig_t *	I	Parameter value to set.

11.24.4 zwif_config_bulk_rpt_set

Set up a multiple configuration parameters report callback function.

Table 527 – zwif_config_bulk_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
rpt_cb	zwrep_cfg_bulk_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 528 – zwrep_cfg_bulk_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle
param	zwcfg_bulk_t *	I	Parameter value

Table 529 – zwcfg_bulk_t Structure

Attribute	Type	I/O	Description
param_start	uint16_t	I/O	Starting parameter number
param_cnt	uint8_t	I/O	Parameter count
size	uint8_t	I/O	Data size (1,2,or 4 bytes)
handshake	uint8_t	I/O	For **bulk_set** command: request the device to send Configuration Bulk Report once the parameters have been written. 1=request report; 0=do not request report. For **bulk report** command: 1=indicate that all configuration parameters have been stored in non-volatile memory; 0=indicate that this report is returned in response to a Configuration Bulk Get command.
deflt_data	uint8_t	I/O	For **bulk_set** command: 1=use default factory setting and ignore data[]; 0=use the value in data[]. For **bulk report** command: 1=all advertised configuration parameters in data[] have the factory default value; 0=one or more of the advertised configuration parameters in data[] do not have the factory default value.
data	union [255]	I/O	Configuration data array with number of parameters specified by param_cnt. It is the union of the following:
u8_data	uint8_t		Unsigned data for size=1.
s8_data	int8_t		Signed data for size=1.
u16_data	uint16_t		Unsigned data for size=2.
s16_data	int16_t		Signed data for size=2.
u32_data	uint32_t		Unsigned data for size=4.
s32_data	int32_t		Signed data for size=4.

11.24.5 zwif_config_bulk_get

Get multiple configuration parameters report through report callback.

Table 530 – zwif_config_bulk_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
param_start	uint16_t	I	Starting parameter number.
param_cnt	uint8_t	I	Number of parameters to retrieve.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.24.6 zwif_config_bulk_set

Set multiple configuration parameters.

Table 531 – zwif_config_bulk_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
param	zwcfg_bulk_t *	I	Parameter to set.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.24.7 zwif_config_prm_reset

Reset all configuration parameters to their default values.

Table 532 – zwif_config_prm_reset Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.24.8 zwif_config_info_get

Get configuration parameters information.

Table 533 – zwif_config_info_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Configuration interface handle
cfg_info	zwcfg_info_cap_t **	O	Configuration parameters information if success; NULL on failure.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX **NOTE**: Caller is required to call **zwif_config_info_free** to free the memory allocated to cfg_info.

Table 534 – zwcfg_info_cap_t structure

Attribute	Type	I/O	Description	
cap	uint8_t	O	Device configuration capabilities, CFG_CAP_XXX	
			XXX	Description
			BULK	Support bulk set and bulk report get APIs.
			DEFAULT_SET	Support API to reset all configuration parameters to their default value.
cnt	uint16_t	O	Number of parameters information pointed by 'info'.	
info	zwcfg_info_t *	O	Pointer to an array of parameters info.	

Table 535 – zwcfg_info_t structure

Attribute	Type	I/O	Description
name	char *	O	Name of a parameter. NULL if unavailable.

info	char *	O	Usage information of a parameter. NULL if unavailable.
param_num	uint16_t	O	Parameter number.
adv	uint8_t	O	Flag to indicate if the parameter is to be presented in the “Advanced” parameter section in the controller GUI. 1=advanced; 0=normal.
read_only	uint8_t	O	Flag to indicate if the parameter is read-only. 1=read only; 0=read-write.
alt_cap	uint8_t	O	Flag to indicate if the advertised parameter triggers a change in the node’s capabilities. 1=yes; 0=no. If yes, the behavior of the device depends on whether it is a Z-Wave Plus version 2 node. If it is, it MUST advertise its new capabilities immediately; else it MUST NOT advertise the new and/or Multi Channel End Point capabilities before being excluded from its current network.
format	uint8_t	O	Format of data, CFG_FMT_XXX

XXX	Description
SIGN_INT	Signed integer.
UNSIGN_INT	Unsigned integer.
ENUM	Enumerated (Radio buttons). It must be treated as unsigned integer.
BIT_FIELD	Bit field (Checkboxes). It must be treated as bit field where each individual bit can be set or reset.

size	uint8_t	O	Data size: 1, 2 or 4 bytes. If zero, the "param_num" is invalid and all the data fields must be ignored.
min_val	gen_dat_u	O	Minimum value of data. Zero if format is bit field.
max_val	gen_dat_u	O	Maximum value of data. If the format is “bit field”, each individual supported bit MUST be set to ‘1’, while each un-supported bit MUST be set to ‘0’.
dflt_val	gen_dat_u	O	Default value of data.

Table 536 – gen_dat_u Union

Attribute	Type	I/O	Description
u8_data	uint8_t	O	Unsigned data for size=1.
s8_data	int8_t	O	Signed data for size=1.
u16_data	uint16_t	O	Unsigned data for size=2.
s16_data	int16_t	O	Signed data for size=2.
u32_data	uint32_t	O	Unsigned data for size=4.
s32_data	int32_t	O	Signed data for size=4.

11.24.9 zwif_config_info_free

Free configuration parameters information. Caller should not use the cfg_info after this call.

Table 537 – zwif_config_info_free Parameters

Attribute	Type	I/O	Description
cfg_info	zwcfg_info_cap_t *	O	Configuration parameters information to be free

11.25 Clock Interface API

This corresponds to the Z-Wave Clock CC.

11.25.1 zwif_clock_rpt_set

Setup a clock report callback function.

Table 538 – zwif_clock_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Clock interface handle.
rpt_cb	zwrep_clock_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 539 – zwrep_clock_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Clock interface handle
weekday	uint8_t	I	Day of week (ZW_CLOCK_XXX)
hour	uint8_t	I	Hour (in 24 hours format)
minute	uint8_t	I	Minute

Table 540 – ZW_CLOCK_XXX

Day of Week	Description
ZW_CLOCK_UNDEFINED	Undefined
ZW_CLOCK_MONDAY	Monday
ZW_CLOCK_TUESDAY	Tuesday
ZW_CLOCK_WEDNESDAY	Wednesday
ZW_CLOCK_THURSDAY	Thursday
ZW_CLOCK_FRIDAY	Friday
ZW_CLOCK_SATURDAY	Saturday
ZW_CLOCK_SUNDAY	Sunday

11.25.2 zwif_clock_get

Get current time and week of day through report callback.

Table 541 – zwif_clock_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Clock interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.25.3 zwif_clock_set

Set current time and week of day in the device.

Table 542 – zwif_clock_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Clock interface handle.
weekday	uint8_t	I	Day of week (ZW_CLOCK_XXX).
hour	uint8_t	I	Hour (in 24 hours format).
minute	uint8_t	I	Minute.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.26 Climate Control Schedule Interface API

This corresponds to the Z-Wave Climate Control Schedule CC.

11.26.1 zwif_clmt_ctl_schd_rpt_set

Setup a climate control schedule report callback function.

Table 543 – zwif_clmt_ctl_schd_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
rpt_cb	zwrep_clmt_ctl_schd_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 544 – zwrep_clmt_ctl_schd_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle
sched	zwcc_shed_t *	I	Climate control schedule

Table 545 – zwcc_shed_t Structure

Attribute	Type	I/O	Description
weekday	uint8_t	O	Day of week. Valid from ZW_CLOCK_MONDAY to ZW_CLOCK_SUNDAY.
total	uint8_t	O	Total number of valid schedule entries.
swpts	zwcc_shed_swpt_t [9]	O	Schedule entries (switchpoints). The entries must be ordered by time, ascending from 00:00 towards 23:59. No duplicates of time shall be allowed.

Table 546 – zwcc_shed_swpt_t Structure

Attribute	Type	I/O	Description
hour	uint8_t	O	Hour: 0 ~ 23.
minute	uint8_t	O	Minute: 0 ~ 59.
state	uint8_t	O	Schedule state, ZW_THRMO_SETB_STA_XXX.
tenth_deg	int8_t	O	1/10 of a degree (Kelvin). This parameter is valid if state equals to ZW_THRMO_SETB_STA_SETB. Valid values: -128 to 120 (inclusive). i.e. setback temperature ranges from -12.8 degree K to 12 degree K.

11.26.2 zwif_clmt_ctl_schd_get

Get the climate control schedule of a specific weekday through report callback.

Table 547 – zwif_clmt_ctl_schd_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
weekday	uint8_t	O	Day of week. Valid from ZW_CLOCK_MONDAY to ZW_CLOCK_SUNDAY.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.26.3 zwif_clmt_ctl_schd_set

Set the climate control schedule in a device for a specific weekday.

Table 548 – zwif_clmt_ctl_schd_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
sched	zwcc_shed_t *	I	Climate control schedule.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.26.4 zwif_clmt_ctl_schd_chg_rpt_set

Setup a climate control schedule change report callback function.

Table 549 – zwif_clmt_ctl_schd_chg_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
rpt_cb	zwrep_clmt_ctl_schd_chg_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 550 – zwrep_clmt_ctl_schd_chg_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule change interface handle.

chg_counter	uint8_t	I	Climate control schedule change counter. If the Change Counter is different from last time, this indicates a change in a climate control schedule. Value from 1 to 255 indicates the climate control schedule change mechanism is enabled. Value of 0 indicates the climate control schedule change mechanism is temporarily disabled by the override function.

11.26.5 zwif_clmt_ctl_schd_chg_get

Get climate control schedule change counter through report callback.

Table 551 – zwif_clmt_ctl_schd_chg_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.26.6 zwif_clmt_ctl_schd_ovr_rpt_set

Setup a climate control schedule override report callback function.

Table 552 – zwif_clmt_ctl_schd_ovr_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
rpt_cb	zwrep_clmt_ctl_schd_ovr_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 553 – zwrep_clmt_ctl_schd_ovr_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle
schd_ovr	zwcc_shed_ovr_t *	I	Climate control schedule override

Table 554 – zwcc_shed_t Structure

Attribute	Type	I/O	Description
type	uint8_t	O	Schedule override type, ZW_THRMO_SETB_TYP_XXX.
state	uint8_t	O	Schedule override state, ZW_THRMO_SETB_STA_XXX.
tenth_deg	int8_t	O	1/10 of a degree (Kelvin). This parameter is valid if state equals to ZW_THRMO_SETB_STA_SETB. Valid values: -128 to 120 (inclusive) i.e. setback temperature ranges from -12.8 degree K to 12 degree K.

11.26.7 zwif_clmt_ctl_schd_ovr_get

Get the climate control schedule override through report callback.

Table 555 – zwif_clmt_ctl_schd_ovr_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.26.8 zwif_clmt_ctl_schd_ovr_set

Set the climate control schedule override in a device.

Table 556 – zwif_clmt_ctl_schd_ovr_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Climate control schedule interface handle.
schd_ovr	zwcc_shed_ovr_t *	I	Climate control schedule override.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.27 AV Interface API

This corresponds to the Z-Wave Simple AV CC, which serves as an AV remote control replacement.

11.27.1 zwif_av_set

Send the status of AV button. If parameter *down* = 1, a timer will be set up. This timer will expire in 0.4 second and send the “key hold” keep alive command to the node. The timer will reload itself upon expiry until another call to this function with parameter *down* = 0.

Table 557 – zwif_av_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Audio video interface handle.
ctl	uint16_t	I	Button number ZW_AV_BTN_XX.
down	uint8_t	I	Button status: 0=button up. 1=button down.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.27.2 zwif_av_caps

Get supported AV commands through report callback function.

Table 558 – zwif_av_caps Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Audio video interface handle.
cb	zwrep_av_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 559 – zwrep_av_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Audio video interface handle.
length	uint16_t	I	Mask byte length.
mask	uint8_t *	I	Mask buffer pointer.
valid	int	I	Validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

11.28 Protection Interface API

This corresponds to the Z-Wave Protection CC.

11.28.1 zwif_prot_rpt_set

Setup a protection report callback function.

Table 560 – zwif_prot_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
rpt_cb	zwrep_prot_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 561 – zwrep_prot_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
local_prot	uint8_t	I	local protection state (ZW_LPROT_XXX).
rf_prot	uint8_t	I	RF protection state (ZW_RFPROT_XXX).
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.
snum_local	uint16_t	I	state number that is incremented by one whenever a local protection state change is detected
snum_rf	uint16_t	I	state number that is incremented by one whenever a RF protection state change is detected

Table 562 – ZW_LPROT_XXX

Local protection state	Description
ZW_LPROT_UNPROT	Unprotected - The device is not protected, and can be operated normally via the user interface.
ZW_LPROT_SEQ	Protection by sequence - The device is protected by altering the way the device normally is operated into a more complicated sequence of actions.
ZW_LPROT_NO_CTL	No operation possible - It is not possible at all to control a device directly via the user interface.

Table 563 – ZW_RFPROT_XXX

RF protection state	Description
ZW_RFPROT_UNPROT	Unprotected - The device must accept and respond to all RF Commands.
ZW_RFPROT_NO_CTL	No RF control - all runtime Commands are ignored by the device. The device must still respond with status on requests.
ZW_RFPROT_NO_RESP	No RF response at all. The device will not even reply to status requests.

11.28.2 zwif_prot_get

Get the protection state through report callback.

Table 564 – zwif_prot_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
flag	int	I	flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.28.3 zwif_prot_set

Set the protection state in the device.

Table 565 – zwif_prot_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle
local_prot	uint8_t	I	Local protection state (ZW_LPROT_XXX).
rf_prot	uint8_t	I	RF protection state (ZW_RFPROT_XXX). For device that supports only version 1, this field will be ignored.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.28.4 zwif_prot_sup_get

Get the supported protection states through report callback.

Table 566 – zwif_prot_sup_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
cb	zwrep_prot_sup_fn	I	Report callback function.
cache	int	I	Flag: to get data from cached only. If set, no fetching from real device when cache unavailable.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 567 – zwrep_prot_sup_fn Parameters

Attribute	Type	I/O	Description

ifd	zwifd_t *	I	Protection interface handle.
sup_sta	zwprot_sup_t *	I	supported Protection States.
valid	int	I	validity of the report. If 1 the report is valid; else this report contains no data and should be ignored.

Table 568 – zwprot_sup_t structure

Attribute	Type	I/O	Description
excl_ctl	uint8_t	O	Flag to indicates whether the device supports Exclusive Control.
tmout	uint8_t	O	Flag to indicates whether the device supports timeout for RF Protection State.
lprot	uint8_t [16]	O	Supported Local Protection States.
lprot_len	uint8_t	O	Number of supported Local Protection States.
rfprot	uint8_t [16]	O	Supported RF Protection States.
rfprot_len	uint8_t	O	Number of supported RF Protection States.

11.28.5 zwif_prot_sup_cache_get

Get supported protection states from cache.

Table 569 – zwif_prot_sup_cache_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
sup_prot	zwprot_sup_t *	O	supported protection states.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.28.6 zwif_prot_ec_rpt_set

Setup a protection exclusive control report callback function.

Table 570 – zwif_prot_ec_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
rpt_cb	zwrep_prot_ec_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 571 – zwrep_prot_ec_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
node_id	uint8_t	I	Node ID that has exclusive control can override the RF protection state of the device and can control it regardless of the protection state. Node ID of zero is used to reset the protection exclusive control state.
ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

stat_num	uint16_t	I	state number that is incremented by one whenever a node id change is detected

11.28.7 zwif_prot_ec_get

Get the protection exclusive control node through report callback.

Table 572 – zwif_prot_ec_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.28.8 zwif_prot_ec_set

Set the protection exclusive control node in the device.

Table 573 – zwif_prot_ec_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle
node_id	uint8_t	I	Node ID that has exclusive control can override the RF protection state of the device and can control it regardless of the protection state. Node ID of zero is used to reset the protection exclusive control state.

11.28.9 zwif_prot_tmout_rpt_set

Setup a RF protection timeout report callback function.

Table 574 – zwif_prot_tmout_rpt_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
rpt_cb	zwrep_prot_tmout_fn	I	Report callback function.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

Table 575 – zwrep_prot_tmout_fn Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle
remain_tm	uint8_t	I	Remaining time that a device will remain in protection mode. 0x00 = No timer is set. All “normal operation” Commands must be accepted. 0x01 to 0x3C = 1 second (0x01) to 60 seconds (0x3C); 0x41 to 0xFE = 2 minutes (0x41) to 191 minutes (0xFE); 0xFF = No Timeout - The Device will remain in RF Protection mode infinitely.

ts	time_t	I	Time stamp. If this is zero, the callback has no data and hence other parameter values should be ignored.

11.28.10 zwif_prot_tmout_get

Get the remaining time that a device will remain in RF protection mode through report callback.

Table 576 – zwif_prot_tmout_get Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
flag	int	I	Flag, see ZWIF_GET_BMSK_XXX.
return	int	O	ZW_ERR_NONE on success; else ZW_ERR_XXX.

11.28.11 zwif_prot_tmout_set

Set the RF protection timeout in the device.

Table 577 – zwif_prot_tmout_set Parameters

Attribute	Type	I/O	Description
ifd	zwifd_t *	I	Protection interface handle.
tmout	uint8_t	I	Timeout specifies the time (in different resolutions) a device will remain in RF Protection mode. 0 = No timer is set. All "normal operation" commands MUST be accepted. 0x01 to 0x3C = 1 second (0x01) to 60 seconds (0x3C); 0x41 to 0xFE = 2 minutes (0x41) to 191 minutes (0xFE); 0xFF = No Timeout - The Device will remain in RF Protection mode infinitely.

12 Device Database File Format

The device database is a text file contains the records in JSON format. The following explains different record types using c-style comments (/* and */). Please note that c-style comments are invalid in JSON standard and hence should be removed in a real-use case.

```
{
  "global_settings":
  [
      {
       "wakeup_interval":480   /* Wakeup interval in seconds to set whenever a new sleep
capable device is added to the network. Value of zero or absent of this entry means do not
change the device default value*/
        }
  ],
  "device_records":
  [
    {
      "_comment": "X company sensors",
      "vid":1, /* vendor or manufacturer id */
      "ptype": 4952, /* product type id */
      "pid":3030, /* product id */
      "category":1, /* device category as defined in DEV_XXX */
      "ep": [ /*endpoints */
        {
          "epid":0, /* Endpoint id (starting from 0 for virtual endpoint which is node level,1
and onwards for real endpoints) */
           "interface":[ /*interfaces*/
             {
               "if_type": 1, /*interface type as defined in IF_REC_TYPE_XXX */
               "grp_id": [1, 3, 5] /*group id for the controller to set its node id into the
group(s)*/
             },
             {
               "if_type": 2, /*interface type as defined in IF_REC_TYPE_XXX */
               "config":[ /*Configuration set parameter*/
                {
                  "param_num": 1, /* parameter number */
                  "param_size":2, /* data size: 1,2,or 4 bytes*/
                  "param_val":-10000 /* configuration parameter value (signed integer)*/
                },
                {
                  "param_num": 2, /* parameter number */
                  "param_size":1, /* data size: 1,2,or 4 bytes*/
                  "param_val":15 /* configuration parameter value (signed integer)*/
                }
               ],
               "config_info":[ /*Configuration parameter information*/
                {
                  "param_num":  1, /* parameter number */
                  "param_name": "Sampling interval", /* parameter name */
                  "param_info": "Control the sampling interval in milliseconds (ms)", /*
parameter info */
                  "param_size": 1,  /* data size: 1,2,or 4 bytes*/
                  "param_min":  10, /* minimum configuration parameter value (signed
integer)*/
                  "param_max":  100,/* maximum configuration parameter value (signed
integer)*/
                  "param_deflt":50  /* default configuration parameter value (signed
integer)*/
                },
                {
                  "param_num":  2, /* parameter number */
                  "param_name": "Sensitivity", /* parameter name */
```

```
                    "param_info": "Sensitivity: 0=low, 1=normal, 2=high", /* parameter info */
                    "param_size": 1,  /* data size: 1,2,or 4 bytes*/
                    "param_min":  0, /* minimum configuration parameter value (signed
integer)*/
                    "param_max":  2,/* maximum configuration parameter value (signed integer)*/
                    "param_deflt":1  /* default configuration parameter value (signed
integer)*/
                  }
                ]
              }
            ]
          },
          {
            "epid":1, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards for
real endpoints) */
            "create_interface":113, /* Create command class ID 113 (alarm command class) in
this endpoint. This interface must not already exist in the actual device and endpoint */
            "create_interface_version":4, /* Version for the created command class */
            "redirect":[  /* Command redirection*/
               {
                 "interface": 156, /* CC ID for redirection*/
                 "command": -1, /* CC command 0-255 for redirection. -1 means don't care*/
                 "target_ep": 0 /* Target endpoint to redirect the command to. In this example,
redirect command from ep1 to ep0 */
               }
             ],
             "interface":[ /*interfaces*/
               {
                 "if_type": 3, /*interface type as defined in IF_REC_TYPE_XXX */
                 "bin_sensor_type": 12, /*supported binary sensor type ZW_BSENSOR_TYPE_XXX.
This field is only valid for version 1 (optional) */
                 "no_event_clear": 1, /*indicate the interface is not capable to sent event
clear, i.e. binary sensor value of zero (optional); 1=not capable; 0 or absent of this
field=capable. */
                 "event_clear_timeout": 60, /*timeout in seconds (max: 3600 s), to generate
event_clear callback to user application. This field is only valid when "no_event_clear" = 1.
Default is 5 minutes.(optional)*/
                 "bin_sensor":[ /*Report conversion array*/
                      {/*Each element must contain "match" and "result" block, in the order of
                        "match" followed by "result"*/
                      "match":[/*The block that contains the original binary sensor report
data to compare against. Cannot be empty */
                                /*All the properties need to be met for the conversion to
happen. If a property is missing, it will not be included for comparison*/

                          {

                      "type": 12, /*binary sensor type ZW_BSENSOR_TYPE_XXX (optional)*/

                      "state": 255,  /*binary sensor state 0 or 255.  -1 means don't care. */

                      "result_if_type": 6    /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/

                          }

                       ],
                        "result":[  /*The block that contains the resultant report. Cannot be
empty */
/*If a property is missing, it will follow the same value from the original report*/

                             {

                              "zw_alarm_type": 7, /*Z-Wave alarm type (optional)*/

                               "alarm_type_name": "Burglar", /*Alarm type name (optional)*/
```

```
                                "zw_alarm_event":8, /*Z-Wave alarm event (optional)*/

                                "alarm_level_name":"Motion detected" /*Alarm level name
(optional)*/

                               }

                         ]
                         },
                        {
                        "match":[/*The block that contains the original binary sensor report
data to compare against. Cannot be empty */
                                 /*All the properties need to be met for the conversion to
happen. If a property is missing, it will not be included for comparison*/

                            {

                        "type": 12, /*binary sensor type ZW_BSENSOR_TYPE_XXX (optional)*/

                        "state": 0,  /*binary sensor state 0 or 255.  -1 means don't care. */

                        "result_if_type": 6    /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/

                            }

                         ],
                        "result":[  /*The block that contains the resultant report. Cannot be
empty */
/*If a property is missing, it will follow the same value from the original report*/

                             {

                            "zw_alarm_type": 7, /*Z-Wave alarm type (optional)*/

                             "alarm_type_name": "Burglar", /*Alarm type name (optional)*/

                             "zw_alarm_event":0, /*Z-Wave alarm event (optional)*/

                             "alarm_level_name":"Event inactive", /*Alarm level name (optional)*/

                             "zw_alarm_param_type":5, /*Z-Wave alarm param type (optional)*/

                             "zw_alarm_param":[8]      /*Z-Wave alarm parameters (optional)*/

                              }

                           ]
                           }
                ]
                },
                {
                  "if_type": 4, /*interface type as defined in IF_REC_TYPE_XXX */
                  "sensor_type": 3, /* sensor type ZW_SENSOR_TYPE_XXX */
                  "sensor_unit": 0 /* supported sensor unit ZW_SENSOR_UNIT_XXX */
                }
              ]
            }
          ]
        },
    {
      "_comment": "X company meter",
      "vid":1, /* vendor or manufacturer id */
      "ptype": 4950, /* product type id */
      "pid":3030, /* product id */
      "category":10, /* device category as defined in DEV_XXX */
```

```
      "ep": [ /*endpoints */
        {
          "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards for
real endpoints) */
          "interface":[ /*interfaces*/
            {
              "if_type": 1, /*interface type as defined in IF_REC_TYPE_XXX */
              "grp_id": [1] /*group id for the controller to set its node id into the group*/
            },
            {
              "if_type": 2, /*interface type as defined in IF_REC_TYPE_XXX */
              "config":[ /*Configuration parameter*/
                {
                  "param_num": 1, /* parameter number */
                  "param_size":2, /* data size: 1,2,or 4 bytes*/
                  "param_val":-10000 /* configuration parameter value (signed integer)*/
                }
              ]
            },
            {
              "if_type": 5, /*interface type as defined in IF_REC_TYPE_XXX */
              "meter_type": 1, /*meter type ZW_METER_TYPE_XXX */
              "meter_supported_units":17, /*supported unit bit-mask; see Meter Supported Unit
Bit-mask table */
              "meter_rate_type": 1, /*supported rate type: ZW_METER_RATE_XXX (optional)*/
              "meter_reset_cap": 1 /*meter reset capability: 1=capable to reset; 0=incapable
to reset (optional). If not present, default to incapable to reset*/
            }
          ]
        }
      ]
    },
    {
      "_comment": "All alarms manufactured by Y company. pid= -1 means don't care; i.e. it
matches anything",
      "vid":2, /* vendor or manufacturer id */
      "ptype": 5, /* product type id */
      "pid":-1, /* product id */
      "category":1, /* device category as defined in DEV_XXX */
      "ep": [ /*endpoints */
        {
          "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards for
real endpoints) */
          "interface":[ /*interfaces*/
            {
              "if_type": 6, /*interface type as defined in IF_REC_TYPE_XXX */
              "user_def_version": 7, /*user-defined interface version which will be reported
to user application (optional)*/
              "no_event_clear": 1, /*indicate the interface is not capable to sent event clear
notification (optional); 1=not capable; 0 or absent of this field=capable.  NOTE: This field
is valid only for notification CC version 4 and higher.*/
              "alarm":[
                 {
                   "match":[  /*The block that contains the original alarm report data to
compare against. Cannot be empty */
/*All the properties need to be met for the conversion to happen. If a property is missing, it
will not be included for comparison*/
                     {
                       "alarm_type": 21, /*Vendor specific alarm type (optional)*/
                       "alarm_level": 2  /*Vendor specific alarm level (optional)*/
                     }
                   ],
                   "result":[  /*The block that contains the resultant report. Cannot be
empty*/
             /*If a property is missing, it will follow the same value from the original
report*/
                     {
```

```
                        "zw_alarm_type": 6, /*Z-Wave alarm type mapped from vendor specific
proprietary alarm type (optional)*/
                        "alarm_type_name": "Access Control", /*Alarm type name (optional)*/
                        "zw_alarm_event":5, /*Z-Wave alarm event mapped from vendor specific
alarm level (optional)*/
                        "alarm_level_name":"Keypad Lock", /*Alarm level name (optional)*/
                        "zw_alarm_param_type":2, /*Z-Wave alarm parameter type as defined in
ZW_ALRM_PARAM_XXX (Compulsory when zw_alarm_param is defined in either match/result block)*/
                        "zw_alarm_param":[0,1]  /*Z-Wave alarm event parameters in an array
(optional)*/
                      }
                    ]
                  },
                  {
                    "match":[  /*The block that contains the original alarm report data to
compare against. Cannot be empty */
/*All the properties need to be met for the conversion to happen. If a property is missing, it
will not be included for comparison*/
                      {
                        "alarm_type": 9,  /*Vendor specific alarm type (optional)*/
                         "alarm_level_range":[10,20]  /*Vendor specific alarm level range, means
all values from 10, 20 (both number inclusive) will be used for translation. And they will be
translated into the same report. (optional)*/
                      }
                    ],
                    "result":[  /*The block that contains the resultant report. Cannot be
empty*/
/*If a property is missing, it will follow the same value from the original report*/
                      {
                        "zw_alarm_type": 6, /*Z-Wave alarm type mapped from vendor specific
proprietary alarm type (optional)*/
                        "alarm_type_name": "Access Control", /*Alarm type name (optional)*/
                        "zw_alarm_event":5, /*Z-Wave alarm event mapped from vendor specific
                alarm level (optional)*/
                        "alarm_level_name":"Keypad Lock", /*Alarm level name (optional)*/
                        "zw_alarm_param_type":4, /*Z-Wave alarm parameter type as defined in
ZW_ALRM_PARAM_XXX (Compulsory when zw_alarm_param is defined in either match/result block)*/
                        "zw_alarm_param":[1]  /*Z-Wave alarm event parameters in an array
(optional)*/
                       }
                    ]
                  },
                  {
                    "match":[ /*The block that contains the original alarm report data to
compare against. Cannot be empty */
/*All the properties need to be met for the conversion to happen. If a property is missing, it
will not be included for comparison*/
                      {
                        "alarm_type": 9,  /*Vendor specific alarm level (optional)*/
                        "alarm_level": 2,  /*Vendor specific alarm level (optional)*/
                        "zw_alarm_type": 6,  /*Z-Wave alarm type (optional)*/
                        "zw_alarm_event": 5,  /*Z-Wave alarm event(optional)*/
                        "zw_alarm_param":[0,1,2] /*Z-Wave alarm event parameters(optional)*/
                      }
                    ],
                    "result":[  /*The block that contains the resultant report. Cannot be
empty */
/*If a property is missing, it will follow the same value from the original report*/
                      {
                        "zw_alarm_event": 7  /*Z-Wave alarm event(optional)*/
                      }
                    ]
                  }
                ]
              }
            ]
          }
```

```
        ]
      },
      {
        "_comment": "X company sensors",
        "vid":271, /* vendor or manufacturer id */
        "ptype": 1792, /* product type id */
        "pid":4096, /* product id */
        "category":1, /* device category as defined in DEV_XXX */
        "ep": [ /*endpoints */
          {
            "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards for
real endpoints) */
            "create_interface":113, /* Create CC ID 113 (alarm CC) in this endpoint. This
       interface must not already exist in the actual device and endpoint */
            "create_interface_version":4, /* Version for the created CC */
             "interface":[ /*interfaces*/
                {
                  "if_type": 7, /*interface type as defined in IF_REC_TYPE_XXX */
                  "basic":[
                    {
                      "match":[  /*The block that contains the original basic command data to
compare against. Cannot be empty */
/*All the properties need to be met for the conversion to happen. If a property is missing, it
will not be included for comparison*/
                        {
                          "command": 1,   /*Basic CC command. eg. BASIC_SET*/
                          "value":0,/* Command value 0-255. -1 means don't care. */
                          "result_if_type": 6 /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/
                        }
                      ],
                      "result":[      /*The block that contains the resultant report. Cannot be
empty.*/
/*Content of this block will depends on the value of "result_if_type" in the match block. */
                        {
                          "zw_alarm_type": 6, /*Z-Wave alarm type (optional)*/
                          "alarm_type_name": "Access Control", /*Alarm type name (optional)*/
                          "zw_alarm_event":11,   /*Z-Wave alarm event(optional)*/
                          "alarm_level_name":"Lock is Jammed" /*Alarm level name (optional)*/
                        }
                      ]
                    },
                    {
                      "match":[
                        {
                          "command": 1,   /*Basic CC command. eg. BASIC_SET*/
                          "value":255, /* Command value 0-255. -1 means don't care. */
                          "result_if_type": 6  /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/
                        }
                      ],
                      "result":[
                        {
                          "zw_alarm_type": 6,         /*Z-Wave alarm type (optional)*/
                          "alarm_type_name": "Access Control",   /*Alarm type name (optional)*/
                          "zw_alarm_event":5,         /*Z-Wave alarm event(optional)*/
                          "alarm_level_name":"Keypad Lock" /*Alarm level name (optional)*/
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        },
        {
```

```
          "_comment": "X company alarm sensors",
          "vid":271, /* vendor or manufacturer id */
          "ptype": 1792, /* product type id */
          "pid":4096, /* product id */
          "category":1, /* device category as defined in DEV_XXX */
          "ep": [ /*endpoints */
            {
              "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards for
real endpoints) */
              "create_interface":113, /* Create CC ID 113 (alarm CC) in this endpoint. This
interface must not already exist in the actual device and endpoint */
              "create_interface_version":4, /* Version for the created CC */
               "interface":[ /*interfaces*/
                 {
                   "if_type": 8, /*interface type as defined in IF_REC_TYPE_XXX */
                   "no_event_clear": 1, /*indicate the interface is not capable to sent event
clear, i.e. alarm sensor value of zero (optional); 1=not capable; 0 or absent of this
field=capable. */
                    "event_clear_timeout": 60, /*timeout in seconds (max: 3600 s), to generate
event_clear callback to user application. This field is only valid when "no_event_clear" = 1.
Default is 5 minutes.(optional)*/
                   "hidden":1,  /*hide this interface from user application (optional)*/
                   "alrm_snsr":[
                     {
                       /*Each element must contain "match" and "result" block, in the order of
"match" followed by "result"*/
                      match":[  /*The block that contains the original alarm sensor command data
to compare against. Cannot be empty */
/*All the properties need to be met for the conversion to happen. If a property is missing, it
will not be included for comparison*/
                        {
                          "type": 5,   /*Alarm sensor type. eg. Water leak*/
                          "state":0,   /* One Alarm sensor state value 0-255. -1 means don't
care. */
                          "result_if_type": 6  /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/
                        }
                      ],
                      "result":[       /*The block that contains the resultant report. Cannot be
empty*/
/*Content of this block will depends on the value of "result_if_type" in the match block. */
                        {
                          "zw_alarm_type": 5,         /*Z-Wave alarm type (optional)*/
                          "alarm_type_name": "Water Alarm", /*Alarm type name (optional)*/
                          "zw_alarm_event":0,         /*Z-Wave alarm event(optional)*/
                          "alarm_level_name":"Event inactive", /*Alarm level name (optional)*/
                          "zw_alarm_param_type":5, /*Z-Wave alarm param type (optional)*/
                          "zw_alarm_param":[4] /*Z-Wave alarm parameters (optional)*/
                        }
                      ]
                    },
                    {
                      "match":[
                        {
                          "type": 5,           /*Alarm sensor type. eg. Water leak*/
                          "state_range":[1,255],      /* All Alarm sensor state values within
the range. */
                          "result_if_type": 6  /* Type of the resulted conversion as defined in
IF_REC_TYPE_XXX*/
                        }
                      ],
                      "result":[
                        {
                          "zw_alarm_type": 5,         /*Z-Wave alarm type (optional)*/
                          "alarm_type_name": "Water Alarm",    /*Alarm type name (optional)*/
                          "zw_alarm_event":4,         /*Z-Wave alarm event(optional)*/
```

```
                        "alarm_level_name":"Water level dropped" /*Alarm level name
(optional)*/
                    }
                ]
            }
        ]
    }
    ]
  },
  {
   "_comment": "X company thermostat setpoint",
   "vid":271, /* vendor or manufacturer id */
   "ptype":1882, /* product type id */
   "pid":30, /* product id */
   "category":27, /* device category as defined in DEV_XXX */
   "ep": [ /*endpoints */
          {
            "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards
for real endpoints) */
            "interface":[ /*interfaces*/
                 {
                   "if_type": 9, /*interface type as defined in IF_REC_TYPE_XXX */
                   "thrmo_setp":[
                           {
                           "type": 11, /* thermostat setpoint type, ZW_THRMO_SETP_TYP_XXX
*/
                           "min_unit": 0, /* Temperature unit, ZW_THRMO_SETP_UNIT_XXX*/
                           "min_value": "25.5", /* Minimum temperature value as string, may
contain decimal point */
                           "max_unit": 0, /* Temperature unit, ZW_THRMO_SETP_UNIT_XXX*/
                           "max_value": "30.5"  /* Maximum temperature value as string, may
contain decimal point */
                           },
                           {
                           "type": 11, /* Same thermostat setpoint type as in above entry*/
                           "min_unit": 1, /* In different temperature unit*/
                           "min_value": "77.9", /* Minimum temperature value as string, may
contain decimal point */
                           "max_unit": 1, /* Temperature unit, ZW_THRMO_SETP_UNIT_XXX*/
                           "max_value": "86.9"  /* Maximum temperature value as string, may
contain decimal point */
                           },
                           {
                           "type": 12, /* thermostat setpoint type, ZW_THRMO_SETP_TYP_XXX
*/
                           "min_unit": 0, /* Temperature unit, ZW_THRMO_SETP_UNIT_XXX*/
                           "min_value": "12.5", /* Minimum temperature value as string, may
contain decimal point */
                           "max_unit": 0, /* Temperature unit, ZW_THRMO_SETP_UNIT_XXX*/
                           "max_value": "28"  /* Maximum temperature value as string, may
contain decimal point */

                          }
                         ]
                    }
                ]
             }
            ]
      },
    {
      "_comment": "XYZ company door lock",
      "vid":234, /* vendor or manufacturer id */
      "ptype":1882, /* product type id */
      "pid":3, /* product id */
      "category":7, /* device category as defined in DEV_XXX */
```

```
      "ep": [ /*endpoints */
                {
                "epid":0, /* Endpoint id (starting from 0 for virtual endpoint, 1 and onwards
for real endpoints) */
                "interface":[ /*interfaces*/
                                {
                                "if_type": 10, /*IF_REC_TYPE_DOOR_LOCK, the interface type as
defined in IF_REC_TYPE_XXX */
                                "op_mode": [0, 1, 255], /*Supported door lock operation modes,
as defined in ZW_DOOR_XXX */
                                "op_type": [1, 2], /*Supported door lock operation types, as
defined in ZW_DOOR_OP_XXX */
                                "out_handle": 7, /*Supported outside door handle mode bitmask.
It's a 4-bit mask. Maximum value is 15 */
                                "in_handle": 15 /*Supported inside door handle mode bitmask.
It's a 4-bit mask. Maximum value is 15 */
                                },
                                {
                                "if_type": 11, /*IF_REC_TYPE_MULTI_CMD, the interface type as
defined in IF_REC_TYPE_XXX */
                                "send_encap": 0 /*control whether to send consecutive commands
using multi-command encapsulation. 0 = no encapsulation; 1 = use encapsulation*/
                                }
                            ]
                }
            ]
    }
  ]
}
```

13 Home Network File Format

13.1 Introduction

The main usage of persistent storage is to store the entire Z-Wave network structure which consists of network, nodes, endpoints and interfaces. The typical data stored are the network entities property, capability, and run-time cache derived from reports sent by devices. The file format of the persistent storage is a widely used lightweight data-interchange JSON (JavaScript Object Notation) format. It is easy for humans to read and write; likewise, it is also easy for machines to parse and generate. All data are stored in **big-endian** format as in Z-Wave CC definitions when applicable. The following sub-sections documents the "name-value" pairs used in the persistent storage file. The main purpose of this documentation is to allow porting of Z-Wave network structure data generated in other formats by other "non-Z-Ware library-based" applications to Z-Ware library-based applications. For this reason, developers may skip reading this section if there is no requirement for data porting. Developers who are interested in porting are strongly advised to read the source code in src/zip_pstorage.c and reuse the functions in the source file in order to speed up the porting process.

13.2 Top Level Entities

The top level JSON entities are persistent file version, network and nodes as shown in the table below.

Table 578 – Top level entities

Name	Value Type	JSON Data Type	Description
ver_major	uint8_t	number	Major version of persistent storage file
ver_minor	uint8_t	number	Minor version of persistent storage file
network		object	Network object
node		array	Node object array

13.3 Network

Z-Wave network entities are shown as in the table below.

Table 579 – Network object

Name	Value Type	JSON Data Type	Description
home_id	uint32_t	number	Z-Wave network home id
assc_node_id	uint8_t	number	Lifeline (group 1) association node id. Value of zero denotes there is no node association.
assc_ep_id	uint8_t	number	Lifeline (group 1) association endpoint id if the association type is endpoint association by way of multi-channel association set command. This entry

			MUST be omitted if the association type is node association.

13.4 Nodes

Z-Wave node entities are shown as in the table below.

Table 580 – Node object

Name	Value Type	JSON Data Type	Description
node_id	uint8_t	number	Z-Wave node id
vendor_id	uint16_t	number	Vendor/Manufacturer id
vendor_product_type	uint16_t	number	Vendor/Manufacturer product type
product_id	uint16_t	number	Product id
endpoint_count	uint8_t	number	Number of endpoints
aggregated_ep_count	uint8_t	number	Number of aggregated endpoints
node_property	uint8_t	number	Node property bitmask as defined in NODE_PROPTY_XXX
sleep_capability	uint8_t	number	Flag to indicate the node is capable to sleep with Reporting Sleeping Slave (RSS) role type. 1=sleep capable; 0=always listening.
always_listening	uint8_t	number	Flag to indicate whether the node is always listening and awake (non-sleeping). 1=always listening and awake; 0=otherwise
zwave_lib_type	uint8_t	number	Z-Wave protocol library type
zwave_protocol_ver	uint16_t	number	Z-Wave protocol version
application_ver	uint16_t	number	Application version (also known as firmware 0 version) assigned by the manufacturer
multi_cmd_support	uint8_t	number	Flag to indicate whether the node supports multi-command encapsulation. 1=support; 0=otherwise
listening_sleeping_slave	uint8_t	number	Flag to indicate whether the node is a listening sleeping slave (LSS). 1=yes; 0=no
device_category	uint8_t	number	Device category as defined in “DEV_XXX Meaning and Values” table.
basic_device_class	uint8_t	number	Z-Wave basic device class
wakeup_interval	int32_t	number	Wake up interval in seconds. Value of -1 = invalid or unknown
device_id_type#	uint8_t	number	Device id type
device_id_format#	uint8_t	number	Device id data format as defined in DEV_ID_FMT_XXX

device_id_data#	uint8_t []	number array	Device id data stored in uint8_t number array
hardware_ver@	uint8_t	number	Hardware version
firmware_versions@	uint16_t []	number array	Firmware versions stored in uint16_t number array with the first item is firmware 1 version
s2_granted_keys*	uint8_t	number	Security 2 keys granted to the node.
s2_dsk*	char []	string	Security 2 Device Specific Key (DSK)
zwave_software_versions^		array	Z-Wave software version object array
ep		array	Endpoint object array

NOTE: Entities marked with (#) are only required if the node supports Manufacturer Specific CC version 2; those marked with (@) are only required if the node supports Version CC version 2 and above; those marked with (*) are only required if the node included in Security 2 capable network; those marked with (^) are only required if the node supports Version CC version 3 and above.

Table 581 – Z-Wave software version object

Name	Value Type	JSON Data Type	Description
sw_ver_type	uint8_t	number	Version type as defined in VER_TYPE_XXX
sw_ver_major	uint8_t	number	Major version
sw_ver_minor	uint8_t	number	Minor version
sw_ver_patch	uint8_t	number	Patch version
sw_ver_build	uint16_t	number	Build number. Zero if unused

13.5 Endpoints

Z-Wave endpoint entities are shown as in the table below.

Table 582 – Endpoint object

Name	Value Type	JSON Data Type	Description
ep_id	uint8_t	number	Z-Wave endpoint id. Value of zero denotes virtual endpoint (i.e. root device); value of 1 onwards denotes actual endpoint.
generic_dev_class	uint8_t	number	Z-Wave generic device class
specific_dev_class	uint8_t	number	Z-Wave specific device class
ep_name	char []	string	Optional endpoint's name
ep_location	char []	string	Optional endpoint's location
zwave+_version#	uint8_t	number	Z-Wave+ version
zwave+_role_type#	uint8_t	number	Z-Wave+ role type
zwave+_node_type#	uint8_t	number	Z-Wave+ node type
zwave+_installer_icon_type#	uint16_t	number	Installer icon type
zwave+_user_icon_type#	uint16_t	number	User icon type

aggregated_ep_members@	uint8_t []	number array	Array of aggregated endpoint id
intf		array	Interface object array

NOTE: Entities marked with (#) are only required if the node supports Z-Wave Plus Info CC version 2; those marked with (@) are only required if the node supports Multi Channel CC version 4 and above

13.6 Interfaces

Z-Wave interface entities are shown as in the tables below. The Common Interface object contains the entities common to all interfaces. The rest of the tables are interface specific to the Z-Wave CC.

Table 583 – Common Interface object

Name	Value Type	JSON Data Type	Description
cc	uint16_t	number	Z-Wave CC
cc_ver	uint8_t	number	Z-Wave CC version
cc_propty	uint8_t	number	Properties of the interface (bitmask) as defined in IF_PROPTY_XXX

13.6.1 Association Group Info CC

Table 584 – Interface object for Association Group Info CC

Name	Value Type	JSON Data Type	Description
assoc_grp_info_dynamic	uint8_t	number	Flag to indicate the group info is dynamic. 1=dynamic; 0=static
assoc_grp_info		array	Association Group Info object array

Table 585 –Association Group Info object

Name	Value Type	JSON Data Type	Description
group_id	uint8_t	number	Group id
group_profile	uint16_t	number	Profile
group_name	char []	string	Group name string in UTF-8
cmd_list		array	Command object array

Table 586 – Command object

Name	Value Type	JSON Data Type	Description
class	uint16_t	number	Z-Wave CC
cmd	uint8_t	number	Command of the CC

13.6.2 Configuration CC

Table 587 – Interface object for Configuration CC

Name	Value Type	JSON Data Type	Description
config_param_capabilities	uint8_t	number	Device configuration capabilities (bitmask) as defined in CFG_CAP_XXX
config_param_info		array	Configuration parameter object array

Table 588 – Configuration parameter object

Name	Value Type	JSON Data Type	Description
name	char []	string	Name of a parameter
info	char []	string	Usage information of a parameter
param_number	uint16_t	number	Parameter number
advanced	uint8_t	number	Flag to indicate if the parameter is to be presented in the "Advanced" parameter section in the controller GUI. 1=advanced; 0=normal
read_only	uint8_t	number	Flag to indicate if the parameter is read-only. 1=read only; 0=read-write
trigger_change	uint8_t	number	Flag to indicate if the advertised parameter triggers a change in the node's capabilities. 1=yes; 0=no. If yes, the behavior of the device depends on whether it is a Z-Wave Plus version 2 node. If it is, it MUST advertise its new capabilities immediately; else it MUST NOT advertise the new and/or Multi Channel End Point capabilities before being excluded from its current network
format	uint8_t	number	Format of data as defined in CFG_FMT_XXX
size	uint8_t	number	Data size: 1, 2 or 4 bytes. If zero, the "param_number" is invalid and all the data fields must be ignored
min_value	Type depends on “format” & “size”	number	Minimum value of data. Zero if format is bit field
max_value		number	Maximum value of data. If the format is "bit field", each individual supported bit MUST be set to '1', while each un-supported bit MUST be set to '0'.
default_value		number	Default value of data.

13.6.3 Association and Multi Channel Association CC

Table 589 – Interface object for Association and Multi channel Association CC

Name	Value Type	JSON Data Type	Description
max_groups	uint8_t	number	Maximum supported group count
cache		array	Association Group Cache object array

Table 590 –Association Group Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
grp_id	uint8_t	number	Group id
max_supp	uint8_t	number	Maximum supported members
member		array	Group member object array

Table 591 –Group member object

Name	Value Type	JSON Data Type	Description
node_id	uint8_t	number	Node id
ep_id	uint8_t	number	Endpoint id. Value of 255 denotes node association; other values denote endpoint association

13.6.4 Central Scene CC

Table 592 – Interface object for Central Scene CC

Name	Value Type	JSON Data Type	Description
scene_count	uint8_t	number	Maximum number of supported scenes
max_key_attrib_count_per_scene	uint8_t	number	Maximum number of key attributes per scene
slow_refresh	uint8_t	number	Flag to indicate support slow refresh of Key Held Down notification. Non-zero=support; 0=unsupported
same_key_attrib	uint8_t	number	Flag to indicate if all scenes are supporting the same key attributes. 1=same; 0=different
scene		array	Scene object array

Table 593 –Scene object

Name	Value Type	JSON Data Type	Description
scene_number	uint8_t	number	Scene id (starting from 1)
key_attrib	uint8_t []	number array	Key attributes stored in uint8_t number array

13.6.5 Multi Level Sensor CC

Table 594 – Interface object for Multi Level Sensor CC

Name	Value Type	JSON Data Type	Description
supp_sensor		array	Multi level sensor object array
cache		array	Multi level sensor Cache object array

Table 595 –Multi level sensor object

Name	Value Type	JSON Data Type	Description
type	uint8_t	number	Supported sensor type
unit_bitmask	uint8_t	number	Supported sensor units (bitmask)

Table 596 – Multi level sensor Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sensor_type	uint8_t	number	Sensor type as defined in ZW_SENSOR_TYPE_XXX
sensor_unit	uint8_t	number	Sensor unit as defined in ZW_SENSOR_UNIT_XXX
precision	uint8_t	number	Decimal places of the value.
value	uint8_t []	number array	Sensor value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.

13.6.6 Indicator CC

Table 597 – Interface object for Indicator CC

Name	Value Type	JSON Data Type	Description
supp_indicator		array	Indicator object array
cache		array	Indicator Cache object array

Table 598 –Indicator object

Name	Value Type	JSON Data Type	Description
id	uint8_t	number	Supported indicator id as defined in ZWIND_ID_XXX
property_id	uint8_t []	number array	Property id stored in uint8_t number array.

Table 599 – Indicator Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
indicator_id	uint8_t	number	Indicator id as defined in ZWIND_ID_XXX
property		array	Indicator property object array

Table 600 – Indicator property object

Name	Value Type	JSON Data Type	Description
id	uint8_t	number	Property id
value	uint8_t	number	Property value

13.6.7 Thermostat Fan Mode CC

Table 601 – Interface object for Thermostat Fan Mode CC

Name	Value Type	JSON Data Type	Description
off_mode_supp	uint8_t	number	Flag to indicate whether off mode is supported
supp_mode	uint8_t []	number array	Supported fan modes stored in uint8_t number array.
cache		array	Thermostat Fan Mode Cache object array

Table 602 – Thermostat Fan Mode Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
off	uint8_t	number	Flag to indicate the the fan is fully switched off.
mode	uint8_t	number	Fan mode as defined in ZW_THRMO_FAN_MD_XXX

13.6.8 Thermostat Mode CC

Table 603 – Interface object for Thermostat Mode CC

Name	Value Type	JSON Data Type	Description
supp_mode	uint8_t []	number array	Supported modes stored in uint8_t number array.
cache		array	Thermostat Mode Cache object array

Table 604 – Thermostat Mode Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
mode	uint8_t	number	Mode as defined in ZW_THRMO_MD_XXX
manf_data	uint8_t []	number array	Manufacturer data (when the mode is ZW_THRMO_MD_MANF_SPECIFIC) stored in uint8_t number array.

13.6.9 Thermostat Setpoint CC

Table 605 – Interface object for Thermostat Setpoint CC

Name	Value Type	JSON Data Type	Description
supp_type_count	uint8_t	number	Number of supported setpoint types
supp_type		array	Thermostat Setpoint object array
cache		array	Thermostat Setpoint Cache object array

Table 606 – Thermostat Setpoint object

Name	Value Type	JSON Data Type	Description
type	uint8_t	number	Supported setpoint type as defined in ZW_THRMO_SETP_TYP_XXX
unit_min#	uint8_t	number	Minimum setpoint temperature unit as defined in ZW_THRMO_SETP_UNIT_XXX
precision_min#	uint8_t	number	Decimal places of the minimum setpoint temperature value
value_min#	uint8_t []	number array	Minimum setpoint temperature value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.
unit_max#	uint8_t	number	Maximum setpoint temperature unit as defined in ZW_THRMO_SETP_UNIT_XXX
precision_max#	uint8_t	number	Decimal places of the maximum setpoint temperature value
value_max#	uint8_t []	number array	Maximum setpoint temperature value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.

NOTE: Entities marked with (#) must be omitted if the information is unavailable because the node supports only Thermostat Setpoint CC version less than 3.

Table 607 – Thermostat Setpoint Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
type	uint8_t	number	Thermostat setpoint type as defined in ZW_THRMO_SETP_TYP_XXX
unit	uint8_t	number	Temperature unit as defined in ZW_THRMO_SETP_UNIT_XXX
precision	uint8_t	number	Decimal places of the value.

value	uint8_t []	number array	Temperature value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.

13.6.10 Thermostat Operating State CC

Table 608 – Interface object for Thermostat Operating State CC

Name	Value Type	JSON Data Type	Description
supp_log	uint8_t []	number array	Supported operating state logging stored in uint8_t number array.
cache		array	Thermostat Operating State Cache object array. The array must not contain more than one cache object for “operating state” but can contain multiple cache objects for “logging”.

Table 609 – Thermostat Operating State Cache object for “Operating State”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
op_state	uint8_t	number	Operating state as defined in ZW_THRMO_OP_STA_XXX

Table 610 – Thermostat Operating State Cache object for “Logging”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
log_type	uint8_t	number	logging type as defined in ZW_THRMO_OP_STA_XXX
today_h	uint8_t	number	The number of hours (00 ~ 24) the thermostat has been in the indicated operating state since 12:00 am of the current day
today_m	uint8_t	number	The number of minutes (00 ~ 59) the thermostat has been in the indicated operating state since 12:00 am of the current day
prev_h	uint8_t	number	The number of hours (00 ~ 24) the thermostat has been in the indicated operating state since 12:00 am of the previous day
prev_m	uint8_t	number	The number of minutes (00 ~ 59) the thermostat has been in the indicated operating state since 12:00 am of the previous day

13.6.11 Thermostat Fan State CC

Table 611 – Interface object for Thermostat Fan State CC

Name	Value Type	JSON Data Type	Description
cache		array	Thermostat Fan State Cache object array

Table 612 – Thermostat Fan State Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
fan_stat	uint8_t	number	Fan state as defined in ZW_THRMO_FAN_STA_XXX

13.6.12 Multi Level Switch CC

Table 613 – Interface object for Multi Level Switch CC

Name	Value Type	JSON Data Type	Description
primary_type	uint8_t	number	Primary switch type as defined in SW_TYPE_XXX
secondary_type	uint8_t	number	Secondary switch type as defined in SW_TYPE_XXX
cache		array	Multi Level Switch Cache object array

Table 614 – Multi Level Switch Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
curr_val	uint8_t	number	Current value: 0 = off; 0x01~0x63 = percentage (%);0xFE = Unknown; 0xFF = On.
tgt_val	uint8_t	number	Target value with same interpretation as curr_val
dur	uint8_t	number	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved

13.6.13 Binary Switch CC

Table 615 – Interface object for Binary Switch CC

Name	Value Type	JSON Data Type	Description
cache		array	Binary Switch Cache object array

Table 616 – Binary Switch Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp

sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
curr_val	uint8_t	number	Current value: 0 = off; 0xFE = Unknown; 0xFF = On.
tgt_val	uint8_t	number	Target value with same interpretation as curr_val
dur	uint8_t	number	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved

13.6.14 Binary Sensor CC

Table 617 – Interface object for Binary Sensor CC

Name	Value Type	JSON Data Type	Description
supp_sensor	uint8_t []	number array	Supported binary sensor types stored in uint8_t number array.
cache		array	Binary Sensor Cache object array

Table 618 – Binary Sensor Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
type	uint8_t	number	Binary sensor type as defined in ZW_BSENSOR_TYPE_XXX
state	uint8_t	number	Sensor state: 0=idle; 0xFF=event detected

13.6.15 Simple AV Control CC

Table 619 – Interface object for Simple AV Control CC

Name	Value Type	JSON Data Type	Description
supp_av_ctl_command_bitmask	uint8_t []	number array	Supported AV Control Commands (bitmask) stored in uint8_t number array.

13.6.16 Alarm or Notification CC

Table 620 – Interface object for Alarm CC

Name	Value Type	JSON Data Type	Description
supp_v1_type	uint8_t	number	Flag to indicate whether vendor specific alarm type supported
supp_type		array	Alarm object array
cache		array	Alarm Cache object array

Table 621 –Alarm object

Name	Value Type	JSON Data Type	Description
type	uint8_t	number	Supported Alarm type as defined in ZW_ALRM_XXX
event_bitmask	uint8_t []	number array	Supported events (bitmask) stored in uint8_t number array.

Table 622 – Alarm Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
v1_type	uint8_t	number	Vendor specific alarm type
v1_level	uint8_t	number	Vendor specific alarm level
type#	uint8_t	number	Z-Wave alarm type as defined in ZW_ALRM_XXX
zensor_id#	uint8_t	number	Zensor Net source node id. This field is 0 if device is not based on Zensor Net
status#	uint8_t	number	Z-Wave alarm status as defined in ZW_ALRM_STS_XXX
event#	uint8_t	number	Z-Wave alarm event as defined in ZW_ALRM_EVT_XXX
seq_no#	uint8_t	number	Z-Wave alarm sequence number
evt_param#		array	Alarm event parameter object array

NOTE: Entities marked with (#) must be omitted if the information is unavailable because the node supports only Alarm or Notification CC version less than 3.

Table 623 – Alarm event parameter object

Name	Value Type	JSON Data Type	Description
evt_param_type	uint8_t	number	Z-Wave alarm event parameter type (ZW_ALRM_PARAM_XXX)
evt_param	uint8_t []	number array	Z-Wave alarm event parameter stored in uint8_t number array.

13.6.17 Protection CC

Table 624 – Interface object for Protection CC

Name	Value Type	JSON Data Type	Description
excl_ctl_supp	uint8_t	number	Flag to indicates whether the device supports Exclusive Control
timeout_supp	uint8_t	number	Flag to indicates whether the device supports timeout for RF Protection State
supp_local_state	uint8_t []	number array	Supported Local Protection States stored in uint8_t number array.
supp_rf_state	uint8_t []	number array	Supported RF Protection States stored in uint8_t number array.
cache		array	Protection Cache object array. The array must not contain more than one cache object for

			“protection state”, “exclusive control” and “timeout” respectively.

Table 625 – Protection Cache object for “Protection State”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
local_state	uint8_t	number	Local protection state as defined in ZW_LPROT_XXX
rf_state	uint8_t	number	RF protection state as defined in ZW_RFPROT_XXX
sn_local	uint16_t	number	State number that is incremented by one whenever the cache for "local protection state" has changed
sn_rf	uint16_t	number	State number that is incremented by one whenever the cache for "RF protection state" has changed

Table 626 – Protection Cache object for “Exclusive Control”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
ec_node	uint8_t	number	NodeID that has exclusive control
sn_ec	uint16_t	number	State number that is incremented by one whenever the cache for "exclusive control node id" has changed

Table 627 – Protection Cache object for “Timeout”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
remain_tmout	uint8_t	number	Remaining timeout. 0x00 = No timer is set. All "normal operation" Commands must be accepted. 0x01 to 0x3C = 1 second (0x01) to 60 seconds (0x3C); 0x41 to 0xFE = 2 minutes (0x41) to 191 minutes (0xFE); 0xFF = No Timeout - The Device will remain in RF Protection mode infinitely.

13.6.18 User Code CC

Table 628 – Interface object for User Code CC

Name	Value Type	JSON Data Type	Description
max_user	uint16_t	number	Maximum number of supported users
cap_bitmask	uint16_t	number	User code capabilities bitmask as defined in ZW_USRCOD_CAP_XXX

supp_status	uint8_t []	number array	supported user id status (ZW_USRCOD_XXX) stored in uint8_t number array.
supp_keypad_mode	uint8_t []	number array	supported keypad modes (ZW_KEYPAD_MOD_XXX) stored in uint8_t number array.
supp_ascii_key	uint8_t []	number array	supported ASCII keys stored in uint8_t number array.
cache		array	User Code Cache object array

Table 629 – User Code Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
keypad_mode	uint8_t	number	Keypad mode

13.6.19 Meter CC

Table 630 – Interface object for Meter CC

Name	Value Type	JSON Data Type	Description
supp_type	uint8_t	number	Supported meter type as defined in ZW_METER_TYPE_XXX
reset_supp	uint8_t	number	Flag to indicate capability to reset all accumulated values stored in the meter device. 1=reset supported, 0= unsupported.
supp_rate	uint8_t	number	Supported rate type as defined in ZW_METER_RATE_XXX or 0 if the supported rate type information is unavailable
supp_unit_bitmask	uint16_t	number	Supported unit bitmask as defined in ZW_METER_SUP_UNIT_XXX
cache		array	Meter Cache object array

Table 631 – Meter Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
type	uint8_t	number	Meter type as defined in ZW_METER_TYPE_XXX
unit	uint8_t	number	Meter unit as defined in ZW_METER_UNIT_XXX
rate	uint8_t	number	Meter rate type as defined in ZW_METER_RATE_XXX
precision	uint8_t	number	Decimal places of the value.
value	uint8_t []	number array	Meter reading value stored in uint8_t number array. The number of items in the array is

			determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.
delta_time#	uint16_t	number	Elapsed time in seconds between the 'Meter Value' and the 'Previous Meter Value' measurements
previous_value#	uint8_t []	number array	Previous meter reading value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value.

NOTE: Entities marked with (#) must be omitted if the information is unavailable because the node supports only Meter CC version less than 2 or "delta_time" equals to zero

13.6.20 Meter Table Monitor CC

Table 632 – Interface object for Meter Table Monitor CC

Name	Value Type	JSON Data Type	Description
cap_bitmask	uint8_t	number	Capability bitmask as defined in ZW_METER_CAP_XXX. ZW_METER_CAP_MON means id & admin number available. ZW_METER_CAP_CFG means admin number can be set (i.e. the device supports Meter Table Configuration CC).
id	char []	string	ID, used for identification of customer and metering device
admin_number	char []	string	Admin number used to identify customer

13.6.21 Door Lock CC

Table 633 – Interface object for Door Lock CC

Name	Value Type	JSON Data Type	Description
cap_bitmask	uint8_t	number	Door lock capabilities bitmask as defined in ZW_DLCK_CAP_XXX
supp_in_handle_bitmask	uint8_t	number	Supported inside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if the corresponding handle can be enabled and disabled; else the corresponding handle cannot be enabled or disabled.
supp_out_handle_bitmask	uint8_t	number	Supported outside door handle mode bitmask. It's a 4-bit mask; bit set to 1 if

			the corresponding handle can be enabled and disabled; else the corresponding handle cannot be enabled or disabled.
supp_component_bitmask	uint8_t	number	Supported door lock component bitmask (ZW_COND_XXX) that can be reported in the Door Lock condition field of the Door Lock Operation Report Command
supp_operation_type	uint8_t []	number array	Supported operation types (ZW_DOOR_OP_XXX) stored in uint8_t number array.
supp_mode	uint8_t []	number array	Supported modes (ZW_DOOR_XXX) stored in uint8_t number array.
cache		array	Door lock Cache object array. The array must not contain more than one cache object for “Operation Status” and “Configuration” respectively.

Table 634 – Door Lock Cache object for “Operation Status”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
out_mode	uint8_t	number	Outside door handles mode. It's a 4-bit mask; bit=0 for disabled, bit=1 for enabled. When disabled, the actual handle cannot open the door locally. When enabled, the actual handle can open the door locally
in_mode	uint8_t	number	Inside door handles mode. It's a 4-bit mask; bit=0 for disabled, bit=1 for enabled
mode	uint8_t	number	Current door lock mode (ZW_DOOR_XXX)
cond	uint8_t	number	Door condition (i.e. status) bitmask as defined in ZW_COND_XXX_MASK
tmout_m	uint8_t	number	The remaining time in minutes before the door lock will automatically be locked again. Value of 0xFE means timeout is not supported
tmout_s	uint8_t	number	The remaining time in seconds before the door lock will automatically be locked again. Value of 0xFE means timeout is not supported
tgt_mode	uint8_t	number	Target door lock mode (ZW_DOOR_XXX)
dur	uint8_t	number	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved

Table 635 – Door Lock Cache object for “Configuration”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
type	uint8_t	number	Door lock operation type (ZW_DOOR_OP_XXX)
out_hdl_stat	uint8_t	number	Outside door handles state. It's a 4-bit mask; bit set to 0 if disabled; bit set to 1 if enabled. When disabled, the actual handle cannot open the door locally. When enabled, the actual handle can open the door locally.
in_hdl_stat	uint8_t	number	Inside door handles state. It's a 4-bit mask; bit=0 for disable, bit=1 for enable
tmout_m	uint8_t	number	Lock timeout in minutes. Valid value: 0 to 253. Value of 0xFE means timeout is not supported
tmout_s	uint8_t	number	Lock timeout in seconds. Valid value: 0 to 59. Value of 0xFE means timeout is not supported
blk_to_blk	uint8_t	number	Flag to indicate if the block-to-block functionality is enabled. Non-zero means enabled; zero means disabled
twist_asst	uint8_t	number	Flag to indicate if the twist assist functionality is enabled. Non-zero means enabled; zero means disabled
auto_rlck_tm	uint16_t	number	Time setting in seconds for auto-relock functionality. Zero means the functionality is disabled
hold_rel_tm	uint16_t	number	Time setting in seconds for letting the latch retracted after the supporting node's mode has been changed to unsecured. Zero means the functionality is disabled

13.6.22 Door Lock Logging CC

Table 636 – Interface object for Door Lock Logging CC

Name	Value Type	JSON Data Type	Description
max_logs	uint8_t	number	Maximum number of reports the audit trail supports

13.6.23 Alarm Sensor CC

Table 637 – Interface object for Alarm Sensor CC

Name	Value Type	JSON Data Type	Description
supp_sensor	uint8_t []	number array	Supported sensor types stored in uint8_t number array.
cache		array	Alarm sensor Cache object array

Table 638 – Alarm Sensor Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
type	uint8_t	number	Alarm Sensor type as defined in ZW_ALRM_SNSR_TYPE_XXX
state	uint8_t	number	Sensor state: 0=no alarm; 0xFF=alarm; 1-99: alarm severity in percentage
src_node_id	uint8_t	number	Source node ID with the alarm condition. Not valid in Zensor Net
active_tm	uint16_t	number	Alarm active time since last received report in seconds. Zero means this field must be ignored.

13.6.24 Barrier Operator CC

Table 639 – Interface object for Barrier Operator CC

Name	Value Type	JSON Data Type	Description
supp_subsys_type	uint8_t []	number array	Supported event signaling capabilities stored in uint8_t number array.
cache		array	Barrier Operator Cache object array

Table 640 – Barrier Operator Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
sub_type	uint8_t	number	Subsystem type as defined in ZW_BAR_NOTIF_TYP_XXX. If zero, **sub_state** represents barrier operator state.
sub_state	uint8_t	number	Subsystem state: 0=off; 0xFF=on

13.6.25 Color Switch CC

Table 641 – Interface object for Color Switch CC

Name	Value Type	JSON Data Type	Description
supp_color_comp	uint8_t []	number array	Supported color components stored in uint8_t number array.
cache		array	Color Switch Cache object array

Table 642 – Color Switch Cache object

Name	Value Type	JSON Data Type	Description

ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
id	uint8_t	number	Color component id as defined in COL_SW_COMP_ID_XXX.
value	uint8_t	number	Value
tgt_value	uint8_t	number	Target value
dur	uint8_t	number	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved

13.6.26 Sound Switch CC

Table 643 – Interface object for Sound Switch CC

Name	Value Type	JSON Data Type	Description
total_tone_supp	uint8_t	number	Total number of tones supported
tone_info		array	Sound Switch object array
cache		array	Sound Switch Cache object array. The array must not contain more than one cache object for “Tone Played” and “Tone Configuration” respectively.

Table 644 – Sound Switch object

Name	Value Type	JSON Data Type	Description
id	uint8_t	number	Tone ID
duration	uint16_t	number	Tone duration in seconds
name	char []	string	Name or label for the Tone ID

Table 645 – Sound Switch Cache object for “Tone Played”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
current_tone	uint8_t	number	Tone ID of current tone being played
current_volume	uint8_t	number	Current tone playing volume (Note: Applicable only to Sound Switch CC version 2 and above)

Table 646 – Sound Switch Cache object for “Tone Configuration”

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected

volume	uint8_t	number	Volume of the device
default_tone	uint8_t	number	Default tone ID

13.6.27 Window Covering CC

Table 647 – Interface object for Window Covering CC

Name	Value Type	JSON Data Type	Description
supp_param_id	uint8_t []	number array	Supported parameters stored in uint8_t number array.
cache		array	Window Covering Cache object array

Table 648 – Window Covering Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
id	uint8_t	number	Parameter id as defined in WIN_COVER_ID_XXX.
value	uint8_t	number	Current value with range 0~99. 0=Closed; 99=Open
tgt_value	uint8_t	number	Target value with same interpretation as value
dur	uint8_t	number	Duration needed to reach the target value: 0 = already at the target; 0x01~0x7F = seconds in 1-second resolution; 0x80~0xFD = 1~126 minutes in 1-minute resolution; 0xFE = Unknown duration; 0xFF = reserved

13.6.28 Battery CC

Table 649 – Interface object for Battery CC

Name	Value Type	JSON Data Type	Description
cache		array	Battery Cache object array

Table 650 – Battery Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp of battery report
level	uint8_t	number	Battery level. Values 0x00 to 0x64 indicate the battery percentage level from 0 to 100%. Level 0xFF indicates a low-battery warning
rechargeable*	uint8_t	number	Flag to indicate if the battery is rechargeable. 1=rechargeable; 0=non-rechargeable

charging_status*	uint8_t	number	Charging status if battery is rechargeable, BATT_STS_XXX. For non-rechargeable battery, set this value to 0.
backup_battery*	uint8_t	number	Flag to indicate if the battery is utilized for back-up purposes of a main powered connected device. 1=backup; 0=battery is used for primary means of power
overheating*	uint8_t	number	Flag to indicate if overheating is detected. 1=overheating; 0=operating within the normal temperature range
low_fluid*	uint8_t	number	Flag to indicate if the battery fluid is low and should be refilled. 1=low; 0=normal
recharging_req*	uint8_t	number	Status to indicate if the battery needs to be recharged (for rechargeable battery) or replaced (for non-rechargeable battery), BATT_RECHG_XXX
disconnected*	uint8_t	number	Flag to indicate if the battery is currently disconnected or removed from the node. 1=disconnected and the node is running on an alternative power source; 0=connected
health_ts#	time_t	number	Time stamp of battery health report
max_capacity#	uint8_t	number	The percentage indicating the maximum capacity of the battery. Values in the range 0x00..0x64 indicate the maximum capacity of the battery in the percentage level from 0 to 100%. Value of 0xFF indicates the maximum capacity of the battery is unknown
precision#	uint8_t	number	Decimal places of the battery temperature value. E.g. the decimal value 1025 with precision 2 is equal to 10.25
unit#	uint8_t	number	Unit used in battery temperature. 0=Celsius; other values are reserved
temperature_value#	uint8_t []	number array	Temperature value stored in uint8_t number array. The number of items in the array is determined by the value data size which could be 1, 2 or 4. The first item in the array stores the most significant byte (MSB) of the value. **Absent** of this field indicates the battery temperature is **unknown**.

Note: Those marked with (*) are only applicable to Battery Report from Battery CC version 2 and above.

Those marked with (#) are only applicable to Battery Health Report from Battery CC version 2 and above.

13.6.29 Basic CC

Table 651 – Interface object for Basic CC

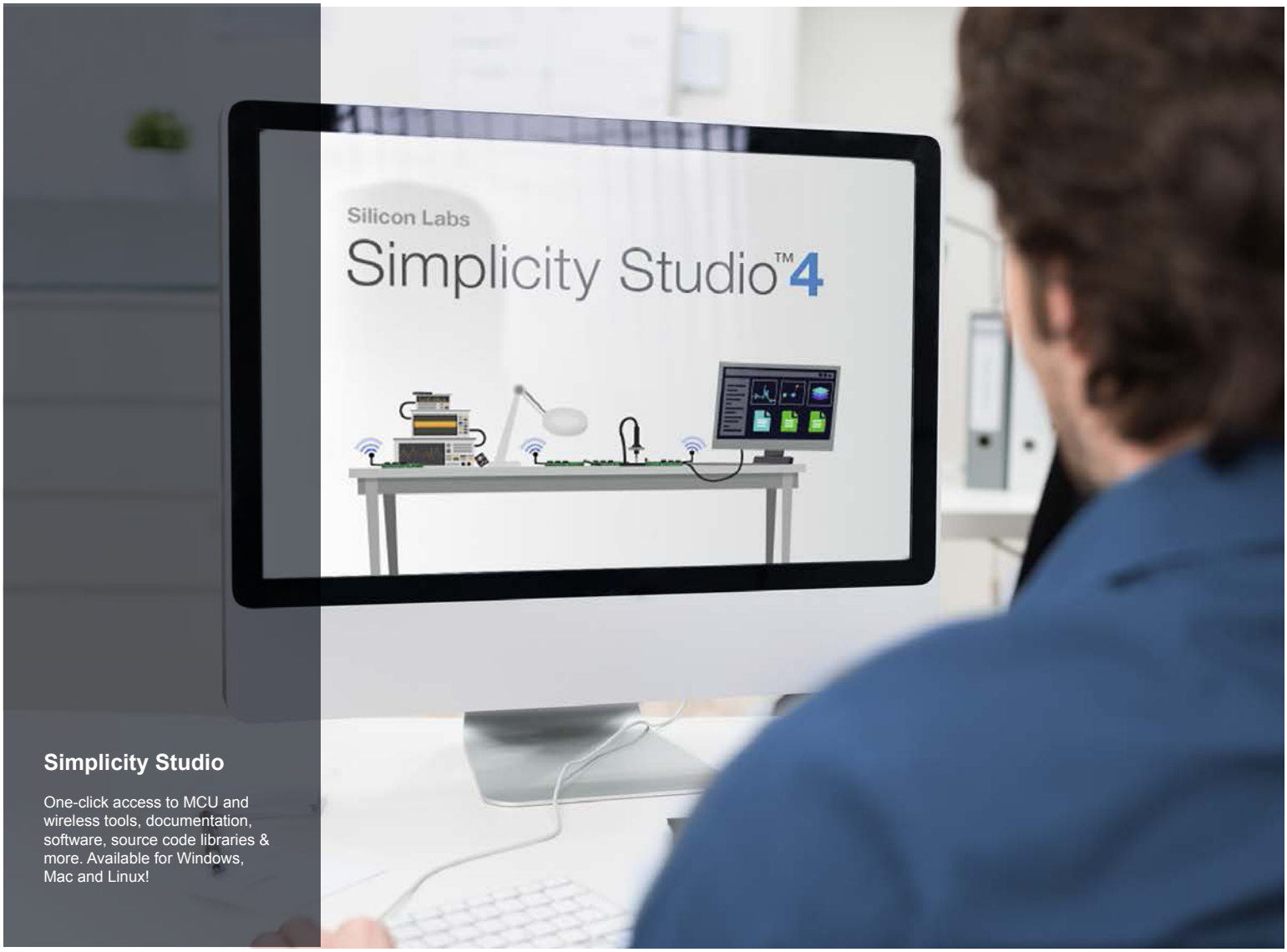
Name	Value Type	JSON Data Type	Description
cache		array	Basic Cache object array

Table 652 – Basic Cache object

Name	Value Type	JSON Data Type	Description
ts	time_t	number	Time stamp
sn	uint16_t	number	State number that is incremented by one whenever a cache value change is detected
curr_val	uint8_t	number	Current value: 0 = off; 0x01~0x63 = percentage (%);0xFE = Unknown; 0xFF = On
tgt_val	uint8_t	number	Target value with same interpretation as curr_val
dur	uint8_t	number	Duration: 0 = already at the target; 0x01~0x7F = seconds; 0x80~0xFD = 1~126 minutes; 0xFE = Unknown duration; 0xFF = reserved

References

[1] Silicon Labs, SDS10243, SDS, Z-Wave Protocol Overview
[2] Silicon Labs, INS10244, INS, Z-Wave Node Type Overview and Network Installation Guide
[3] Silicon Labs, SDS10242, SDS, Z-Wave Device Classes
[4] Silicon Labs, SDS13781, SDS, Z-Wave Application Command Class Specification
[5] Silicon Labs, SDS13782, SDS, Z-Wave Management Command Class Specification
[6] Silicon Labs, SDS13783, SDS, Z-Wave Transport-Encapsulation Command Class Specification
[7] Silicon Labs, SDS13784, SDS, Z-Wave Network-Protocol Command Class Specification



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com