

# Host Libraries API Documentation

Generated by Doxygen 1.7.1

Wed Mar 28 2012 00:44:15



# Contents

<b>1</b>	<b>Host Libraries</b>	<b>1</b>
<b>2</b>	<b>Deprecated List</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>5</b>
3.1	Modules . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Setup . . . . .	9
5.1.1	Function Documentation . . . . .	11
5.1.1.1	apdm_apply_autoconfigure_sensor_config . . .	11
5.1.1.2	apdm_autoconfigure_devices_and_accesspoint	11
5.1.1.3	apdm_autoconfigure_devices_and_accesspoint2	11
5.1.1.4	apdm_autoconfigure_devices_and_accesspoint3	12
5.1.1.5	apdm_autoconfigure_devices_and_accesspoint4	12
5.1.1.6	apdm_autoconfigure_devices_and_- accesspoint_streaming . . . . .	13
5.1.1.7	apdm_autoconfigure_devices_and_- accesspoint_wireless . . . . .	13
5.1.1.8	apdm_autoconfigure_mesh_sync . . . . .	14
5.1.1.9	apdm_autoconfigure_mesh_sync2 . . . . .	15

5.1.1.10	apdm_calibration_override_minimum_supported_version . . . . .	15
5.1.1.11	apdm_configure_all_attached_sensors . . . . .	15
5.1.1.12	apdm_configure_all_attached_sensors_mesh . . . . .	16
5.1.1.13	apdm_ctx_ap_get_gpio_value . . . . .	17
5.1.1.14	apdm_ctx_ap_get_io_value . . . . .	17
5.1.1.15	apdm_ctx_ap_set_gpio_value . . . . .	18
5.1.1.16	apdm_ctx_ap_set_io_value . . . . .	18
5.1.1.17	apdm_ctx_autoconfigure_devices_and_accesspoint5 . . . . .	19
5.1.1.18	apdm_ctx_open_all_access_points . . . . .	19
5.1.1.19	apdm_ctx_set_correlation_fifo_temp_directory . . . . .	19
5.1.1.20	apdm_exit . . . . .	20
5.1.1.21	apdm_init_access_point_wireless . . . . .	20
5.1.1.22	apdm_init_streaming_config . . . . .	21
5.2	Context . . . . .	21
5.2.1	Function Documentation . . . . .	24
5.2.1.1	apdm_ctx_allocate_new_context . . . . .	24
5.2.1.2	apdm_ctx_avg_retry_count_for_device . . . . .	24
5.2.1.3	apdm_ctx_disable_accesspoint_wireless . . . . .	24
5.2.1.4	apdm_ctx_disconnect . . . . .	25
5.2.1.5	apdm_ctx_estimate_now_sync_value . . . . .	25
5.2.1.6	apdm_ctx_extract_data_by_device_id . . . . .	25
5.2.1.7	apdm_ctx_extract_next_sample . . . . .	26
5.2.1.8	apdm_ctx_flush_ap_fifos . . . . .	26
5.2.1.9	apdm_ctx_free_context . . . . .	27
5.2.1.10	apdm_ctx_get_ap_id_for_ap_index . . . . .	27
5.2.1.11	apdm_ctx_get_device_id_by_index . . . . .	27
5.2.1.12	apdm_ctx_get_device_id_list . . . . .	28
5.2.1.13	apdm_ctx_get_device_index_by_id3 . . . . .	28
5.2.1.14	apdm_ctx_get_device_info . . . . .	28
5.2.1.15	apdm_ctx_get_expected_number_of_sensors2 . . . . .	29

5.2.1.16 apdm_ctx_get_expected_sync_delta . . . . .	29
5.2.1.17 apdm_ctx_get_last_received_timestamp_for_- device . . . . .	30
5.2.1.18 apdm_ctx_get_metadata_uint32 . . . . .	30
5.2.1.19 apdm_ctx_get_monitor_latency . . . . .	30
5.2.1.20 apdm_ctx_get_next_access_point_record . . .	31
5.2.1.21 apdm_ctx_get_next_access_point_record_list .	31
5.2.1.22 apdm_ctx_get_next_record . . . . .	32
5.2.1.23 apdm_ctx_get_next_synchronization_event . . .	32
5.2.1.24 apdm_ctx_get_num_access_points_found . . .	33
5.2.1.25 apdm_ctx_get_num_omitted_sample_sets . . .	33
5.2.1.26 apdm_ctx_get_num_omitted_samples . . . . .	33
5.2.1.27 apdm_ctx_get_num_sample_lists_collected . .	34
5.2.1.28 apdm_ctx_get_num_samples_collected . . . . .	34
5.2.1.29 apdm_ctx_get_num_samples_collected_- from_device . . . . .	34
5.2.1.30 apdm_ctx_get_sampling_frequency . . . . .	34
5.2.1.31 apdm_ctx_get_sensor_compensation_data . . .	35
5.2.1.32 apdm_ctx_get_total_omitted_sample_sets . . .	35
5.2.1.33 apdm_ctx_get_total_omitted_samples . . . . .	35
5.2.1.34 apdm_ctx_get_wireless_configuration_mode . .	35
5.2.1.35 apdm_ctx_get_wireless_reliability_value . . . .	36
5.2.1.36 apdm_ctx_get_wireless_streaming_status . . .	36
5.2.1.37 apdm_ctx_initialize_context . . . . .	37
5.2.1.38 apdm_ctx_is_more_data_immediately_available	37
5.2.1.39 apdm_ctx_re_enable_accesspoint_wireless . .	37
5.2.1.40 apdm_ctx_reset_num_samples_from_ap . . . .	38
5.2.1.41 apdm_ctx_set_error_handling_mode . . . . .	38
5.2.1.42 apdm_ctx_set_max_sample_delay_seconds . .	38
5.2.1.43 apdm_ctx_set_metadata_string . . . . .	39
5.2.1.44 apdm_ctx_set_metadeta_uint32 . . . . .	39
5.2.1.45 apdm_ctx_set_sensor_compensation_data . . .	40

5.2.1.46	<a href="#">apdm_ctx_sync_record_list_head</a>	40
5.2.1.47	<a href="#">apdm_get_max_sample_delay_seconds</a>	41
5.2.1.48	<a href="#">apdm_get_num_samples_from_ap</a>	41
5.3	<a href="#">AccessPoint</a>	41
5.3.1	<a href="#">Function Documentation</a>	43
5.3.1.1	<a href="#">adpm_ap_set_max_latency_value</a>	43
5.3.1.2	<a href="#">adpm_ap_set_max_latency_value_seconds</a>	44
5.3.1.3	<a href="#">apdm_ap_allocate_handle</a>	44
5.3.1.4	<a href="#">apdm_ap_connect</a>	44
5.3.1.5	<a href="#">apdm_ap_disconnect</a>	45
5.3.1.6	<a href="#">apdm_ap_get_board_version_string</a>	45
5.3.1.7	<a href="#">apdm_ap_get_case_id</a>	46
5.3.1.8	<a href="#">apdm_ap_get_gpio_value</a>	46
5.3.1.9	<a href="#">apdm_ap_get_id</a>	47
5.3.1.10	<a href="#">apdm_ap_get_id_and_board_version</a>	47
5.3.1.11	<a href="#">apdm_ap_get_io_value</a>	47
5.3.1.12	<a href="#">apdm_ap_get_mode</a>	48
5.3.1.13	<a href="#">apdm_ap_get_monitor_latency</a>	48
5.3.1.14	<a href="#">apdm_ap_get_num_access_points_on_host1</a>	49
5.3.1.15	<a href="#">apdm_ap_get_protocol_subversion</a>	49
5.3.1.16	<a href="#">apdm_ap_get_version</a>	49
5.3.1.17	<a href="#">apdm_ap_get_version_string</a>	50
5.3.1.18	<a href="#">apdm_ap_get_wireless_streaming_led_status</a>	50
5.3.1.19	<a href="#">apdm_ap_init_handle</a>	50
5.3.1.20	<a href="#">apdm_ap_override_minimum_supported_version</a>	51
5.3.1.21	<a href="#">apdm_ap_reset_into_bootloader</a>	51
5.3.1.22	<a href="#">apdm_ap_reset_into_firmware</a>	51
5.3.1.23	<a href="#">apdm_ap_set_error_blink_threshold</a>	52
5.3.1.24	<a href="#">apdm_ap_set_gpio_value</a>	52
5.3.1.25	<a href="#">apdm_ap_set_io_value</a>	53
5.3.1.26	<a href="#">apdm_ap_set_warning_blink_threshold</a>	53

5.3.1.27	<a href="#">apdm_ap_verify_supported_version</a>	53
5.3.1.28	<a href="#">apdm_ap_wireless_streaming_status_t_str</a>	54
5.3.1.29	<a href="#">apdm_configure_accesspoint</a>	54
5.3.1.30	<a href="#">apdm_ctx_get_all_ap_debug_info</a>	55
5.3.1.31	<a href="#">apdm_free_ap_handle</a>	55
5.3.1.32	<a href="#">apdm_send_accesspoint_cmd</a>	55
5.4	<a href="#">DataFiles</a>	56
5.4.1	<a href="#">Function Documentation</a>	57
5.4.1.1	<a href="#">apdm_close_file_csv</a>	57
5.4.1.2	<a href="#">apdm_close_file_hdf</a>	57
5.4.1.3	<a href="#">apdm_create_file_csv</a>	58
5.4.1.4	<a href="#">apdm_create_file_hdf</a>	58
5.4.1.5	<a href="#">apdm_get_hdf_dataset_shape</a>	58
5.4.1.6	<a href="#">apdm_get_hdf_device_list</a>	59
5.4.1.7	<a href="#">apdm_get_hdf_device_list_swig</a>	59
5.4.1.8	<a href="#">apdm_get_hdf_label_list</a>	60
5.4.1.9	<a href="#">apdm_get_hdf_label_list_swig</a>	60
5.4.1.10	<a href="#">apdm_process_raw</a>	61
5.4.1.11	<a href="#">apdm_read_hdf_calibration_data</a>	61
5.4.1.12	<a href="#">apdm_read_hdf_dataset</a>	62
5.4.1.13	<a href="#">apdm_read_hdf_timestamps</a>	63
5.4.1.14	<a href="#">apdm_read_raw_file_info</a>	63
5.4.1.15	<a href="#">apdm_write_annotation</a>	64
5.4.1.16	<a href="#">apdm_write_record_csv</a>	64
5.4.1.17	<a href="#">apdm_write_record_hdf</a>	65
5.5	<a href="#">Monitor</a>	66
5.5.1	<a href="#">Function Documentation</a>	67
5.5.1.1	<a href="#">apdm_device_extract_module_id_from_case_id_string</a>	67
5.5.1.2	<a href="#">apdm_halt_all_attached_sensors</a>	67
5.5.1.3	<a href="#">apdm_initialize_device_info</a>	67

5.5.1.4	<a href="#">apdm_sensor_allocate_and_open</a>	68
5.5.1.5	<a href="#">apdm_sensor_allocate_handle</a>	68
5.5.1.6	<a href="#">apdm_sensor_apply_configuration</a>	69
5.5.1.7	<a href="#">apdm_sensor_close</a>	69
5.5.1.8	<a href="#">apdm_sensor_comm_channel_verify_supported_version</a>	69
5.5.1.9	<a href="#">apdm_sensor_free_handle</a>	70
5.5.1.10	<a href="#">apdm_sensor_get_device_id_list</a>	70
5.5.1.11	<a href="#">apdm_sensor_list_attached_sensors</a>	70
5.5.1.12	<a href="#">apdm_sensor_list_attached_sensors3</a>	71
5.5.1.13	<a href="#">apdm_sensor_open</a>	71
5.5.1.14	<a href="#">apdm_sensor_override_minimum_supported_version</a>	72
5.5.1.15	<a href="#">apdm_sensor_populate_device_info</a>	72
5.5.1.16	<a href="#">apdm_sensor_verify_supported_calibration_version</a>	73
5.5.1.17	<a href="#">apdm_sensor_verify_supported_version</a>	73
5.6	<a href="#">MonitorCommands</a>	73
5.6.1	<a href="#">Function Documentation</a>	77
5.6.1.1	<a href="#">apdm_sensor_close_and_free</a>	77
5.6.1.2	<a href="#">apdm_sensor_cmd_battery_charge_rate</a>	78
5.6.1.3	<a href="#">apdm_sensor_cmd_battery_charge_status</a>	78
5.6.1.4	<a href="#">apdm_sensor_cmd_battery_voltage</a>	79
5.6.1.5	<a href="#">apdm_sensor_cmd_bootloader_version</a>	79
5.6.1.6	<a href="#">apdm_sensor_cmd_calibration_data</a>	79
5.6.1.7	<a href="#">apdm_sensor_cmd_calibration_data_blob</a>	80
5.6.1.8	<a href="#">apdm_sensor_cmd_calibration_version</a>	80
5.6.1.9	<a href="#">apdm_sensor_cmd_case_id</a>	80
5.6.1.10	<a href="#">apdm_sensor_cmd_config_check</a>	81
5.6.1.11	<a href="#">apdm_sensor_cmd_config_commit</a>	81
5.6.1.12	<a href="#">apdm_sensor_cmd_config_get</a>	81
5.6.1.13	<a href="#">apdm_sensor_cmd_config_set</a>	82



5.6.1.14 apdm_sensor_cmd_config_status . . . . .	82
5.6.1.15 apdm_sensor_cmd_debug_get . . . . .	83
5.6.1.16 apdm_sensor_cmd_debug_set . . . . .	83
5.6.1.17 apdm_sensor_cmd_device_id . . . . .	83
5.6.1.18 apdm_sensor_cmd_dock . . . . .	84
5.6.1.19 apdm_sensor_cmd_dock_status . . . . .	84
5.6.1.20 apdm_sensor_cmd_enter_bootloader . . . . .	84
5.6.1.21 apdm_sensor_cmd_error_clear . . . . .	85
5.6.1.22 apdm_sensor_cmd_error_count . . . . .	85
5.6.1.23 apdm_sensor_cmd_error_log_get . . . . .	85
5.6.1.24 apdm_sensor_cmd_error_log_size . . . . .	86
5.6.1.25 apdm_sensor_cmd_error_name . . . . .	86
5.6.1.26 apdm_sensor_cmd_error_stats_get . . . . .	86
5.6.1.27 apdm_sensor_cmd_error_stats_size . . . . .	87
5.6.1.28 apdm_sensor_cmd_flash_block_get . . . . .	87
5.6.1.29 apdm_sensor_cmd_flash_block_set . . . . .	87
5.6.1.30 apdm_sensor_cmd_flash_format . . . . .	87
5.6.1.31 apdm_sensor_cmd_halt . . . . .	88
5.6.1.32 apdm_sensor_cmd_hw_id . . . . .	88
5.6.1.33 apdm_sensor_cmd_last_standby_uptime . . . . .	88
5.6.1.34 apdm_sensor_cmd_last_uptime . . . . .	89
5.6.1.35 apdm_sensor_cmd_led_pattern . . . . .	89
5.6.1.36 apdm_sensor_cmd_led_reset . . . . .	89
5.6.1.37 apdm_sensor_cmd_memory_crc16 . . . . .	90
5.6.1.38 apdm_sensor_cmd_memory_dump . . . . .	90
5.6.1.39 apdm_sensor_cmd_off_reason . . . . .	91
5.6.1.40 apdm_sensor_cmd_peek . . . . .	91
5.6.1.41 apdm_sensor_cmd_peek2 . . . . .	91
5.6.1.42 apdm_sensor_cmd_ping . . . . .	92
5.6.1.43 apdm_sensor_cmd_poke . . . . .	92
5.6.1.44 apdm_sensor_cmd_poke2 . . . . .	92

5.6.1.45 apdm_sensor_cmd_protocol_version . . . . .	93
5.6.1.46 apdm_sensor_cmd_reset . . . . .	93
5.6.1.47 apdm_sensor_cmd_run . . . . .	93
5.6.1.48 apdm_sensor_cmd_sample_get . . . . .	94
5.6.1.49 apdm_sensor_cmd_sample_start . . . . .	94
5.6.1.50 apdm_sensor_cmd_standby . . . . .	95
5.6.1.51 apdm_sensor_cmd_stats_clear . . . . .	95
5.6.1.52 apdm_sensor_cmd_stats_count_get . . . . .	95
5.6.1.53 apdm_sensor_cmd_stats_max_get . . . . .	96
5.6.1.54 apdm_sensor_cmd_stats_min_get . . . . .	96
5.6.1.55 apdm_sensor_cmd_stats_size . . . . .	96
5.6.1.56 apdm_sensor_cmd_stats_sum_get . . . . .	96
5.6.1.57 apdm_sensor_cmd_sync_commit . . . . .	97
5.6.1.58 apdm_sensor_cmd_sync_dock_wait . . . . .	97
5.6.1.59 apdm_sensor_cmd_sync_get . . . . .	97
5.6.1.60 apdm_sensor_cmd_sync_set . . . . .	98
5.6.1.61 apdm_sensor_cmd_time_get . . . . .	98
5.6.1.62 apdm_sensor_cmd_time_set . . . . .	99
5.6.1.63 apdm_sensor_cmd_time_set2 . . . . .	99
5.6.1.64 apdm_sensor_cmd_timer_adjust_get . . . . .	100
5.6.1.65 apdm_sensor_cmd_undock . . . . .	100
5.6.1.66 apdm_sensor_cmd_unlock_bootloader_flash . . . . .	100
5.6.1.67 apdm_sensor_cmd_uptime_get . . . . .	100
5.6.1.68 apdm_sensor_cmd_uptime_reset . . . . .	101
5.6.1.69 apdm_sensor_cmd_user_calibration_data . . . . .	101
5.6.1.70 apdm_sensor_cmd_user_calibration_data_blob . . . . .	101
5.6.1.71 apdm_sensor_cmd_version_string_1 . . . . .	102
5.6.1.72 apdm_sensor_cmd_version_string_2 . . . . .	102
5.6.1.73 apdm_sensor_cmd_version_string_3 . . . . .	102
5.6.1.74 apdm_sensor_cmd_write_flash_block . . . . .	103
5.6.1.75 apdm_sensor_config_get_label . . . . .	103

5.6.1.76	<a href="#">apdm_sensor_config_set_label</a>	104
5.6.1.77	<a href="#">apdm_sensor_get_monitor_type</a>	104
5.7	<a href="#">DockingStation</a>	104
5.7.1	<a href="#">Function Documentation</a>	105
5.7.1.1	<a href="#">apdm_ds_get_case_id</a>	105
5.7.1.2	<a href="#">apdm_ds_get_docked_module_id</a>	105
5.7.1.3	<a href="#">apdm_ds_get_firmware_version</a>	106
5.7.1.4	<a href="#">apdm_ds_get_hardware_version</a>	106
5.7.1.5	<a href="#">apdm_ds_get_index_by_serial_number</a>	107
5.7.1.6	<a href="#">apdm_ds_get_protocol_subversion</a>	107
5.7.1.7	<a href="#">apdm_ds_get_serial</a>	107
5.7.1.8	<a href="#">apdm_ds_get_serial_number_by_index</a>	108
5.7.1.9	<a href="#">apdm_ds_is_monitor_data_forwarding_enabled</a>	108
5.7.1.10	<a href="#">apdm_ds_is_monitor_present</a>	108
5.7.1.11	<a href="#">apdm_ds_override_minimum_supported_version</a>	109
5.7.1.12	<a href="#">apdm_sensor_get_num_attached_dockingstations1</a>	109
5.8	<a href="#">DataHandling</a>	109
5.8.1	<a href="#">Function Documentation</a>	110
5.8.1.1	<a href="#">apdm_calculate_sync_value_age</a>	110
5.8.1.2	<a href="#">apdm_epoch_access_point_to_epoch_microsecond</a>	110
5.8.1.3	<a href="#">apdm_epoch_access_point_to_epoch_millisecond</a>	111
5.8.1.4	<a href="#">apdm_epoch_access_point_to_epoch_second</a>	111
5.8.1.5	<a href="#">apdm_epoch_second_to_epoch_access_point</a>	111
5.8.1.6	<a href="#">apdm_extract_next_sample_set</a>	112
5.8.1.7	<a href="#">apdm_recalibrate_gyroscopes</a>	112
5.8.1.8	<a href="#">apdm_recalibrate_gyroscopes_from_h5</a>	113
5.8.1.9	<a href="#">apdm_recalibrate_magnetometers</a>	113
5.8.1.10	<a href="#">apdm_recalibrate_magnetometers_from_h5</a>	114
5.9	<a href="#">Logging</a>	115

5.9.1	Function Documentation . . . . .	116
5.9.1.1	apdm_close_log_file . . . . .	116
5.9.1.2	apdm_log . . . . .	116
5.9.1.3	apdm_log_context . . . . .	116
5.9.1.4	apdm_log_debug . . . . .	117
5.9.1.5	apdm_log_error . . . . .	117
5.9.1.6	apdm_log_info . . . . .	118
5.9.1.7	apdm_log_warning . . . . .	119
5.9.1.8	apdm_logging_level_t_str . . . . .	119
5.9.1.9	apdm_logl . . . . .	119
5.9.1.10	apdm_set_log_file . . . . .	120
5.9.1.11	apdm_set_log_level . . . . .	120
5.10	Misc . . . . .	120
5.10.1	Function Documentation . . . . .	121
5.10.1.1	apdm_error_severity . . . . .	121
5.10.1.2	apdm_get_library_build_datetime . . . . .	122
5.10.1.3	apdm_get_library_version . . . . .	122
5.10.1.4	apdm_get_now_sync_value_host . . . . .	122
5.10.1.5	apdm_get_time_ms_64 . . . . .	122
5.10.1.6	apdm_monitor_decimation_rate_t_str . . . . .	123
5.10.1.7	apdm_monitor_decimation_rate_t_to_int . . . . .	123
5.10.1.8	apdm_monitor_error_id_str . . . . .	123
5.10.1.9	apdm_monitor_get_expected_sync_delta . . . . .	124
5.10.1.10	apdm_monitor_output_select_rate_t_to_int . . . . .	124
5.10.1.11	apdm_msleep . . . . .	124
5.10.1.12	apdm_output_select_rate_t_str . . . . .	125
5.10.1.13	apdm_streaming_config_get_device_info . . . . .	125
5.10.1.14	apdm_strerror . . . . .	125
5.10.1.15	apdm_usleep . . . . .	126
5.10.1.16	apdm_wireless_mode_t_str . . . . .	126

<b>6 Class Documentation</b>	<b>127</b>
6.1 <a href="#">__attribute__ Struct Reference</a>	127
6.2 <a href="#">apdm_access_point_configuration_t Struct Reference</a>	130
6.3 <a href="#">apdm_access_point_handle Struct Reference</a>	130
6.3.1 Detailed Description	131
6.4 <a href="#">apdm_annotation_t Struct Reference</a>	132
6.5 <a href="#">apdm_bulk_in_buffer_t Struct Reference</a>	132
6.6 <a href="#">apdm_byte_array_ring_buffer_t Struct Reference</a>	132
6.7 <a href="#">apdm_case_id_t Struct Reference</a>	133
6.8 <a href="#">apdm_context_t Struct Reference</a>	133
6.8.1 Detailed Description	134
6.9 <a href="#">apdm_device_dtemp_filter_state_t Struct Reference</a>	134
6.9.1 Member Data Documentation	134
6.9.1.1 <a href="#">error_covariance_matrix</a>	134
6.9.1.2 <a href="#">filtered_measurement</a>	134
6.9.1.3 <a href="#">measurement</a>	134
6.9.1.4 <a href="#">measurement_matrix</a>	134
6.9.1.5 <a href="#">measurement_noise_matrix</a>	135
6.9.1.6 <a href="#">process_noise_matrix</a>	135
6.9.1.7 <a href="#">state_transition_matrix</a>	135
6.10 <a href="#">apdm_device_fifo_t Struct Reference</a>	135
6.11 <a href="#">apdm_device_info_t Struct Reference</a>	135
6.11.1 Detailed Description	137
6.11.2 Member Data Documentation	137
6.11.2.1 <a href="#">decimation_factor</a>	137
6.11.2.2 <a href="#">dock_id_during_configuration</a>	137
6.11.2.3 <a href="#">protocol_version</a>	137
6.11.2.4 <a href="#">timezone</a>	137
6.11.2.5 <a href="#">wireless_addr_id</a>	137
6.11.2.6 <a href="#">wireless_block0</a>	138
6.11.2.7 <a href="#">wireless_block1</a>	138

6.11.2.8 wireless_block2 . . . . .	138
6.11.2.9 wireless_block3 . . . . .	138
6.11.2.10 wireless_channel1 . . . . .	138
6.11.2.11 wireless_channel2 . . . . .	139
6.11.2.12 wireless_channel3 . . . . .	139
6.11.2.13 wireless_timeslice . . . . .	139
6.12 apdm_device_sample_buffer_row_t Struct Reference . . . . .	139
6.13 apdm_device_state_data_t Struct Reference . . . . .	140
6.14 apdm_device_status_t Struct Reference . . . . .	140
6.14.1 Detailed Description . . . . .	141
6.15 apdm_disk_ll_t Struct Reference . . . . .	141
6.16 apdm_external_sync_data_t Struct Reference . . . . .	141
6.17 apdm_mag_dechop_state_t Struct Reference . . . . .	142
6.17.1 Detailed Description . . . . .	143
6.17.2 Member Data Documentation . . . . .	143
6.17.2.1 error_covariance_matrix . . . . .	143
6.17.2.2 filtered_measurement . . . . .	143
6.17.2.3 iSample . . . . .	143
6.17.2.4 measurement . . . . .	143
6.17.2.5 measurement_matrix . . . . .	143
6.17.2.6 measurement_noise_matrix . . . . .	143
6.17.2.7 polarity . . . . .	143
6.17.2.8 process_noise_matrix . . . . .	143
6.17.2.9 state_transition_matrix . . . . .	143
6.17.2.10 stepResponse . . . . .	143
6.17.2.11 stepResponseEstimate . . . . .	143
6.18 apdm_mag_opt_data_t Struct Reference . . . . .	144
6.19 apdm_mag_step_response_state_t Struct Reference . . . . .	144
6.19.1 Member Data Documentation . . . . .	145
6.19.1.1 error_covariance_matrix . . . . .	145
6.19.1.2 filtered_measurement . . . . .	145

6.19.1.3 measurement	145
6.19.1.4 measurement_matrix	145
6.19.1.5 measurement_noise_matrix	145
6.19.1.6 process_noise_matrix	145
6.19.1.7 state_transition_matrix	145
6.20 apdm_magnetometer_recalibration_t Struct Reference	145
6.21 apdm_monitor_error_stat_t Struct Reference	146
6.22 apdm_monitor_label_t Struct Reference	146
6.23 apdm_orientation_info_t Struct Reference	146
6.24 apdm_orientation_info_ukf_t Struct Reference	147
6.25 apdm_orientation_ukf_fdata_t Struct Reference	148
6.26 apdm_orientation_ukf_hdata_t Struct Reference	148
6.27 apdm_progress_t Struct Reference	149
6.28 apdm_record_ll_disk_data_t Struct Reference	149
6.29 apdm_record_t Struct Reference	149
6.29.1 Detailed Description	151
6.29.2 Member Data Documentation	151
6.29.2.1 accl_full_scale_mode	151
6.29.2.2 accl_isPopulated	151
6.29.2.3 accl_x_axis	152
6.29.2.4 accl_y_axis	152
6.29.2.5 accl_y_axis_si	152
6.29.2.6 accl_z_axis	152
6.29.2.7 accl_z_axis_si	152
6.29.2.8 batt_voltage_isPopulated	152
6.29.2.9 battery_level	152
6.29.2.10 debug_data	153
6.29.2.11 device_info_isPopulated	153
6.29.2.12 device_info_serial_number	153
6.29.2.13 device_info_wireless_address	153
6.29.2.14 device_info_wireless_channel_id	153

6.29.2.15events_num_events . . . . .	153
6.29.2.16flag_accel_enabled . . . . .	153
6.29.2.17gyro_isPopulated . . . . .	153
6.29.2.18gyro_temperature_sensor_selected . . . . .	153
6.29.2.19gyro_x_axis . . . . .	153
6.29.2.20gyro_x_axis_si . . . . .	154
6.29.2.21gyro_y_axis . . . . .	154
6.29.2.22gyro_y_axis_si . . . . .	154
6.29.2.23gyro_z_axis . . . . .	154
6.29.2.24gyro_z_axis_si . . . . .	154
6.29.2.25mag_common_axis . . . . .	154
6.29.2.26mag_isPopulated . . . . .	154
6.29.2.27mag_x_axis . . . . .	155
6.29.2.28mag_x_axis_si . . . . .	155
6.29.2.29mag_y_axis . . . . .	155
6.29.2.30mag_z_axis . . . . .	155
6.29.2.31num_retrys . . . . .	155
6.29.2.32opt_select . . . . .	155
6.29.2.33source_ap_index . . . . .	155
6.29.2.34sync_val32_low . . . . .	156
6.29.2.35temperature_derivative_si . . . . .	156
6.30 apdm_recording_info_t Struct Reference . . . . .	156
6.31 apdm_sensor_cmd Struct Reference . . . . .	156
6.32 apdm_sensor_compensation_t Struct Reference . . . . .	157
6.32.1 Detailed Description . . . . .	157
6.33 apdm_sensor_device_handle_t Struct Reference . . . . .	157
6.33.1 Detailed Description . . . . .	158
6.34 apdm_sensor_response Struct Reference . . . . .	158
6.35 apdm_streaming_config_t Struct Reference . . . . .	160
6.35.1 Member Data Documentation . . . . .	160
6.35.1.1 accel_full_scale_mode . . . . .	160



6.35.1.2	<a href="#">apply_new_sensor_modes</a>	160
6.35.1.3	<a href="#">button_enable</a>	160
6.35.1.4	<a href="#">decimation_rate</a>	161
6.35.1.5	<a href="#">device_info_cache</a>	161
6.35.1.6	<a href="#">enable_accel</a>	161
6.35.1.7	<a href="#">enable_gyro</a>	161
6.35.1.8	<a href="#">enable_mag</a>	161
6.35.1.9	<a href="#">enable_sd_card</a>	161
6.35.1.10	<a href="#">erase_sd_card</a>	162
6.35.1.11	<a href="#">set_configuration_on_device</a>	162
6.36	<a href="#">apdm_ukf_state_t</a> Struct Reference	162
6.37	<a href="#">apdm_usb_device_list_t</a> Struct Reference	162
6.38	<a href="#">apdm_usb_sorted_device_element_t</a> Struct Reference	163
6.39	<a href="#">calibration_v4_t</a> Struct Reference	163
6.39.1	Member Data Documentation	167
6.39.1.1	<a href="#">accl_error_matrix</a>	167
6.39.1.2	<a href="#">accl_x_bias_temp</a>	167
6.39.1.3	<a href="#">accl_x_scale</a>	167
6.39.1.4	<a href="#">accl_x_scale_temp</a>	167
6.39.1.5	<a href="#">accl_xy_sensitivity</a>	167
6.39.1.6	<a href="#">accl_xz_sensitivity</a>	167
6.39.1.7	<a href="#">accl_y_bias</a>	167
6.39.1.8	<a href="#">accl_y_bias_temp</a>	167
6.39.1.9	<a href="#">accl_y_scale</a>	167
6.39.1.10	<a href="#">accl_y_scale_temp</a>	167
6.39.1.11	<a href="#">accl_yz_sensitivity</a>	167
6.39.1.12	<a href="#">accl_z_bias</a>	167
6.39.1.13	<a href="#">accl_z_bias_dtemp</a>	167
6.39.1.14	<a href="#">accl_z_bias_temp</a>	167
6.39.1.15	<a href="#">accl_z_scale</a>	167
6.39.1.16	<a href="#">accl_z_scale_temp</a>	167

6.39.1.17gyro_accl_pitch . . . . .	167
6.39.1.18gyro_accl_roll . . . . .	167
6.39.1.19gyro_accl_yaw . . . . .	167
6.39.1.20gyro_error_matrix . . . . .	167
6.39.1.21gyro_x_bias_temp . . . . .	167
6.39.1.22gyro_x_bias_temp2 . . . . .	167
6.39.1.23gyro_x_scale . . . . .	167
6.39.1.24gyro_x_scale_temp . . . . .	167
6.39.1.25gyro_xy_sensitivity . . . . .	167
6.39.1.26gyro_xz_sensitivity . . . . .	167
6.39.1.27gyro_y_bias . . . . .	167
6.39.1.28gyro_y_bias_temp . . . . .	167
6.39.1.29gyro_y_bias_temp2 . . . . .	167
6.39.1.30gyro_y_scale . . . . .	167
6.39.1.31gyro_y_scale_temp . . . . .	167
6.39.1.32gyro_yz_sensitivity . . . . .	167
6.39.1.33gyro_z_bias . . . . .	167
6.39.1.34gyro_z_bias_temp . . . . .	167
6.39.1.35gyro_z_scale . . . . .	167
6.39.1.36gyro_z_scale_temp . . . . .	167
6.39.1.37mag_x_scale . . . . .	167
6.39.1.38mag_y_bias . . . . .	167
6.39.1.39mag_y_state . . . . .	167
6.39.1.40mag_z_bias . . . . .	167
6.39.1.41mag_z_state . . . . .	167
6.39.1.42temperature_bias . . . . .	167
6.39.1.43temperature_bias_msp . . . . .	167
6.39.1.44temperature_scale . . . . .	167
6.39.1.45temperature_scale_msp . . . . .	167
6.40 calibration_v5_t Struct Reference . . . . .	168
6.40.1 Member Data Documentation . . . . .	169

6.40.1.1 accl_error_matrix . . . . .	169
6.40.1.2 accl_x_scale . . . . .	169
6.40.1.3 accl_x_scale_temp . . . . .	169
6.40.1.4 accl_xy_sensitivity . . . . .	169
6.40.1.5 accl_xz_sensitivity . . . . .	169
6.40.1.6 accl_y_bias . . . . .	169
6.40.1.7 accl_y_scale . . . . .	170
6.40.1.8 accl_y_scale_temp . . . . .	170
6.40.1.9 accl_yz_sensitivity . . . . .	170
6.40.1.10accl_z_bias . . . . .	170
6.40.1.11accl_z_bias_dtemp . . . . .	170
6.40.1.12accl_z_scale . . . . .	170
6.40.1.13accl_z_scale_temp . . . . .	170
6.40.1.14gyro_accl_pitch . . . . .	170
6.40.1.15gyro_accl_roll . . . . .	170
6.40.1.16gyro_accl_yaw . . . . .	170
6.40.1.17gyro_error_matrix . . . . .	170
6.40.1.18gyro_x_scale . . . . .	170
6.40.1.19gyro_x_scale_temp . . . . .	170
6.40.1.20gyro_xy_sensitivity . . . . .	170
6.40.1.21gyro_xz_sensitivity . . . . .	170
6.40.1.22gyro_y_bias . . . . .	170
6.40.1.23gyro_y_scale . . . . .	171
6.40.1.24gyro_y_scale_temp . . . . .	171
6.40.1.25gyro_yz_sensitivity . . . . .	171
6.40.1.26gyro_z_bias . . . . .	171
6.40.1.27gyro_z_scale . . . . .	171
6.40.1.28gyro_z_scale_temp . . . . .	171
6.40.1.29mag_x_scale . . . . .	171
6.40.1.30mag_y_bias . . . . .	171
6.40.1.31mag_y_state . . . . .	171

---

6.40.1.32mag_z_bias . . . . .	171
6.40.1.33mag_z_state . . . . .	171
6.40.1.34temperature_bias . . . . .	171
6.40.1.35temperature_bias_msp . . . . .	171
6.40.1.36temperature_scale . . . . .	171
6.40.1.37temperature_scale_msp . . . . .	171
6.41 per_device_info_t Struct Reference . . . . .	172
6.42 tekhex_t Struct Reference . . . . .	172
6.43 WIRELESS_PACKET Union Reference . . . . .	172
6.44 WP_CONFIG Struct Reference . . . . .	173
6.45 WP_CONFIG_ACK Struct Reference . . . . .	173
6.46 WP_DATA Struct Reference . . . . .	174
6.47 WP_EVENT Struct Reference . . . . .	174
6.48 WP_RAW Struct Reference . . . . .	175
6.49 WP_SYNC Struct Reference . . . . .	175

# Chapter 1

## Host Libraries

### Typical Configuration During Configuration

#### Typical Function Call Sequence for Auto-Configuring a System

1. [apdm\\_ctx\\_allocate\\_new\\_context](#)
2. [apdm\\_open\\_all\\_access\\_points](#)
3. [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint2](#)
4. [apdm\\_ctx\\_disconnect](#)
5. [apdm\\_ctx\\_free\\_context](#)

### Typical Configuration During Data Straming

#### Typical Function Call Sequence for Streaming Data From an Already Configured System

1. [apdm\\_ctx\\_allocate\\_new\\_context](#)
2. [apdm\\_open\\_all\\_access\\_points](#)
3. [apdm\\_set\\_max\\_sample\\_delay\\_ms](#)
4. [apdm\\_get\\_device\\_id\\_list](#)
5. [apdm\\_sync\\_record\\_list\\_head](#)
6. [apdm\\_get\\_next\\_access\\_point\\_record\\_list](#) (called many times in a loop)

7. `apdm_extract_data_by_device_id` (called to retrieve per-device data from the last list retrieved)
8. [`apdm\_ctx\_disconnect`](#)
9. [`apdm\_ctx\_free\_context`](#)

## Chapter 2

# Deprecated List

**Member [apdm\\_ap\\_set\\_max\\_latency\\_value](#)**(`apdm_ap_handle_t ap_handle`, `const uint32_t max_latency_ms`)

This has been replaced by [apdm\\_ap\\_set\\_max\\_latency\\_value\\_seconds\(\)](#). This function will be removed after Jan 2011.

**Member [apdm\\_ap\\_get\\_gpio\\_value](#)**(`apdm_ap_handle_t ap_handle`, `const apdm_ap_gpio_pin_t gpio_pin`, `bool`)

replaced with [apdm\\_ap\\_get\\_io\\_value\(\)](#) for more general purpose IO features

**Member [apdm\\_ap\\_set\\_gpio\\_value](#)**(`apdm_ap_handle_t ap_handle`, `const apdm_ap_gpio_pin_t gpio_pin`, `bool`)

replaced with [apdm\\_ap\\_set\\_io\\_value\(\)](#) for more general purpose IO features

**Member [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint](#)**(`apdm_ctx_t context`, `const uint8_t wireless_channel`)

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

**Member [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint2](#)**(`apdm_ctx_t context`, `const uint8_t wireless_channel`)

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

**Member [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint3](#)**(`apdm_ctx_t context`, `const uint8_t wireless_channel`)

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

Member [apdm\\_ctx\\_ap\\_get\\_gpio\\_value](#)([apdm\\_ctx\\_t](#) context, [const uint32\\_t](#) ap\_id, [const apdm\\_a](#)  
replaced with [apdm\\_ctx\\_ap\\_get\\_io\\_value\(\)](#) for more general IO

Member [apdm\\_ctx\\_ap\\_set\\_gpio\\_value](#)([apdm\\_ctx\\_t](#) context, [const uint32\\_t](#) ap\_id, [const apdm\\_a](#)  
replaced with [apdm\\_ctx\\_ap\\_get\\_io\\_value\(\)](#) for more general IO

Member [apdm\\_sensor\\_list\\_attached\\_sensors](#)([uint32\\_t](#) \*serial\_number\_buffer, [const uint32\\_t](#) bu  
non-standard function semantics, see [apdm\\_sensor\\_list\\_attached\\_  
sensors3\(\)](#). Will be removed after March 2011.



# Chapter 3

## Module Index

### 3.1 Modules

Here is a list of all modules:

Setup . . . . .	9
Context . . . . .	21
AccessPoint . . . . .	41
DataFiles . . . . .	56
Monitor . . . . .	66
MonitorCommands . . . . .	73
DockingStation . . . . .	104
DataHandling . . . . .	109
Logging . . . . .	115
Misc . . . . .	120



# Chapter 4

## Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">__attribute__</a>	127
<a href="#">apdm_access_point_configuration_t</a>	130
<a href="#">apdm_access_point_handle</a>	130
<a href="#">apdm_annotation_t</a>	132
<a href="#">apdm_bulk_in_buffer_t</a>	132
<a href="#">apdm_byte_array_ring_buffer_t</a>	132
<a href="#">apdm_case_id_t</a>	133
<a href="#">apdm_context_t</a>	133
<a href="#">apdm_device_dtemp_filter_state_t</a>	134
<a href="#">apdm_device_fifo_t</a>	135
<a href="#">apdm_device_info_t</a>	135
<a href="#">apdm_device_sample_buffer_row_t</a>	139
<a href="#">apdm_device_state_data_t</a>	140
<a href="#">apdm_device_status_t</a>	140
<a href="#">apdm_disk_ll_t</a>	141
<a href="#">apdm_external_sync_data_t</a>	141
<a href="#">apdm_mag_dechop_state_t</a>	142
<a href="#">apdm_mag_opt_data_t</a>	144
<a href="#">apdm_mag_step_response_state_t</a>	144
<a href="#">apdm_magnetometer_recalibration_t</a>	145
<a href="#">apdm_monitor_error_stat_t</a>	146
<a href="#">apdm_monitor_label_t</a>	146
<a href="#">apdm_orientation_info_t</a>	146
<a href="#">apdm_orientation_info_ukf_t</a>	147
<a href="#">apdm_orientation_ukf_fdata_t</a>	148

---

<a href="#">apdm_orientation_ukf_hdata_t</a>	148
<a href="#">apdm_progress_t</a>	149
<a href="#">apdm_record_ll_disk_data_t</a>	149
<a href="#">apdm_record_t</a>	149
<a href="#">apdm_recording_info_t</a>	156
<a href="#">apdm_sensor_cmd</a>	156
<a href="#">apdm_sensor_compensation_t</a>	157
<a href="#">apdm_sensor_device_handle_t</a>	157
<a href="#">apdm_sensor_response</a>	158
<a href="#">apdm_streaming_config_t</a>	160
<a href="#">apdm_ukf_state_t</a>	162
<a href="#">apdm_usb_device_list_t</a>	162
<a href="#">apdm_usb_sorted_device_element_t</a>	163
<a href="#">calibration_v4_t</a>	163
<a href="#">calibration_v5_t</a>	168
<a href="#">per_device_info_t</a>	172
<a href="#">tekhex_t</a>	172
<a href="#">WIRELESS_PACKET</a>	172
<a href="#">WP_CONFIG</a>	173
<a href="#">WP_CONFIG_ACK</a>	173
<a href="#">WP_DATA</a>	174
<a href="#">WP_EVENT</a>	174
<a href="#">WP_RAW</a>	175
<a href="#">WP_SYNC</a>	175

## Chapter 5

# Module Documentation

### 5.1 Setup

#### Functions

- APDM\_EXPORT int [apdm\\_ctx\\_set\\_correlation\\_fifo\\_temp\\_directory](#) (const char \*directory)
- APDM\_EXPORT int [apdm\\_calibration\\_override\\_minimum\\_supported\\_version](#) (const uint32\_t new\_version)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_ctx\\_ap\\_get\\_gpio\\_value](#) (apdm\_ctx\_t context, const uint32\_t ap\_id, const apdm\_ap\_gpio\_pin\_t gpio\_pin, bool \*output\_value)
- APDM\_EXPORT int [apdm\\_ctx\\_ap\\_get\\_io\\_value](#) (apdm\_ctx\_t context, const uint32\_t ap\_id, const apdm\_ap\_gpio\_pin\_t gpio\_pin, uint32\_t \*output\_value)
- APDM\_EXPORT int [apdm\\_ctx\\_ap\\_set\\_gpio\\_value](#) (apdm\_ctx\_t context, const uint32\_t ap\_id, const apdm\_ap\_gpio\_pin\_t gpio\_pin, const bool output\_value)
- APDM\_EXPORT int [apdm\\_ctx\\_ap\\_set\\_io\\_value](#) (apdm\_ctx\_t context, const uint32\_t ap\_id, const apdm\_ap\_gpio\_pin\_t gpio\_pin, const uint32\_t output\_value)
- APDM\_EXPORT int [apdm\\_ctx\\_open\\_all\\_access\\_points](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_init\\_access\\_point\\_wireless](#) (apdm\_ap\_handle\_t ap\_handle, const uint8\_t wireless\_channel\_1, const uint8\_t wireless\_channel\_2, const uint32\_t device\_rx\_address\_high\_order\_bytes\_A, const uint32\_t device\_rx\_address\_high\_order\_bytes\_B, const uint8\_t radio1\_pipe\_count, const uint8\_t radio2\_pipe\_count)
- APDM\_EXPORT int [apdm\\_exit](#) (void)

- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint2](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number, const bool enable\_sd\_card)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint3](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number, const bool enable\_sd\_card, const bool erase\_sd\_card)
- APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag)
- APDM\_EXPORT int [apdm\\_ctx\\_autoconfigure\\_devices\\_and\\_accesspoint5](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag, const apdm\_monitor\_decimation\_rate\_t decimation\_rate)
- APDM\_EXPORT int [apdm\\_init\\_streaming\\_config](#) (apdm\_streaming\_config\_t \*streaming\_config)
- APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint\\_streaming](#) (apdm\_ctx\_t context, apdm\_streaming\_config\_t \*streaming\_config)
- APDM\_EXPORT int [apdm\\_apply\\_autoconfigure\\_sensor\\_config](#) (apdm\_ctx\_t context, apdm\_device\_handle\_t ds\_handle)
- APDM\_EXPORT int [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint\\_wireless](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number)
- APDM\_EXPORT int [apdm\\_autoconfigure\\_mesh\\_sync](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag)
- APDM\_EXPORT int [apdm\\_autoconfigure\\_mesh\\_sync2](#) (apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number)
- APDM\_EXPORT int [apdm\\_configure\\_all\\_attached\\_sensors](#) (apdm\_ctx\_t context, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag)
- int [apdm\\_configure\\_all\\_attached\\_sensors\\_mesh](#) (apdm\_ctx\_t context, const uint32\_t wireless\_channel, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag)

### 5.1.1 Function Documentation

#### 5.1.1.1 **APDM\_EXPORT** int **apdm\_apply\_autoconfigure\_sensor\_config** ( apdm\_ctx\_t *context*, apdm\_device\_handle\_t *ds\_handle* )

When [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint\\_streaming\(\)](#) is called with `set_configuration_on_device` set to false, this function can be called after the fact to apply the configuration of the monitor to the respective monitor that is on the docking station.

##### Parameters

*context*

*ds\_handle* A docking handle, that has been opened, and has a monitor present in it.

##### Returns

APDM\_OK on success, error code otherwise.

References [apdm\\_ds\\_get\\_docked\\_module\\_id\(\)](#), [apdm\\_log\\_debug\(\)](#), and [apdm\\_log\\_error\(\)](#).

#### 5.1.1.2 **APDM\_DEPRECATED APDM\_EXPORT** int **apdm\_autoconfigure\_devices\_and\_accesspoint** ( apdm\_ctx\_t *context*, const uint8\_t *wireless\_channel\_number* )

##### Deprecated

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

##### See also

[apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#)

References [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#).

#### 5.1.1.3 **APDM\_DEPRECATED APDM\_EXPORT** int **apdm\_autoconfigure\_devices\_and\_accesspoint2** ( apdm\_ctx\_t *context*, const uint8\_t *wireless\_channel\_number*, const bool *enable\_sd\_card* )

##### Deprecated

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

**See also**

[apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#)

References [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#).

**5.1.1.4 APDM\_DEPRECATED APDM\_EXPORT int**  
**apdm\_autoconfigure\_devices\_and\_accesspoint3 ( apdm\_ctx\_t**  
**context, const uint8\_t wireless\_channel\_number, const bool**  
**enable\_sd\_card, const bool erase\_sd\_card )**

**Deprecated**

use [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), will be removed after March 2011

**See also**

[apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#)

References [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#).

**5.1.1.5 APDM\_EXPORT int apdm\_autoconfigure\_devices\_and\_ -**  
**accesspoint4 ( apdm\_ctx\_t context, const uint8\_t**  
**wireless\_channel\_number, const bool enable\_sd\_card, const**  
**bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const**  
**bool enable\_accel, const bool enable\_gyro, const bool**  
**enable\_mag )**

This function will automatically configure all attached access points and devices in such a way that data can be streamed from the the system.

**Parameters****context**

**wireless\_channel\_number** The base wireless channel to transmit data on, 0-100.

**enable\_sd\_card** Boolean indicating weather or not data should be logged to the SD card on the device.

**erase\_sd\_card** Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

**accel\_full\_scale\_mode** If true, then accelerometers will be in +/- 6g mode, if false, then they will be in +/- 2g mode

**enable\_accel** Enable the accelerometers



***enable\_gyro*** Enable the gyros

***enable\_mag*** Enable the magnetometers

### Returns

APDM\_OK if everything worked, something else if configuration failed.

Referenced by `apdm_autoconfigure_devices_and_accesspoint()`, `apdm_autoconfigure_devices_and_accesspoint2()`, and `apdm_autoconfigure_devices_and_accesspoint3()`.

#### 5.1.1.6 APDM\_EXPORT int apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming ( apdm\_ctx\_t context, apdm\_streaming\_config\_t \* streaming\_config )

Used to autoconfigure accesspoints and sensors based on contents of `apdm_streaming_config_t` data structure, replacement for the numeric variations of `apdm_autoconfigure_devices_and_accesspoint###()` functions.

### Parameters

***context***

***\*streaming\_config*** The configuration to be applied to the system.

### Returns

APDM\_OK on success, error code otherwise

References `apdm_ap_set_max_latency_value_seconds()`, `apdm_ap_set_error_blink_threshold()`, `apdm_ap_set_warning_blink_threshold()`, `apdm_ap_verify_supported_version()`, `apdm_configure_accesspoint()`, `apdm_ctx_set_max_sample_delay_seconds()`, `apdm_get_now_sync_value_host()`, `apdm_get_time_ms_64()`, `apdm_log_debug()`, `apdm_log_error()`, `apdm_log_info()`, `apdm_sensor_allocate_and_open()`, `apdm_sensor_close_and_free()`, `apdm_sensor_comm_channel_verify_supported_version()`, `apdm_strerror()`, and `apdm_device_info_t::wireless_channel1`.

#### 5.1.1.7 APDM\_EXPORT int apdm\_autoconfigure\_devices\_and\_accesspoint\_wireless ( apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number )

This function is similar to `apdm_autoconfigure_devices_and_accesspoint4()`, except that it doesn't override whatever device settings are already present on the attached devices.

### Parameters

***context***

***wireless\_channel\_number*** The base wireless channel to transmit data on, 0-100.

### Returns

APDM\_OK if everything worked, something else if configuration failed.

**5.1.1.8 APDM\_EXPORT int apdm\_autoconfigure\_mesh\_sync (**  
**apdm\_ctx\_t *context*, const uint8\_t *wireless\_channel\_number*,**  
**const bool *enable\_sd\_card*, const bool *erase\_sd\_card*, const**  
**bool *accel\_full\_scale\_mode*, const bool *enable\_accel*, const**  
**bool *enable\_gyro*, const bool *enable\_mag* )**

This function is used to configure all Motion Monitors currently attached to the host in synchronized logging mode, maximum of 32 devices.

### Parameters

***context***

***wireless\_channel\_number*** The wireless channel used to synchronize time between the motion monitors in the mesh time sync group.

***enable\_sd\_card*** Boolean indicating whether or not data should be logged to the SD card on the device.

***erase\_sd\_card*** Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

***accel\_full\_scale\_mode*** If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode

***enable\_accel*** Enable the accelerometers

***enable\_gyro*** Enable the gyros

***enable\_mag*** Enable the magnetometers

### Returns

APDM\_OK on success, error code otherwise

References apdm\_configure\_all\_attached\_sensors\_mesh(), and apdm\_log\_info().

#### 5.1.1.9 APDM\_EXPORT int apdm\_autoconfigure\_mesh\_sync2 ( apdm\_ctx\_t context, const uint8\_t wireless\_channel\_number )

Similar to [apdm\\_autoconfigure\\_mesh\\_sync\(\)](#), however this will configure all attached monitors into synchronized logging mode without modifying the pre-existing sensor settings on the monitors.

##### Parameters

**context**

**wireless\_channel\_number** The wireless channel used to synchronize time between the motion monitors in the mesh time sync group.

##### Returns

APDM\_OK on success, error code otherwise

References [apdm\\_log\\_info\(\)](#).

#### 5.1.1.10 APDM\_EXPORT int apdm\_calibration\_override\_minimum\_supported\_version ( const uint32\_t new\_version )

Allows you to override the minimum calibration version number used to validate calibration versions on motion sensors.

##### Parameters

**new\_version** Version number, e.g. 4 Set this to zero to use library default version number.

##### Returns

APDM\_OK on success, error code otherwise

#### 5.1.1.11 APDM\_EXPORT int apdm\_configure\_all\_attached\_sensors ( apdm\_ctx\_t context, const bool enable\_sd\_card, const bool erase\_sd\_card, const bool accel\_full\_scale\_mode, const bool enable\_accel, const bool enable\_gyro, const bool enable\_mag )

This function will automatically configure all attached access points and devices in such a way that data can be streamed from the the system.

**Parameters*****context***

***enable\_sd\_card*** Boolean indicating whether or not data should be logged to the SD card on the device.

***erase\_sd\_card*** Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

***accel\_full\_scale\_mode*** If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode

***enable\_accel*** Enable the accelerometers

***enable\_gyro*** Enable the gyros

***enable\_mag*** Enable the magnetometers

**Returns**

APDM\_OK if everything worked, something else if configuration failed.

References `apdm_streaming_config_t::accel_full_scale_mode`, `apdm_init_streaming_config()`, `apdm_streaming_config_t::enable_accel`, `apdm_streaming_config_t::enable_gyro`, `apdm_streaming_config_t::enable_mag`, `apdm_streaming_config_t::enable_sd_card`, and `apdm_streaming_config_t::erase_sd_card`.

**5.1.1.12** `int apdm_configure_all_attached_sensors_mesh ( apdm_ctx_t context, const uint32_t wireless_channel, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag )`

This function is used to configure all opals currently attached to the host in mesh time synchronization and data logging mode.

**Parameters*****context***

***wireless\_channel*** The wireless channel used to synchronize time between the opals in the mesh time sync group.

***enable\_sd\_card*** Boolean indicating whether or not data should be logged to the SD card on the device.

***erase\_sd\_card*** Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

***accel\_full\_scale\_mode*** If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode

***enable\_accel*** Enable the accelerometers

***enable\_gyro*** Enable the gyros

***enable\_mag*** Enable the magnetometers

Referenced by `apdm_autoconfigure_mesh_sync()`.

**5.1.1.13 APDM\_DEPRECATED APDM\_EXPORT int**  
**`apdm_ctx_ap_get_gpio_value ( apdm_ctx_t context, const`**  
**`uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, bool *`**  
**`output_value )`**

#### Parameters

***context*** The context of communications.

***ap\_id*** The ID number of the AP to manipulate the GPIO on.

***gpio\_pin*** The pin in question.

***\*output\_value*** Destination into which to store the current value of the GPIO pin.

#### Returns

APDM\_OK on success, error code otherwise.

#### Deprecated

replaced with [apdm\\_ctx\\_ap\\_get\\_io\\_value\(\)](#) for more general IO

References `apdm_ctx_ap_get_io_value()`.

**5.1.1.14 APDM\_EXPORT int**  
**`apdm_ctx_ap_get_io_value ( apdm_ctx_t`**  
**`context, const uint32_t ap_id, const apdm_ap_gpio_pin_t`**  
**`gpio_pin, uint32_t * output_value )`**

#### Parameters

***context*** The context of communications.

***ap\_id*** The ID number of the AP to manipulate the GPIO on.

***gpio\_pin*** The pin in question.

***\*output\_value*** Destination into which to store the current value of the GPIO pin.

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_ap_get_io_value()`, and `apdm_log_error()`.

Referenced by `apdm_ctx_ap_get_gpio_value()`.

**5.1.1.15** `APDM_EXPORT int apdm_ctx_ap_set_gpio_value ( apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, const bool output_value )`

#### Parameters

**context** The context of communications.

**ap\_id** The ID number of the AP to manipulate the GPIO on.

**gpio\_pin** The pin in question.

**output\_value** New value to set on a GPIO pin that has been configured as an output pin.

#### Returns

APDM\_OK on success, error code otherwise.

#### Deprecated

replaced with `apdm_ctx_ap_get_io_value()` for more general IO

References `apdm_ctx_ap_set_io_value()`.

**5.1.1.16** `APDM_EXPORT int apdm_ctx_ap_set_io_value ( apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, const uint32_t output_value )`

#### Parameters

**context** The context of communications.

**ap\_id** The ID number of the AP to manipulate the GPIO on.

**gpio\_pin** The pin in question.

**output\_value** New value to set on a GPIO pin that has been configured as an output pin.

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_ap_set_io_value()`, and `apdm_log_error()`.

Referenced by `apdm_ctx_ap_set_gpio_value()`.

**5.1.1.17** `APDM_EXPORT int apdm_ctx_autoconfigure_devices_and_accesspoint5 ( apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag, const apdm_monitor_decimation_rate_t decimation_rate )`

Same as [apdm\\_autoconfigure\\_devices\\_and\\_accesspoint4\(\)](#), but extra parameter allows you to set the decimation rate.

#### Returns

APDM\_OK if everything worked, something else if configuration failed.

**5.1.1.18** `APDM_EXPORT int apdm_ctx_open_all_access_points ( apdm_ctx_t context )`

Will cause all access points connected to the host to be opened and associated with the passed handle.

#### Parameters

**context** The handle for which to associate all opened access points.

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_ap_connect()`, `apdm_ap_disconnect()`, `apdm_ap_get_case_id()`, `apdm_ap_get_id_and_board_version()`, `apdm_ap_get_num_access_points_on_host1()`, `apdm_ctx_initialize_context()`, `apdm_get_time_ms_64()`, `apdm_log_context()`, `apdm_log_debug()`, `apdm_log_error()`, `apdm_log_info()`, and `apdm_strerror()`.

**5.1.1.19** `APDM_EXPORT int apdm_ctx_set_correlation_fifo_temp_directory ( const char * directory )`

Only relevant to windows. Sets the directory name into which correlation fifo temp files should be located.

#### Parameters

**\*directory** Directory into which fifo files should be placed, with trailing slash.

**Returns**

APDM\_OK on success, error code otherwise

**5.1.1.20 APDM\_EXPORT int apdm\_exit ( void )**

This function clears out any kernel event handlers or callbacks. Before unloading the DLL/SO/DYLIB and program termination, this function should be called.

**Returns**

APDM\_OK on success, error code otherwise

**5.1.1.21 APDM\_EXPORT int apdm\_init\_access\_point\_wireless ( apdm\_ap\_handle\_t *ap\_handle*, const uint8\_t *wireless\_channel\_1*, const uint8\_t *wireless\_channel\_2*, const uint32\_t *device\_rx\_address\_high\_order\_bytes\_A*, const uint32\_t *device\_rx\_address\_high\_order\_bytes\_B*, const uint8\_t *radio1\_pipe\_count*, const uint8\_t *radio2\_pipe\_count* )****Parameters**

***ap\_handle*** The access point handle to be configured

***wireless\_channel\_1*** Wireless channel number to use on the first radio

***wireless\_channel\_2*** Wireless channel number to use on the second radio

***device\_rx\_address\_high\_order\_bytes\_A*** The high-order block ID used for data filtering/matching by the radio (has bit-sequencing constraints for proper hardware behavior)

***deviceRXAddressHighOrderBytesB*** The high-order block ID used for data filtering/matching by the radio (has bit-sequencing constraints for proper hardware behavior)

***radio1\_pipe\_count*** Number of pipes to enable

***radio2\_pipe\_count*** Number of pipes to enable

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_configure\_accesspoint().



### 5.1.1.22 APDM\_EXPORT int apdm\_init\_streaming\_config ( apdm\_streaming\_config\_t \* *streaming\_config* )

Initializes streaming\_configuration data structure to default values.

wireless\_channel\_number = 80; enable\_sd\_card = true; erase\_sd\_card = false; accel\_full\_scale\_mode = true; enable\_accel = true; enable\_gyro = true; enable\_mag = true; apply\_new\_sensor\_modes = true; decimation\_rate = APDM\_DECIMATE\_5x2; output\_select\_rate = APDM\_OUTPUT\_SELECT\_RATE\_128; button\_enable = false;

#### Parameters

\****streaming\_config*** Pointer to [apdm\\_streaming\\_config\\_t](#) structure to be initialized

#### Returns

APDM\_OK on success, error code otherwise

References [apdm\\_streaming\\_config\\_t::accel\\_full\\_scale\\_mode](#), [apdm\\_streaming\\_config\\_t::apply\\_new\\_sensor\\_modes](#), [apdm\\_streaming\\_config\\_t::button\\_enable](#), [apdm\\_streaming\\_config\\_t::decimation\\_rate](#), [apdm\\_streaming\\_config\\_t::enable\\_accel](#), [apdm\\_streaming\\_config\\_t::enable\\_gyro](#), [apdm\\_streaming\\_config\\_t::enable\\_mag](#), [apdm\\_streaming\\_config\\_t::enable\\_sd\\_card](#), [apdm\\_streaming\\_config\\_t::erase\\_sd\\_card](#), and [apdm\\_streaming\\_config\\_t::set\\_configuration\\_on\\_device](#).

Referenced by [apdm\\_configure\\_all\\_attached\\_sensors\(\)](#).

## 5.2 Context

### Functions

- APDM\_EXPORT int [apdm\\_ctx\\_get\\_expected\\_number\\_of\\_sensors2](#) (apdm\_ctx\_t context, uint32\_t \*dest)
- APDM\_EXPORT enum APDM\_Status [apdm\\_ctx\\_set\\_error\\_handling\\_mode](#) (apdm\_ctx\_t context, enum APDMErrorHandlingBehavior new\_mode)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_sensor\\_compensation\\_data](#) (apdm\_ctx\_t context, [apdm\\_sensor\\_compensation\\_t](#) \*dest\_comp\_data, const int32\_t sensor\_index)
- APDM\_EXPORT int [apdm\\_ctx\\_set\\_sensor\\_compensation\\_data](#) (apdm\_ctx\_t context, const [apdm\\_sensor\\_compensation\\_t](#) \*src\_comp\_data, const int32\_t sensor\_index)

- APDM\_EXPORT int [apdm\\_ctx\\_get\\_expected\\_sync\\_delta](#) (apdm\_ctx\_t context, uint16\_t \*dest\_expected\_sync\_delta)
- APDM\_EXPORT int [apdm\\_ctx\\_set\\_metadeta\\_uint32](#) (apdm\_ctx\_t context, const uint32\_t device\_id, const uint32\_t value)
- APDM\_EXPORT int [apdm\\_ctx\\_set\\_metadata\\_string](#) (apdm\_ctx\_t context, const uint32\_t device\_id, const char \*str)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_metadata\\_uint32](#) (apdm\_ctx\_t context, const uint32\_t device\_id)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_wireless\\_configuration\\_mode](#) (apdm\_ctx\_t context, int \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_device\\_info](#) (apdm\_ctx\_t context, const uint32\_t device\_id, [apdm\\_device\\_info\\_t](#) \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_num\\_access\\_points\\_found](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_ap\\_id\\_for\\_ap\\_index](#) (apdm\_ctx\_t context, const int ap\_index, uint32\_t \*dest)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_num\\_sample\\_lists\\_collected](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_num\\_samples\\_collected](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_num\\_samples\\_collected\\_from\\_device](#) (apdm\_ctx\_t context, const uint32\_t device\_id)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_total\\_omitted\\_sample\\_sets](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_num\\_omitted\\_sample\\_sets](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_num\\_omitted\\_samples](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint32\_t [apdm\\_ctx\\_get\\_total\\_omitted\\_samples](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_sampling\\_frequency](#) (apdm\_ctx\_t context, uint32\_t \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_extract\\_data\\_by\\_device\\_id](#) (apdm\_ctx\_t context, const uint32\_t device\_id, [apdm\\_record\\_t](#) \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_next\\_access\\_point\\_record](#) (apdm\_ctx\_t context, [apdm\\_record\\_t](#) \*data, const int ap\_index\_number, const bool allowAPTransferFlag)
- APDM\_EXPORT int [apdm\\_ctx\\_sync\\_record\\_list\\_head](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_next\\_access\\_point\\_record\\_list](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_flush\\_ap\\_fifos](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_extract\\_next\\_sample](#) (apdm\_ctx\_t context, [apdm\\_record\\_t](#) \*dest\_record)

- APDM\_EXPORT int [apdm\\_ctx\\_get\\_next\\_synchronization\\_event](#) (apdm\_ctx\_t context, [apdm\\_external\\_sync\\_data\\_t](#) \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_next\\_record](#) (apdm\_ctx\_t context, [apdm\\_record\\_t](#) \*dest)
- APDM\_EXPORT int32\_t [apdm\\_ctx\\_get\\_device\\_id\\_by\\_index](#) (apdm\_ctx\_t context, const uint32\_t sensor\_index)
- APDM\_EXPORT int [apdm\\_ctx\\_set\\_requested\\_device\\_state](#) (apdm\_ctx\_t context, const enum RequestedDeviceState state, const int ap\_index\_number)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_device\\_index\\_by\\_id3](#) (apdm\_ctx\_t context, const uint32\_t id, uint32\_t \*dest\_index)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_device\\_id\\_list](#) (apdm\_ctx\_t context, uint32\_t \*dest, const uint32\_t destSize)
- APDM\_EXPORT int [apdm\\_ctx\\_initialize\\_context](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_disconnect](#) (apdm\_ctx\_t context)
- APDM\_EXPORT apdm\_ctx\_t [apdm\\_ctx\\_allocate\\_new\\_context](#) (void)
- APDM\_EXPORT int [apdm\\_ctx\\_free\\_context](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_disable\\_accesspoint\\_wireless](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_re\\_enable\\_accesspoint\\_wireless](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_is\\_more\\_data\\_immediately\\_available](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_ctx\\_avg\\_retry\\_count\\_for\\_device](#) (apdm\_ctx\_t context, const uint32\_t device\_id)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_wireless\\_reliability\\_value](#) (apdm\_ctx\_t context, const uint32\_t device\_id)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_wireless\\_streaming\\_status](#) (apdm\_ctx\_t context, uint32\_t \*dest)
- APDM\_EXPORT time\_t [apdm\\_ctx\\_get\\_last\\_received\\_timestamp\\_for\\_device](#) (apdm\_ctx\_t context, const uint32\_t device\_id)
- APDM\_EXPORT int [apdm\\_ctx\\_set\\_max\\_sample\\_delay\\_seconds](#) (apdm\_ctx\_t context, const uint16\_t max\_data\_delay\_seconds)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_monitor\\_latency](#) (apdm\_ctx\_t context, const uint32\_t monitor\_id, int64\_t \*dest)
- APDM\_EXPORT int [apdm\\_get\\_max\\_sample\\_delay\\_seconds](#) (apdm\_ctx\_t context, uint16\_t \*dest)
- APDM\_EXPORT int [apdm\\_ctx\\_reset\\_num\\_samples\\_from\\_ap](#) (apdm\_ctx\_t context)
- APDM\_EXPORT int [apdm\\_get\\_num\\_samples\\_from\\_ap](#) (apdm\_ctx\_t context)
- APDM\_EXPORT uint64\_t [apdm\\_ctx\\_estimate\\_now\\_sync\\_value](#) (apdm\_ctx\_t context)

## 5.2.1 Function Documentation

### 5.2.1.1 **APDM\_EXPORT** `apdm_ctx_t apdm_ctx_allocate_new_context ( void )`

Allocates memory a handle to be used by the apdm libraries.

#### Returns

Non-zero on success, zero otherwise

References `apdm_log_error()`, and `apdm_log_info()`.

### 5.2.1.2 **APDM\_EXPORT** `int apdm_ctx_avg_retry_count_for_device ( apdm_ctx_t context, const uint32_t device_id )`

Returns the average number of retries for samples coming from the given device, useful as a wireless reliability indicator for the device, (only accurate while actively streaming data thru the host libraries).

#### Parameters

*context*

*device\_id* The device ID

#### Returns

Negative error code on error, zero or higher with average number of retries per second for device id specified over previous 3 seconds.

Referenced by `apdm_ctx_get_wireless_reliability_value()`.

### 5.2.1.3 **APDM\_EXPORT** `int apdm_ctx_disable_accesspoint_wireless ( apdm_ctx_t context )`

This function will disable the wireless radios and protocol on all the access points in the context, causing them to no longer transmit sync packets, nor be able to RX data from monitors.

#### Parameters

*context*

#### Returns

APDM\_OK if everything worked, something else if configuration failed.

#### 5.2.1.4 APDM\_EXPORT int apdm\_ctx\_disconnect ( apdm\_ctx\_t context )

Disconnects from access points that are currently attached (USB bus handle disconnect)

##### Parameters

**context** The handle to be disconnected

##### Returns

APDM\_OK if successful, error code otherwise.

References apdm\_ap\_disconnect(), and apdm\_sensor\_close().

#### 5.2.1.5 APDM\_EXPORT uint64\_t apdm\_ctx\_estimate\_now\_sync\_value ( apdm\_ctx\_t context )

This function will estimate the current sync value of the system. This should be good to within about 50ms, and is dependant on the timing latency of the USB bus on the host and the clock drift rate delta between the AP and the host computer.

##### Parameters

**context** The context

##### Returns

An estimate of the current sync value.

References apdm\_get\_now\_sync\_value\_host().

Referenced by apdm\_ctx\_get\_wireless\_reliability\_value(), and apdm\_extract\_next\_sample\_set().

#### 5.2.1.6 APDM\_EXPORT int apdm\_ctx\_extract\_data\_by\_device\_id ( apdm\_ctx\_t context, const uint32\_t device\_id, apdm\_record\_t \* dest )

Gets data for a particular device id from the most record list.

##### Parameters

**context**

***device\_id*** The device id for which you want to retrieve data for

***\*dest*** The destination into which to put data.

### Returns

APDM\_OK on success, APDM\_NO\_MORE\_DATA if no more data, error code otherwise.

References apdm\_record\_t::device\_info\_serial\_number.

#### 5.2.1.7 APDM\_EXPORT int apdm\_ctx\_extract\_next\_sample ( apdm\_ctx\_t *context*, apdm\_record\_t \* *dest\_record* )

Extracts the next single sample from the set of AP's used in the context

### Parameters

***context*** The apdm handle

***\*dest\_record*** The record into which the sample data is to be stored.

### Returns

APDM\_OK on success, error code otherwise.

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), and apdm\_log\_error().

#### 5.2.1.8 APDM\_EXPORT int apdm\_ctx\_flush\_ap\_fifos ( apdm\_ctx\_t *context* )

This function is used to flush any data buffers or samples stored in RAM on the AP.

### Parameters

***context*** The apdm handle

### Returns

APDM\_OK on success, error code otherwise.

#### 5.2.1.9 APDM\_EXPORT int apdm\_ctx\_free\_context ( apdm\_ctx\_t context )

De-allocates memory used for the APDM handle context

##### Parameters

**context** The handle to be deallocated.

References apdm\_log\_error().

#### 5.2.1.10 APDM\_EXPORT int apdm\_ctx\_get\_ap\_id\_for\_ap\_index ( apdm\_ctx\_t context, const int ap\_index, uint32\_t \* dest )

##### Parameters

**context**

**ap\_index** The AP index number for which you want AP ID. This should be  $\geq 0$  and less than the number of APs configured.

**\*dest** Destination address into which the AP ID should be stored.

##### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

#### 5.2.1.11 APDM\_EXPORT int32\_t apdm\_ctx\_get\_device\_id\_by\_index ( apdm\_ctx\_t context, const uint32\_t sensor\_index )

##### Parameters

**context** The context for which you want the device id

**sensor\_index** The index of the device id for which you want

##### Returns

negative error code on error, zero if device is not found at the specified index, device ID  $> 0$  on success, only relevant after auto\_configure has been called.

**5.2.1.12** `APDM_EXPORT int apdm_ctx_get_device_id_list ( apdm_ctx_t context, uint32_t * dest, const uint32_t destSize )`

#### Parameters

- context** The context for which you want the device id
- \*dest** Destination array of uint32\_t's into which you want to store devices IDs, the (last+1) element will have a device id of zero.
- destSize** The number of elements in the destination array.

#### Returns

APDM\_OK On success, error code otherwise

**5.2.1.13** `APDM_EXPORT int apdm_ctx_get_device_index_by_id3 ( apdm_ctx_t context, const uint32_t id, uint32_t * dest_index )`

#### Parameters

- context** The context for which you want the device id
- id** The motion monitor ID for which you want the index of.
- \*dest\_index** The index of the specified device id

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_log_error()`.

Referenced by `apdm_ctx_get_device_info()`, `apdm_ctx_get_next_access_point_record()`, and `apdm_ctx_get_wireless_streaming_status()`.

**5.2.1.14** `APDM_EXPORT int apdm_ctx_get_device_info ( apdm_ctx_t context, const uint32_t device_id, apdm_device_info_t * dest )`

Gets device detailed information about the device id passed in.

#### Parameters

- context** The apdm handle
- device\_id** The ID of the device for which you'd like to get data
- \*dest** The destination structure into which you'd like to store the data



**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_ctx\_get\_device\_index\_by\_id3().

### 5.2.1.15 APDM\_EXPORT int apdm\_ctx\_get\_expected\_number\_of\_sensors2 ( apdm\_ctx\_t *context*, uint32\_t \* *dest* )

Returns the number of sensors that are configured in the context. This is with respect to an already-configured context. It is not necessarily the number of sensors attached to the system.

**Parameters**

***context*** The context of communications.

**Returns**

The number of sensors configured in the context.

Referenced by apdm\_ctx\_extract\_next\_sample(), apdm\_ctx\_get\_next\_record(), apdm\_ctx\_get\_num\_samples\_collected(), apdm\_ctx\_get\_num\_samples\_collected\_from\_device(), apdm\_ctx\_get\_sampling\_frequency(), apdm\_ctx\_get\_sensor\_compensation\_data(), apdm\_ctx\_get\_wireless\_streaming\_status(), apdm\_ctx\_is\_more\_data\_immediately\_available(), apdm\_ctx\_set\_sensor\_compensation\_data(), apdm\_ctx\_sync\_record\_list\_head(), and apdm\_extract\_next\_sample\_set().

### 5.2.1.16 APDM\_EXPORT int apdm\_ctx\_get\_expected\_sync\_delta ( apdm\_ctx\_t *context*, uint16\_t \* *dest\_expected\_sync\_delta* )

Depending on the output rate (e.g. 128 samples per second, 80 samples per second etc), this will return the expected sync delta between any two samples.

**Parameters**

***context*** The apdm context

**\**dest\_expected\_sync\_delta*** The destination into which to store the expected sync delta between any two samples

**Returns**

APDM\_OK on success

#### 5.2.1.17 **APDM\_EXPORT** time\_t apdm\_ctx\_get\_last\_received\_timestamp\_for\_device ( apdm\_ctx\_t *context*, const uint32\_t *device\_id* )

Gets the unix epoch time of when the last time a sample was received for the specified device ID. If you find that it's been a "long" time since a sample has been received from a device, you may check the device is powered and within range of an access point, (only accurate while actively streaming data thru the host libraries).

##### Parameters

*context*

*device\_id* The device ID

##### Returns

Zero on error or if no data has been received, non-zero with the epoch time otherwise.

#### 5.2.1.18 **APDM\_EXPORT** uint32\_t apdm\_ctx\_get\_metadata\_uint32 ( apdm\_ctx\_t *context*, const uint32\_t *device\_id* )

Allows for the retrieval of metadata for a device id.

##### Parameters

*context* The apdm handle

*device\_id* The device\_id for which you want metadata

##### Returns

The data associated with the device id, or zero if an error or no data having been set.

#### 5.2.1.19 **APDM\_EXPORT** int apdm\_ctx\_get\_monitor\_latency ( apdm\_ctx\_t *context*, const uint32\_t *monitor\_id*, int64\_t \* *dest* )

Retrieves the latency of an individual monitor from the given context.

##### Parameters

*context* The apdm context

***monitor\_id*** The ID of the monitor for which you want to know the latency.  
***\*dest*** The destination into which to store the latency value.

### Returns

APDM\_OK on success, error code otherwise.

References apdm\_ap\_get\_monitor\_latency(), and apdm\_log\_error().

**5.2.1.20 APDM\_EXPORT int apdm\_ctx\_get\_next\_access\_point\_record (**  
**apdm\_ctx\_t context, apdm\_record\_t \* data, const int**  
**ap\_index\_number, const bool allowAPTransferFlag )**

Gets the next record from the access point indicated

### Parameters

***context***

***\*data*** Destination into which to place data

***ap\_index\_number*** The index number of the AP on the host for which to retrieve data.

***allowAPTransferFlag*** Allow for the initiation of a new usb data transfer from the AP.

### Returns

APDM\_OK if data was retrieved, APDM\_NO\_MORE\_DATA if no more data, error code otherwise.

References apdm\_ctx\_get\_device\_index\_by\_id3(), apdm\_log\_debug(), apdm\_log\_error(), apdm\_log\_warning(), apdm\_strerror(), apdm\_device\_info\_t::decimation\_factor, apdm\_record\_t::device\_info\_serial\_number, apdm\_record\_t::num\_retrys, apdm\_record\_t::opt\_select, apdm\_device\_info\_t::protocol\_version, apdm\_record\_t::source\_ap\_index, and apdm\_record\_t::sync\_val32\_low.

**5.2.1.21 APDM\_EXPORT int apdm\_ctx\_get\_next\_access\_point\_record\_list ( apdm\_ctx\_t context**  
**)**

This function populates an list of records internal to the handle with a set of samples all corresponding to the same sync value (point in time). Depending on the error handling mode set, there may be some samples that are not populated or some partial sample sets that are skipped over.

### Parameters

***context*** The apdm handle

### Returns

APDM\_OK If it was able to get a sample set according to the error handling mode, APDM\_NO\_MORE\_DATA if there is no more data ready (just wait longer for more data to come in), or another code indicating what error occurred.

References apdm\_extract\_next\_sample\_set(), apdm\_log\_error(), and apdm\_strerror().

#### 5.2.1.22 APDM\_EXPORT int apdm\_ctx\_get\_next\_record ( apdm\_ctx\_t *context*, apdm\_record\_t \* *dest* )

This function will retrieve the oldest sample currently in the library buffers. In the case of multiple samples having the same age, it will return one of the oldest. This function will provide good realtime responsiveness to the caller, however, you may experience duplicates in the data stream or samples coming slightly out of order as samples are emitted as soon as it's available. This function takes a 2-5 milliseconds to execute, so avoid using it in tight data processing loops.

### Parameters

***context*** The apdm handle

***\*dest*** The record into which the data is stored.

### Returns

APDM\_OK upon success, APDM\_NO\_MORE\_DATA if no data is available, error code otherwise.

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), and apdm\_log\_error().

#### 5.2.1.23 APDM\_EXPORT int apdm\_ctx\_get\_next\_synchronization\_event ( apdm\_ctx\_t *context*, apdm\_external\_sync\_data\_t \* *dest* )

This function is used to gather external synchronization I/O data events. For GPIO inputs, input signals are debounced over a 1/2560 second period of time, and the sync value tagged on the synchronization sample will be that of the sync value of the time of the rising edge of the signal.

**Parameters**

- context*** The apdm handle
- \****dest*** Destination into which synchronization data is to be stored.

**Returns**

APDM\_OK if dest was populated with data, APDM\_NO\_MORE\_DATA if there is no synchronization event data available, error code otherwise.

**5.2.1.24 APDM\_EXPORT int apdm\_ctx\_get\_num\_access\_points\_found ( apdm\_ctx\_t *context* )****Parameters**

***context***

**Returns**

The number of access points attached to the host

**5.2.1.25 APDM\_EXPORT uint32\_t apdm\_ctx\_get\_num\_omitted\_sample\_sets ( apdm\_ctx\_t *context* )****Returns**

The number of omitted sample sets since the most recently requested sample set and the previously retrieved sample set.

**5.2.1.26 APDM\_EXPORT uint32\_t apdm\_ctx\_get\_num\_omitted\_samples ( apdm\_ctx\_t *context* )****Returns**

The number of omitted samples between the most recently requested sample set and the previously retrieved sample set.

**5.2.1.27** **APDM\_EXPORT** **uint32\_t** **apdm\_ctx\_get\_num\_**  
**sample\_lists\_collected** ( **apdm\_ctx\_t** *context*  
 )

#### Returns

The total number of sample lists collected since the handle was initialized.

**5.2.1.28** **APDM\_EXPORT** **uint32\_t** **apdm\_ctx\_get\_num\_**  
**samples\_collected** ( **apdm\_ctx\_t** *context*  
 )

#### Returns

The total number of samples collected since the handle was initialized.

References `apdm_ctx_get_expected_number_of_sensors2()`, `apdm_log_error()`, and `apdm_strerror()`.

**5.2.1.29** **APDM\_EXPORT** **uint32\_t** **apdm\_ctx\_get\_num\_samples\_**  
**collected\_from\_device** ( **apdm\_ctx\_t** *context*, **const** **uint32\_t**  
*device\_id* )

#### Returns

The total number of samples collected since the handle was initialized for the device id specified

References `apdm_ctx_get_expected_number_of_sensors2()`, `apdm_log_error()`, and `apdm_strerror()`.

**5.2.1.30** **APDM\_EXPORT** **int** **apdm\_ctx\_get\_sampling\_frequency** (  
**apdm\_ctx\_t** *context*, **uint32\_t** \* *dest* )

#### Parameters

*context*

\**dest* Destination into which to store the sampling frequency

#### Returns

Returns the sampling frequency that the devices are running at

References `apdm_ctx_get_expected_number_of_sensors2()`.

**5.2.1.31** **APDM\_EXPORT** int apdm\_ctx\_get\_sensor\_compensation\_data  
( apdm\_ctx\_t *context*, apdm\_sensor\_compensation\_t \*  
*dest\_comp\_data*, const int32\_t *sensor\_index* )

#### Parameters

***context*** The apdm context

**\**dest\_comp\_data*** The destination into which to store compensation data for the sensor of index *sensor\_index*.

***sensor\_index*** The index of the sensor for which you want to retrieve compensation data.

#### Returns

APDM\_OK on success

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), and apdm\_log\_error().

**5.2.1.32** **APDM\_EXPORT** uint32\_t apdm\_ctx\_get\_total\_omitted\_sample\_sets ( apdm\_ctx\_t *context* )

#### Returns

The number of omitted sample sets since the last time the context was initialized or since the last time apdm\_sync\_record\_head\_list() was called.

**5.2.1.33** **APDM\_EXPORT** uint32\_t apdm\_ctx\_get\_total\_omitted\_samples ( apdm\_ctx\_t *context* )

#### Returns

The number of omitted samples since the context was initialized, or since the last time apdm\_sync\_record\_head\_list() was called.

**5.2.1.34** **APDM\_EXPORT** int apdm\_ctx\_get\_wireless\_configuration\_mode ( apdm\_ctx\_t *context*, int \* *dest* )

#### Parameters

***context*** The apdm handle

\***dest** The destination into which to store the configured wireless mode.  
The value will be from the enum `apdm_wireless_mode_t`

### Returns

APDM\_OK on success, error code otherwise.

#### 5.2.1.35 APDM\_EXPORT int apdm\_ctx\_get\_wireless\_reliability\_value ( apdm\_ctx\_t context, const uint32\_t device\_id )

Used to get a number between 0 and 100 on how reliable the wireless connection is for a given device, (only accurate while actively streaming data thru the host libraries).

### Parameters

**context**

**device\_id** The device ID

### Returns

Negative error code on error, Zero to 100 on success, 100 being the best signal, zero being the worst (or no).

References `apdm_calculate_sync_value_age()`, `apdm_ctx_avg_retry_count_for_device()`, and `apdm_ctx_estimate_now_sync_value()`.

#### 5.2.1.36 APDM\_EXPORT int apdm\_ctx\_get\_wireless\_streaming\_status ( apdm\_ctx\_t context, uint32\_t \* dest )

### Parameters

**context**

\***dest** Destination into which to store the wireless streaming status by AP, of type `apdm_ap_wireless_streaming_status_t`.

### Returns

APDM\_OK on success, error code otherwise

References `apdm_ctx_get_device_index_by_id3()`, and `apdm_ctx_get_expected_number_of_sensors2()`.



**5.2.1.37 APDM\_EXPORT int apdm\_ctx\_initialize\_context ( apdm\_ctx\_t context )**

Used to initialize a handle context

**Parameters**

**context** The handle to be initialized

**Returns**

APDM\_OK if initialization was successful, error code otherwise.

References apdm\_ap\_init\_handle().

Referenced by apdm\_ctx\_open\_all\_access\_points().

**5.2.1.38 APDM\_EXPORT int apdm\_ctx\_is\_more\_data\_immediately\_available ( apdm\_ctx\_t context )**

Checks to see if more data is available in the host-resident sample buffers and if a subsequent call to get data would return without doing a USB bus transfer

**Parameters**

**context** The apdm context to check data on

**Returns**

Zero if there is no more data, non-zero if there is more data available.

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), and apdm\_log\_error().

**5.2.1.39 APDM\_EXPORT int apdm\_ctx\_re\_enable\_accesspoint\_wireless ( apdm\_ctx\_t context )**

After wireless has been disabled on an AP using the [apdm\\_ctx\\_disable\\_accesspoint\\_wireless\(\)](#) function, it can be re-enabled using this function

**Parameters**

**context**

**Returns**

APDM\_OK if everything worked, something else if configuration failed.

#### 5.2.1.40 **APDM\_EXPORT** int apdm\_ctx\_reset\_num\_samples\_from\_ap ( apdm\_ctx\_t *context* )

Used to reset the counter which tracks the number of samples received from the access point by the host libraries.

##### Parameters

***context*** The context

##### Returns

APDM\_OK on success, error code otherwise.

#### 5.2.1.41 **APDM\_EXPORT** enum APDM\_Status apdm\_ctx\_set\_error\_handling\_mode ( apdm\_ctx\_t *context*, enum APDMErrHandlingBehavior *new\_mode* )

Sets the error handling behavior of the underlying APDM libraries. Particularly affects when errors, partial records or full records are returned from a call to getting a record list.

##### Parameters

***context*** The apdm context

***newMode*** An enum APDMErrHandlingBehavior indicating what error handling mode to set.

##### Returns

APDM\_OK on success

#### 5.2.1.42 **APDM\_EXPORT** int apdm\_ctx\_set\_max\_sample\_delay\_seconds ( apdm\_ctx\_t *context*, const uint16\_t *max\_data\_delay\_seconds* )

This function sets the maximum amount of delay allowable for data returned from the host libraries. The default is 5ms. The max is 15 minutes.

##### Parameters

***context*** The apdm handle to check data on

***max\_data\_delay\_seconds*** The maximum age of returned packets from the library, set APDM\_INFINITE\_MAX\_LATENCY for infinity

**Returns**

APDM\_OK upon success, error code if failure.

References `apdm_ap_set_max_latency_value_seconds()`.

Referenced by `apdm_autoconfigure_devices_and_accesspoint_streaming()`.

**5.2.1.43** **APDM\_EXPORT** int `apdm_ctx_set_metadata_string (`  
    `apdm_ctx_t context, const uint32_t device_id, const char *`  
    `str )`

Metadata can be stored in the context with respect to a given device id, and later retrieved.

**Parameters**

**context** The apdm handle

**device\_id** The device ID for which you want the metadata string.

**str** The char\* type meta data to be stored, maximum length is USER\_META\_DATA\_STRING\_SIZE(64)

**Returns**

APDM\_OK on success

**5.2.1.44** **APDM\_EXPORT** int `apdm_ctx_set_metadeta_uint32 (`  
    `apdm_ctx_t context, const uint32_t device_id, const uint32_t`  
    `value )`

Metadata can be stored in the context with respect to a given device id, and later retrieved.

**Parameters**

**context** The apdm handle

**value** The uint32\_t type meta data to be stored. You can optionally associate meta-data with a device ID in a context. It's an arbitrary number for which the meaning is defined by the application using the library. E.G. You could use this to specify what limb of a persons body each motion monitor is attached to.

**Returns**

APDM\_OK on success

**5.2.1.45** **APDM\_EXPORT** int apdm\_ctx\_set\_sensor\_compensation\_data ( apdm\_ctx\_t *context*, const apdm\_sensor\_compensation\_t \* *src\_comp\_data*, const int32\_t *sensor\_index* )

#### Parameters

***context*** The apdm context

***\*src\_comp\_data*** The source of compensation data which to store into the context for the sensor of index *sensor\_index*.

***sensor\_index*** The index of the sensor for which you want to retrieve compensation data.

#### Returns

APDM\_OK on success

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), and apdm\_log\_error().

**5.2.1.46** **APDM\_EXPORT** int apdm\_ctx\_sync\_record\_list\_head ( apdm\_ctx\_t *context* )

This function will drop all data stored in the library correlation FIFOs and start reading data from the attached access points until it is able to get a full set of data from all sensors with the same sync value.

This function may return failure if one (or more) monitors is catching up it's data stream, you should wait until the AP's are blinking green. If one or more AP's is blinking green-red this function will likely fail.

It will also reset the total omitted samples counter.

#### Parameters

***context***

#### Returns

APDM\_OK if successful, APDM\_UNABLE\_TO\_SYNC\_RECORD\_HEAD\_LIST\_ERROR if it can't sync the list due to lack of data (usually because motion monitors are still docked, or out of range of the access point), error code otherwise

References apdm\_ctx\_get\_expected\_number\_of\_sensors2(), apdm\_get\_time\_ms\_64(), apdm\_log\_context(), apdm\_log\_debug(), apdm\_log\_error(), and apdm\_usleep().

#### 5.2.1.47 APDM\_EXPORT int apdm\_get\_max\_sample\_delay\_seconds ( apdm\_ctx\_t context, uint16\_t \* dest )

Gets the current max-sample-delay setting, in seconds (aka max latency)

##### Parameters

**context** The apdm handle to check data on

**\*dest** Destination into which the setting should be stored

##### Returns

APDM\_OK upon success, error code if failure.

#### 5.2.1.48 APDM\_EXPORT int apdm\_get\_num\_samples\_from\_ap ( apdm\_ctx\_t context )

Returns the number of sensor samples that have been transfered from all the attached AP's to the host. This number includes all samples that are currently in the correlation FIFO's of the library. It is useful if you want to verify data transfer from device->access point->host, but where configured library processing policys might be causing delays in data returned by the libraries.

##### Parameters

**context** The context

##### Returns

If zero or positive, the number of samples that have been transfered from the AP to the host libraries since the last time apdm\_reset\_num\_samples\_from\_ap was called, negative error code otherwise.

## 5.3 AccessPoint

### Functions

- APDM\_EXPORT int [apdm\\_ap\\_get\\_num\\_access\\_points\\_on\\_host1](#) (uint32\_t \*dest)
- APDM\_EXPORT int [apdm\\_ap\\_connect](#) (apdm\_ap\_handle\_t ap\_handle, const int indexNumber)
- APDM\_EXPORT int [apdm\\_ap\\_disconnect](#) (apdm\_ap\_handle\_t ap\_handle)

- APDM\_EXPORT int [apdm\\_ap\\_init\\_handle](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_ap\\_set\\_max\\_latency\\_value](#) (apdm\_ap\_handle\_t ap\_handle, const uint32\_t max\_latency\_ms)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_monitor\\_latency](#) (apdm\_ap\_handle\_t ap\_handle, const uint32\_t monitor\_id, int64\_t \*dest)
- APDM\_EXPORT int [apdm\\_ap\\_set\\_warning\\_blink\\_threshold](#) (apdm\_ap\_handle\_t ap\_handle, const uint32\_t delta\_threshold)
- APDM\_EXPORT int [apdm\\_ap\\_set\\_error\\_blink\\_threshold](#) (apdm\_ap\_handle\_t ap\_handle, const uint32\_t delta\_threshold)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_wireless\\_streaming\\_led\\_status](#) (apdm\_ap\_handle\_t ap\_handle, uint32\_t \*dest)
- APDM\_EXPORT const char \* [apdm\\_ap\\_wireless\\_streaming\\_status\\_t\\_str](#) (const apdm\_ap\_wireless\_streaming\_status\_t streaming\_status)
- APDM\_EXPORT int [apdm\\_ap\\_set\\_max\\_latency\\_value\\_seconds](#) (apdm\_ap\_handle\_t ap\_handle, const uint16\_t max\_latency\_seconds)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_ap\\_get\\_gpio\\_value](#) (apdm\_ap\_handle\_t ap\_handle, const apdm\_ap\_gpio\_pin\_t gpio\_pin, bool \*output\_value)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_io\\_value](#) (apdm\_ap\_handle\_t ap\_handle, const apdm\_ap\_gpio\_pin\_t gpio\_pin, uint32\_t \*output\_value)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_ap\\_set\\_gpio\\_value](#) (apdm\_ap\_handle\_t ap\_handle, const apdm\_ap\_gpio\_pin\_t gpio\_pin, const bool output\_value)
- APDM\_EXPORT int [apdm\\_ap\\_set\\_io\\_value](#) (apdm\_ap\_handle\_t ap\_handle, const apdm\_ap\_gpio\_pin\_t gpio\_pin, const uint32\_t output\_value)
- APDM\_EXPORT int [apdm\\_send\\_accesspoint\\_cmd](#) (apdm\_ap\_handle\_t ap\_handle, const char \*cmdToSend, char \*BYTE\_ARRAY, const uint32\_t outputBufferLength, const uint32\_t numLinesToRead, const uint32\_t timeoutMilliseconds)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_version\\_string](#) (apdm\_ap\_handle\_t ap\_handle, char \*BYTE\_ARRAY, const int destLength)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_version](#) (apdm\_ap\_handle\_t ap\_handle, uint64\_t \*dest)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_board\\_version\\_string](#) (apdm\_ap\_handle\_t ap\_handle, char \*BYTE\_ARRAY, const int destLength)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_id\\_and\\_board\\_version](#) (apdm\_ap\_handle\_t ap\_handle, uint32\_t \*dest\_id, uint32\_t \*dest\_board\_version)
- APDM\_EXPORT int [apdm\\_ap\\_verify\\_supported\\_version](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_EXPORT int [apdm\\_ap\\_override\\_minimum\\_supported\\_version](#) (const uint64\_t new\_version)

- APDM\_EXPORT int [apdm\\_ap\\_get\\_id](#) (apdm\_ap\_handle\_t ap\_handle, uint32\_t \*dest)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_case\\_id](#) (apdm\_ap\_handle\_t ap\_handle, char \*BYTE\_ARRAY, const int dest\_buffer\_length)
- APDM\_EXPORT int [apdm\\_ap\\_reset\\_into\\_bootloader](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_EXPORT int [apdm\\_ap\\_reset\\_into\\_firmware](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_EXPORT int [apdm\\_free\\_ap\\_handle](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_EXPORT apdm\_ap\_handle\_t [apdm\\_ap\\_allocate\\_handle](#) (void)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_mode](#) (apdm\_ap\_handle\_t ap\_handle)
- APDM\_EXPORT int [apdm\\_ap\\_get\\_protocol\\_subversion](#) (apdm\_ap\_handle\_t ap\_handle, int64\_t \*dest\_protocol\_subversion)
- APDM\_EXPORT int [apdm\\_configure\\_accesspoint](#) (apdm\_ap\_handle\_t ap\_handle, const uint8\_t radio1\_pipe\_count, const uint8\_t radio2\_pipe\_count)
- APDM\_EXPORT int [apdm\\_ctx\\_get\\_all\\_ap\\_debug\\_info](#) (apdm\_ctx\_t context)

### 5.3.1 Function Documentation

#### 5.3.1.1 APDM\_DEPRECATED APDM\_EXPORT int apdm\_ap\_set\_max\_latency\_value ( apdm\_ap\_handle\_t ap\_handle, const uint32\_t max\_latency\_ms )

Sets the maximum latency of packets that should be coming from devices to the access point. If set to zero, greater than  $(1000 * 65535 * 24 / 128) = 12287812.5\text{ms}$  then not latency constraint will be applied by the device. Default is 15,000ms, max is 60,000ms.

#### Deprecated

This has been replaced by [apdm\\_ap\\_set\\_max\\_latency\\_value\\_seconds\(\)](#). This function will be removed after Jan 2011.

#### Parameters

**ap\_handle** The AP handle for which this value is to be set.

**max\_latency\_ms** The maximum delay, in mS, which a device should send buffered packets to the AP.

#### Returns

APDM\_OK on success, error code otherwise.

References `adpm_ap_set_max_latency_value_seconds()`.

#### 5.3.1.2 **APDM\_EXPORT** int `adpm_ap_set_max_latency_value_seconds` ( `adpm_ap_handle_t` *ap\_handle*, const uint16\_t *max\_latency\_seconds* )

Sets the maximum latency of packets that should be coming from devices to the access point.

##### Parameters

***ap\_handle*** The AP handle for which this value is to be set.

***max\_latency\_seconds*** The maximum delay, in seconds, which a device should send buffered packets to the AP. A value of `APDM_INFINITE_MAX_LATENCY` implies infinity.

##### Returns

`APDM_OK` on success, error code otherwise.

Referenced by `adpm_ap_set_max_latency_value()`, `adpm_autoconfigure_devices_and_accesspoint_streaming()`, and `adpm_ctx_set_max_sample_delay_seconds()`.

#### 5.3.1.3 **APDM\_EXPORT** `adpm_ap_handle_t` `adpm_ap_allocate_handle` ( void )

Allocates memory for an access point handle.

##### Returns

NULL on failure, non-NULL on success.

#### 5.3.1.4 **APDM\_EXPORT** int `adpm_ap_connect` ( `adpm_ap_handle_t` *ap\_handle*, const int *indexNumber* )

Used to connect to an access point.

##### Parameters

***ap\_handle*** An un-configured access point handle.

***indexNumber*** The index number, starting at zero, of the AP on the host to which to connect.



**Returns**

APDM\_OK on success, other on error.

References `apdm_ap_get_protocol_subversion()`, `apdm_ap_get_version()`, `apdm_ap_get_version_string()`, `apdm_get_time_ms_64()`, `apdm_log_debug()`, `apdm_log_warning()`, and `apdm_strerror()`.

Referenced by `apdm_ctx_open_all_access_points()`.

**5.3.1.5 APDM\_EXPORT int apdm\_ap\_disconnect ( apdm\_ap\_handle\_t ap\_handle )**

Disconnects the access point handle from the underlying OS binding

**Parameters**

**ap\_handle** The handle to be disconnected.

**Returns**

APDM\_OK if successful

Referenced by `apdm_ctx_disconnect()`, and `apdm_ctx_open_all_access_points()`.

**5.3.1.6 APDM\_EXPORT int apdm\_ap\_get\_board\_version\_string ( apdm\_ap\_handle\_t ap\_handle, char \* BYTE\_ARRAY, const int destLength )**

Returns the board hardware version string from the access point.

**Parameters**

**ap\_handle** The handle with respect to what version string you want.

**\*BYTE\_ARRAY** The destination buffer into which you want the version string copied into, must not be NULL.

**destLength** The maximum length of the destination string, must be >0

**Returns**

APDM\_OK on success.

References `apdm_send_accesspoint_cmd()`.

**5.3.1.7 APDM\_EXPORT** int apdm\_ap\_get\_case\_id ( apdm\_ap\_handle\_t *ap\_handle*, char \* *BYTE\_ARRAY*, const int *dest\_buffer\_length* )

Retrieves the case ID of the AP.

#### Parameters

***ap\_handle*** The AP handle

***\*BYTE\_ARRAY*** The destination buffer into which to store the case ID string.

***dest\_buffer\_length*** The max length of the destination buffer

#### Returns

APDM\_OK on success, error code otherwise

References apdm\_log\_error(), and apdm\_log\_warning().

Referenced by apdm\_ctx\_open\_all\_access\_points().

**5.3.1.8 APDM\_DEPRECATED APDM\_EXPORT** int apdm\_ap\_get\_gpio\_value ( apdm\_ap\_handle\_t *ap\_handle*, const apdm\_ap\_gpio\_pin\_t *gpio\_pin*, bool \* *output\_value* )

#### Parameters

***ap\_handle*** The AP handle.

***gpio\_pin*** The pin in question.

***\*output\_value*** Destination into which to store the current value of the GPIO pin.

#### Returns

APDM\_OK on success, error code otherwise.

#### Deprecated

replaced with [apdm\\_ap\\_get\\_io\\_value\(\)](#) for more general purpose IO features

References apdm\_ap\_get\_io\_value().

#### 5.3.1.9 APDM\_EXPORT int apdm\_ap\_get\_id ( apdm\_ap\_handle\_t ap\_handle, uint32\_t \* dest )

This returns the serial number of the access point

##### Parameters

**ap\_handle** The AP handle associated with the AP for which you want the serial number.

**\*dest** The destination into which the serial number is to be stored.

##### Returns

APDM\_OK on success, error code otherwise

References apdm\_ap\_get\_id\_and\_board\_version(), apdm\_log\_debug(), and apdm\_send\_accesspoint\_cmd().

#### 5.3.1.10 APDM\_EXPORT int apdm\_ap\_get\_id\_and\_board\_version ( apdm\_ap\_handle\_t ap\_handle, uint32\_t \* dest\_id, uint32\_t \* dest\_board\_version )

##### Parameters

**\*ap\_handle** The access point handle

**\*dest\_id** The destination into which to store the ID of the access point

**\*dest\_board\_version** The destination into which to store the printed circuit board version

##### Returns

APDM\_OK on success, error code otherwise

Referenced by apdm\_ap\_get\_id(), and apdm\_ctx\_open\_all\_access\_points().

#### 5.3.1.11 APDM\_EXPORT int apdm\_ap\_get\_io\_value ( apdm\_ap\_handle\_t ap\_handle, const apdm\_ap\_gpio\_pin\_t gpio\_pin, uint32\_t \* output\_value )

##### Parameters

**ap\_handle** The AP handle.

**gpio\_pin** The pin in question.

**\*output\_value** Destination into which to store the current value of the GPIO pin.

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

Referenced by apdm\_ap\_get\_gpio\_value(), and apdm\_ctx\_ap\_get\_io\_value().

**5.3.1.12 APDM\_EXPORT int apdm\_ap\_get\_mode ( apdm\_ap\_handle\_t  
ap\_handle )****Parameters**

**\*ap\_handle** A handle that is already connected to an AP via usb.

**Returns**

APM\_FIRMWARE if the AP is in firmware mode, APM\_BOOTLOADER if it's in bootloader, APM\_UNKNOWN if the mode cannot be determined,

References apdm\_log\_error(), and apdm\_send\_accesspoint\_cmd().

**5.3.1.13 APDM\_EXPORT int apdm\_ap\_get\_monitor\_latency ( apdm\_ap\_handle\_t ap\_handle, const uint32\_t monitor\_id, int64\_t \* dest )**

Retrieves the latency of an individual monitor from the given AP.

**Parameters**

**ap\_handle** The ap handle which is to be queried.

**monitor\_id** The ID of the monitor for which you want to know the latency.

**\*dest** The destination into which to store the latency value.

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

Referenced by apdm\_ctx\_get\_monitor\_latency().

**5.3.1.14** **APDM\_EXPORT** int apdm\_ap\_get\_num\_  
access\_points\_on\_host1 ( uint32\_t \* *dest*  
)

#### Parameters

\****dest*** Destination into which to store the number of USB access points attached to the host based on the VID/PID listing from the OS.

#### Returns

APDM\_OK on success, error code otherwise

Referenced by apdm\_ctx\_open\_all\_access\_points().

**5.3.1.15** **APDM\_EXPORT** int apdm\_ap\_get\_protocol\_subversion  
( apdm\_ap\_handle\_t *ap\_handle*, int64\_t \*  
*dest\_protocol\_subversion* )

#### Parameters

***ap\_handle*** The AP handle

\****dest\_protocol\_subversion*** The destination into which to store the protocol version

#### Returns

APDM\_OK on success, error code otherwise.

Referenced by apdm\_ap\_connect().

**5.3.1.16** **APDM\_EXPORT** int apdm\_ap\_get\_version (   
apdm\_ap\_handle\_t *ap\_handle*, uint64\_t \* *dest* )

#### Parameters

***ap\_handle*** The access point handle for which you want the numeric representation of the version

\****dest*** The destination into which to store the ap firmware version number

#### Returns

APDM\_OK on success.

Referenced by apdm\_ap\_connect(), and apdm\_ap\_verify\_supported\_version().

**5.3.1.17** **APDM\_EXPORT** int apdm\_ap\_get\_version\_string ( apdm\_ap\_handle\_t *ap\_handle*, char \* *BYTE\_ARRAY*, const int *destLength* )

Returns the firmware version string from the access point.

#### Parameters

- ap\_handle*** The handle with respect to what version string you want.
- \**BYTE\_ARRAY*** The destination buffer into which you want the version string copied into, must not be NULL.
- destLength*** The maximum length of the destination string, must be greater then zero.

#### Returns

APDM\_OK on success.

References apdm\_get\_time\_ms\_64(), apdm\_log\_debug(), and apdm\_send\_accesspoint\_cmd().

Referenced by apdm\_ap\_connect(), and apdm\_ap\_verify\_supported\_version().

**5.3.1.18** **APDM\_EXPORT** int apdm\_ap\_get\_wireless\_streaming\_led\_status ( apdm\_ap\_handle\_t *ap\_handle*, uint32\_t \* *dest* )

#### Parameters

- ap\_handle*** The AP handle
- \**dest*** The destination into which to store the LED status, of type apdm\_ap\_wireless\_streaming\_status\_t

#### Returns

APDM\_OK on success, error code otherwise

**5.3.1.19** **APDM\_EXPORT** int apdm\_ap\_init\_handle ( apdm\_ap\_handle\_t *ap\_handle* )

Initializes an access point handle

#### Parameters

- ap\_handle*** Pointer to the handle to be initialized

**Returns**

APDM\_OK if successful.

Referenced by apdm\_ctx\_initialize\_context().

**5.3.1.20 APDM\_EXPORT int apdm\_ap\_override\_minimum\_supported\_version ( const uint64\_t *new\_version* )**

Allows you to override the minimum access point station version number used to validate AP versions.

**Parameters**

***new\_version*** Version number, e.g. 20100902170629 Set this to zero to use library default version number.

**Returns**

APDM\_OK on success, error code otherwise

**5.3.1.21 APDM\_EXPORT int apdm\_ap\_reset\_into\_bootloader ( apdm\_ap\_handle\_t *ap\_handle* )**

This function will reset the given access point into bootloader

**Parameters**

***\*ap\_handle*** A handle that is already connected to an AP via usb.

**Returns**

APDM\_OK on success, and if successful, the handle will have been DISCONNECTED and you must re-connect to the AP.

**5.3.1.22 APDM\_EXPORT int apdm\_ap\_reset\_into\_firmware ( apdm\_ap\_handle\_t *ap\_handle* )**

This function will reset the given access point into firmware

**Parameters**

***\*ap\_handle*** A handle that is already connected to an AP via usb.

**Returns**

APDM\_OK on success, and if successful, the handle will have been DISCONNECTED and you must re-connect to the AP.

**5.3.1.23 APDM\_EXPORT int apdm\_ap\_set\_error\_blink\_threshold**  
( **apdm\_ap\_handle\_t** *ap\_handle*, **const uint32\_t**  
*delta\_threshold* )

**Parameters**

***ap\_handle*** The access point handle

***delta\_threshold*** The threshold, in milliseconds, for the AP to start blinking green/red if one (or more) monitors are falling behind in their transmission (e.g. out of range).

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming().

**5.3.1.24 APDM\_DEPRECATED APDM\_EXPORT int**  
**apdm\_ap\_set\_gpio\_value** ( **apdm\_ap\_handle\_t** *ap\_handle*,  
**const apdm\_ap\_gpio\_pin\_t** *gpio\_pin*, **const bool**  
*output\_value* )

**Parameters**

***ap\_handle*** The AP handle.

***gpio\_pin*** The pin in question.

***output\_value*** New value to set on a GPIO pin that has been configured as an output pin.

**Returns**

APDM\_OK on success, error code otherwise.

**Deprecated**

replaced with [apdm\\_ap\\_set\\_io\\_value\(\)](#) for more general purpose IO features

References [apdm\\_ap\\_set\\_io\\_value\(\)](#).



**5.3.1.25** **APDM\_EXPORT** int apdm\_ap\_set\_io\_value (  
apdm\_ap\_handle\_t *ap\_handle*, const apdm\_ap\_gpio\_pin\_t  
*gpio\_pin*, const uint32\_t *output\_value* )

#### Parameters

***ap\_handle*** The AP handle.

***gpio\_pin*** The pin in question.

***output\_value*** New value to set on a GPIO pin that has been configured as an output pin.

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

Referenced by apdm\_ap\_set\_gpio\_value(), and apdm\_ctx\_ap\_set\_io\_value().

**5.3.1.26** **APDM\_EXPORT** int apdm\_ap\_set\_warning\_blink\_threshold (  
apdm\_ap\_handle\_t *ap\_handle*, const uint32\_t  
*delta\_threshold* )

#### Parameters

***ap\_handle*** The access point handle

***delta\_threshold*** The threshold, in milliseconds, for the AP to start blinking green/blue if one (or more) monitors are falling behind in their transmission (e.g. out of range).

#### Returns

APDM\_OK on success, error code otherwise.

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming().

**5.3.1.27** **APDM\_EXPORT** int apdm\_ap\_verify\_supported\_version (  
apdm\_ap\_handle\_t *ap\_handle* )

this function is used to verify that the given access point has a version of firmware that is supported by the libraries.

#### Parameters

***ap\_handle*** The access point handle

**Returns**

APDM\_OK if the version is OK, respective error code otherwise.

References `apdm_ap_get_version()`, `apdm_ap_get_version_string()`, and `apdm_log_error()`.

Referenced by `apdm_autoconfigure_devices_and_accesspoint_streaming()`.

### 5.3.1.28 **APDM\_EXPORT** `const char* apdm_ap_wireless_streaming_status_t_str ( const apdm_ap_wireless_streaming_status_t streaming_status )`

**Parameters**

***streaming\_status*** Streaming status, of type `apdm_ap_wireless_streaming_status_t`, for which you want the string representation.

**Returns**

Pointer to a string for the given status type

### 5.3.1.29 **APDM\_EXPORT** `int apdm_configure_accesspoint ( apdm_ap_handle_t ap_handle, const uint8_t radio1_pipe_count, const uint8_t radio2_pipe_count )`

Internal function to configure a single access point with a given pipe count and assign wireless channels based on whats configured in the AP handle data structure.

**Parameters**

***ap\_handle*** The handle for the AP to be configured.

***radio1\_pipe\_count*** The number of pipes that will be used for radio 1 of the AP (first three devices usually)

***radio2\_pipe\_count*** The number of pipes that will be used for radio 2 of the AP (first three devices usually)

**Returns**

APDM\_OK on success, error code otherwise.

References `apdm_init_access_point_wireless()`, `apdm_log_error()`, and `apdm_log_info()`.

Referenced by `apdm_autoconfigure_devices_and_accesspoint_streaming()`.

#### 5.3.1.30 APDM\_EXPORT int apdm\_ctx\_get\_all\_ap\_debug\_info ( apdm\_ctx\_t context )

The access point tracks some internal debugging stats and numbers. This function will retrieve those debugging statistic and print to the debug logging subsystem.

##### Parameters

**context**

##### Returns

APDM\_OK if successful, error code otherwise

#### 5.3.1.31 APDM\_EXPORT int apdm\_free\_ap\_handle ( apdm\_ap\_handle\_t ap\_handle )

Frees memory for the given access point handle

##### Parameters

**ap\_handle** The handle to be freed

##### Returns

APDM\_OK on success.

References apdm\_log\_warning().

#### 5.3.1.32 APDM\_EXPORT int apdm\_send\_accesspoint\_cmd ( apdm\_ap\_handle\_t ap\_handle, const char \* cmdToSend, char \* BYTE\_ARRAY, const uint32\_t outputBufferLength, const uint32\_t numLinesToRead, const uint32\_t timeoutMilliseconds )

Sends a string-command to the access point, mostly used for debugging and non-standard functionality.

##### Parameters

**ap\_handle** The handle for the AP to which the command is to be sent

**\*cmdToSend** The command to send

**\*BYTE\_ARRAY** The destination into which the response from the AP is to be placed.

***outputBufferLength*** The length of the outputStringBuffer.

***numLinesToRead*** The number of lines that are expected in response to the command being sent.

***timeoutMilliseconds*** Maximum time length to wait for the response.

### Returns

APDM\_OK on success.

References `apdm_get_time_ms_64()`, `apdm_log_debug()`, and `apdm_log_error()`.

Referenced by `apdm_ap_get_board_version_string()`, `apdm_ap_get_id()`, `apdm_ap_get_mode()`, and `apdm_ap_get_version_string()`.

## 5.4 DataFiles

### Functions

- APDM\_EXPORT int [apdm\\_read\\_raw\\_file\\_info](#) (const char \*filename, [apdm\\_recording\\_info\\_t](#) \*recording\_info)
- APDM\_EXPORT int [apdm\\_process\\_raw](#) (char \*\*file\_in, char \*\*calibration\_file, int nFiles, const char \*file\_out, const bool store\_raw, const bool store\_si, const bool format\_hdf, const bool compress, char csv\_delimiter, [apdm\\_progress\\_t](#) \*progress)
- APDM\_EXPORT hid\_t [apdm\\_create\\_file\\_hdf](#) (const char \*filename, [apdm\\_device\\_info\\_t](#) \*device\_info, int nMonitors)
- APDM\_EXPORT int [apdm\\_close\\_file\\_hdf](#) (hid\_t file)
- APDM\_EXPORT [apdm\\_csv\\_t](#) [apdm\\_create\\_file\\_csv](#) (char \*filename)
- APDM\_EXPORT int [apdm\\_close\\_file\\_csv](#) ([apdm\\_csv\\_t](#) file\_handle)
- APDM\_EXPORT int [apdm\\_write\\_record\\_hdf](#) (hid\_t file, [apdm\\_device\\_info\\_t](#) \*info, [apdm\\_record\\_t](#) \*records, int sampleNumber, int nDevices, bool store\_raw, bool store\_si, bool compress)
- APDM\_EXPORT int [apdm\\_write\\_record\\_csv](#) ([apdm\\_csv\\_t](#) file, [apdm\\_device\\_info\\_t](#) \*info, [apdm\\_record\\_t](#) \*records, int sampleNumber, int nDevices, bool store\_raw, bool store\_si, char delimiter)
- APDM\_EXPORT int [apdm\\_write\\_annotation](#) (hid\_t file, [apdm\\_annotation\\_t](#) \*annotation)
- APDM\_EXPORT int [apdm\\_read\\_hdf\\_dataset](#) (const char \*file, char \*monitor\_id, const char \*datasetName, double \*data, int ndims, const int \*start\_index, const int \*shape, const int \*strideLength)
- APDM\_EXPORT int [apdm\\_read\\_hdf\\_timestamps](#) (char \*file, char \*monitor\_id, char \*datasetName, [uint64\\_t](#) \*data, int start\_index, int nSamples, int strideLength)

- APDM\_EXPORT int [apdm\\_get\\_hdf\\_dataset\\_shape](#) (char \*file, char \*monitor\_id, char \*datasetName, int \*shape, int \*ndims)
- APDM\_EXPORT int [apdm\\_read\\_hdf\\_calibration\\_data](#) (char \*file, char \*case\_id, [apdm\\_sensor\\_compensation\\_t](#) \*sensor\_comp)
- APDM\_EXPORT int [apdm\\_get\\_hdf\\_device\\_list](#) (char \*file, char \*\*monitor\_ids, int \*nDevices)
- APDM\_EXPORT int [apdm\\_get\\_hdf\\_device\\_list\\_swig](#) (char \*file, [apdm\\_case\\_id\\_t](#) \*monitor\_ids, int \*nDevices)
- APDM\_EXPORT int [apdm\\_get\\_hdf\\_label\\_list](#) (char \*file, char \*\*monitor\_labels, int \*nDevices)
- APDM\_EXPORT int [apdm\\_get\\_hdf\\_label\\_list\\_swig](#) (char \*file, [apdm\\_monitor\\_label\\_t](#) \*monitor\_labels, int \*nDevices)

### 5.4.1 Function Documentation

#### 5.4.1.1 APDM\_EXPORT int apdm\_close\_file\_csv ( [apdm\\_csv\\_t file\\_handle](#) )

Close a file opened with [apdm\\_create\\_file\\_csv](#).

##### Parameters

***file\_handle*** The file handle returned from [apdm\\_create\\_file\\_csv](#)

##### Returns

APDM\_OK on success

#### 5.4.1.2 APDM\_EXPORT int apdm\_close\_file\_hdf ( [hid\\_t file](#) )

Closes a file previously opened with [apdm\\_create\\_file\\_hdf](#).

##### Parameters

***file*** The HDF5 file handle returned from [apdm\\_create\\_file\\_hd](#)

##### Returns

APDM\_OK on success

#### 5.4.1.3 APDM\_EXPORT apdm\_csv\_t apdm\_create\_file\_csv ( char \* *filename* )

Creates a new file and returns a file pointer.

##### Parameters

*filename* Name of the file to create

##### Returns

apdm\_csv\_t (actually a FILE \*) file handle for the new file

#### 5.4.1.4 APDM\_EXPORT hid\_t apdm\_create\_file\_hdf ( const char \* *filename*, apdm\_device\_info\_t \* *device\_info*, int *nMonitors* )

Creates a new APDM HDF5 file and opens it for writing. Used with apdm\_write\_record\_hdf, apdm\_write\_annotation, and apdm\_close\_file\_hdf to stream to a data file.

##### Parameters

*filename* The filename to create. Should have the ".h5" extension.

*device\_info* The configuration information for each monitor. Used to write various metadata.

*nMonitors* The number of monitors in the device\_info array.

##### Returns

hid\_t HDF5 file handle, always > 0 on success.

#### 5.4.1.5 APDM\_EXPORT int apdm\_get\_hdf\_dataset\_shape ( char \* *file*, char \* *monitor\_id*, char \* *datasetName*, int \* *shape*, int \* *ndims* )

Helper function for working with HDF5 files. Gets the size of a specified dataset.

##### Parameters

*file* The .h5 file to inspect.

*monitor\_id* The group name for the monitor (returned by apdm\_get\_hdf\_device\_list). For v1 files, this is the 'Opal\_xx' where xx is the monitor id. For v2 files, this is the case id.

**datasetName** The name of the dataset to inspect. Must be "Accelerometers", "Gyroscopes", "Magnetometers", "Temperature", )

**shape** Output array containing the size in each dimension of the dataset. If shape is NULL, then only ndims will be set.

**ndims** Output containing the number of dimensions in the dataset.

### Returns

APDM\_OK on success

Referenced by apdm\_recalibrate\_gyroscopes\_from\_h5().

#### 5.4.1.6 APDM\_EXPORT int apdm\_get\_hdf\_device\_list ( char \* file, char \*\* monitor\_ids, int \* nDevices )

Helper function for working with HDF5 files. Gets the list of monitor IDs stored in the file.

### Parameters

**file** The .h5 file to inspect.

**\*\*monitor\_ids** Output array containing the group name for each monitor in the file. If NULL, only nDevices will be set.

**\*nDevices** Output number of monitors in the file.

### Returns

APDM\_OK on success

Referenced by apdm\_get\_hdf\_device\_list\_swig(), and apdm\_recalibrate\_gyroscopes\_from\_h5().

#### 5.4.1.7 APDM\_EXPORT int apdm\_get\_hdf\_device\_list\_swig ( char \* file, apdm\_case\_id\_t \* monitor\_ids, int \* nDevices )

Helper function for working with HDF5 files that is compatible with SWIG based Java bindings. Gets the list of monitor IDs stored in the file.

### Parameters

**file** The .h5 file to inspect.

**\*monitor\_ids** Output array containing the group name for each monitor in the file. If NULL, only nDevices will be set.

\***nDevices** Output number of monitors in the file.

### Returns

APDM\_OK on success

References `apdm_get_hdf_device_list()`.

#### 5.4.1.8 APDM\_EXPORT int apdm\_get\_hdf\_label\_list ( char \* *file*, char \*\* *monitor\_labels*, int \* *nDevices* )

Helper function for working with HDF5 files. Gets the list of monitor labels stored in the file. This list is in the same order as the list returned by [apdm\\_get\\_hdf\\_device\\_list\(\)](#).

### Parameters

**file** The .h5 file to inspect.

**monitor\_labels** Output array containing the user specified label for each monitor in the file. If NULL, only nDevices will be set.

**nDevices** Output number of monitors in the file.

### Returns

APDM\_OK on success

Referenced by `apdm_get_hdf_label_list_swig()`.

#### 5.4.1.9 APDM\_EXPORT int apdm\_get\_hdf\_label\_list\_swig ( char \* *file*, apdm\_monitor\_label\_t \* *monitor\_labels*, int \* *nDevices* )

Helper function for working with HDF5 files that is compatible with SWIG based Java bindings. Gets the list of monitor labels stored in the file. This list is in the same order as the list returned by [apdm\\_get\\_hdf\\_device\\_list\(\)](#).

### Parameters

**file** The .h5 file to inspect.

**monitor\_labels** Output array containing the user specified label for each monitor in the file. If NULL, only nDevices will be set.

**nDevices** Output number of monitors in the file.

### Returns

APDM\_OK on success

References `apdm_get_hdf_label_list()`.



**5.4.1.10** `APDM_EXPORT int apdm_process_raw ( char ** file_in, char ** calibration_file, int nFiles, const char * file_out, const bool store_raw, const bool store_si, const bool format_hdf, const bool compress, char csv_delimiter, apdm_progress_t * progress )`

Reads a .apdm file, and writes either raw, calibrated, or both, to either a .csv or a .h5 file.

#### Parameters

***file\_in*** An array of .apdm input file names from an unique monitors.

***calibration\_file*** array of optional files containing .hex calibration data for the monitors. If NULL, the calibration data in the .apdm file is used.

***nFiles*** The number of input files present in *file\_in*.

***file\_out*** The output filename (should end in .csv or .h5). If NULL, .csv format is forced and output is written to stdout.

***store\_raw*** If true, raw data is stored.

***store\_si*** If true, calibrated data (in SI units) is stored.

***format\_hdf*** If true, *file\_out* will be written to in HDF5 format. If false, *file\_out* will be written to in .csv format. If multiple input files are present, HDF output must be selected.

***csv\_delimiter*** Column delimiter to use if *format\_hdf* is false (writing in csv format).

***progress*** If not null, this is an [apdm\\_progress\\_t](#) structure allocated by the caller and modified as this function runs.

#### Returns

APDM\_OK on success, error code otherwise

**5.4.1.11** `APDM_EXPORT int apdm_read_hdf_calibration_data ( char * file, char * case_id, apdm_sensor_compensation_t * sensor_comp )`

Helper function for working with HDF5 files. Gets the calibration data from a file and converts it to an [apdm\\_sensor\\_compensation\\_t](#) struct.

#### Parameters

***file*** The .h5 file to read.

***case\_id*** The Case ID string of the monitor to get the calibration data for.

**sensor\_comp** [apdm\\_sensor\\_compensation\\_t](#) structure to be populated with the calibration data.

### Returns

APDM\_OK on success

References [apdm\\_log\\_debug\(\)](#).

Referenced by [apdm\\_recalibrate\\_gyroscopes\\_from\\_h5\(\)](#).

**5.4.1.12** **APDM\_EXPORT** int [apdm\\_read\\_hdf\\_dataset](#) ( const char \* *file*, char \* *monitor\_id*, const char \* *datasetName*, double \* *data*, int *ndims*, const int \* *start\_index*, const int \* *shape*, const int \* *strideLength* )

Helper function for working with HDF5 files. Loads a segment of data from one of the datasets in one of the monitors stored in the .h5 file.

### Parameters

**file** The .h5 file to load data from

**monitor\_id** The group name for the monitor (returned by [apdm\\_get\\_hdf\\_device\\_list](#)). For v1 files, this is the 'Opal\_xx' where xx is the monitor id. For v2 files, this is the case id.

**datasetName** The name of the dataset to load. Must be ("Accelerometers", "Gyroscopes", "Magnetometers", or "Temperature");

**data** Output array to load data into. This array is "flattened" so that the nth sample of the mth channel is at location `data[n+m*N]` where N is the total number of samples for each channel in the array.

**ndims** Number of dimensions in the dataset to be read. Should always be 2.

**start\_index** Initial index to start reading from. First element is sample number, second element is channel number.

**shape** Size of the output array (data). First element is the number of samples, second is the number of channels.

**strideLength** Number of data points to skip by in each dimension.

### Returns

APDM\_OK on success

References [apdm\\_log\\_debug\(\)](#).

Referenced by [apdm\\_recalibrate\\_gyroscopes\\_from\\_h5\(\)](#).

#### 5.4.1.13 APDM\_EXPORT int apdm\_read\_hdf\_timestamps ( char \* *file*, char \* *monitor\_id*, char \* *datasetName*, uint64\_t \* *data*, int *start\_index*, int *nSamples*, int *strideLength* )

Helper function for working with HDF5 files. Loads a segment of data from one of the time datasets in one of the monitors stored in the .h5 file.

##### Parameters

***file*** The .h5 file to load data from

***monitor\_id*** The group name for the monitor (returned by apdm\_get\_hdf\_device\_list). For v1 files, this is the 'Opal\_xx' where xx is the monitor id. For v2 files, this is the case id.

***datasetName*** The name of the dataset to load. Must be "Time" or "Sync-Value"

***data*** Output array to load data into.

***start\_index*** Initial index to start reading from

***nSamples*** Size of the output array (data).

***strideLength*** Number of data points to skip by.

##### Returns

APDM\_OK on success

References apdm\_log\_debug().

#### 5.4.1.14 APDM\_EXPORT int apdm\_read\_raw\_file\_info ( const char \* *filename*, apdm\_recording\_info\_t \* *recording\_info* )

Reads metadata from a .apdm file and uses it to populate a [apdm\\_recording\\_info\\_t](#) structure.

##### Parameters

***filename*** The filename of the .apdm file

**\**recording\_info*** A pointer to the file\_info structure to populate with meta-data.

##### Returns

APDM\_OK on success, error code otherwise

References apdm\_log\_debug(), and apdm\_log\_error().

#### 5.4.1.15 **APDM\_EXPORT** int apdm\_write\_annotation ( hid\_t *file*, apdm\_annotation\_t \* *annotation* )

Adds an annotation to the HDF5 file.

##### Parameters

**file** The HDF5 file handle returned by apdm\_create\_file\_hdf

**annotation** An [apdm\\_annotation\\_t](#) struct containing a monitor ID, timestamp (epoch microseconds), and string (max 2048 characters).

##### Returns

APDM\_OK on success

Referenced by apdm\_write\_record\_hdf().

#### 5.4.1.16 **APDM\_EXPORT** int apdm\_write\_record\_csv ( apdm\_csv\_t *file*, apdm\_device\_info\_t \* *info*, apdm\_record\_t \* *records*, int *sampleNumber*, int *nDevices*, bool *store\_raw*, bool *store\_si*, char *delimiter* )

Write an array of records to a CSV file (previously created with apdm\_create\_file\_csv). The sample number must be tracked by the caller, and incremented for each new sample.

##### Parameters

**file** The file handle returned from apdm\_create\_file\_csv

**info** Array of [apdm\\_device\\_info\\_t](#) structs containing information about each monitor.

**records** Array of [apdm\\_record\\_t](#) structs containing the data for one sample set of each monitor.

**sampleNumber** It is important that the first time this function is called sampleNumber is 0. This can not be used as a row index. One row is created every time this function is called, regardless of the value of sampleNumber. The meta data written in the first column depends on sampleNumber.

**nDevices** Number of monitors in the info and records arrays.

**store\_raw** Flag indicating whether raw data should be stored. (True: yes, False: no)

**store\_si** Flag indicating whether SI data should be stored. (True: yes, False: no)

**delimiter** Delimiter to use to separate columns.

### Returns

APDM\_OK on success

**5.4.1.17 APDM\_EXPORT** `int apdm_write_record_hdf ( hid_t file, apdm_device_info_t * info, apdm_record_t * records, int sampleNumber, int nDevices, bool store_raw, bool store_si, bool compress )`

Write an array of records to a HDF5 file (previously created with `apdm_create_file_hdf`). The sample number must be tracked by the caller, and incremented for each new sample. In case of dropped data, it may be desired to increment the `sampleNumber` for each dropped sample.

### Parameters

- file** The HDF5 file handle returned from `apdm_create_file_hdf`
- info** Array of `apdm_device_info_t` structs containing information about each monitor.
- records** Array of `apdm_record_t` structs containing the data for one sample set of each monitor.
- sampleNumber** An index into the arrays stored in the HDF5 file. It is important that the first time this function is called `sampleNumber` is 0.
- nDevices** Number of monitors in the `info` and `records` arrays.
- store\_raw** Flag indicating whether raw data should be stored. (True: yes, False: no)
- store\_si** Flag indicating whether SI data should be stored. (True: yes, False: no)
- compress** Flag indicating whether data should be compressed. This is almost always a good idea, but some old versions of Matlab (<2008b) have been found to have difficulty reading compressed data. (True: yes, False: no)

### Returns

APDM\_OK on success

References `apdm_record_t::accl_x_axis`, `apdm_record_t::accl_y_axis`, `apdm_record_t::accl_y_axis_si`, `apdm_record_t::accl_z_axis`, `apdm_record_t::accl_z_axis_si`, `apdm_log_debug()`, `apdm_write_annotation()`,

apdm\_record\_t::gyro\_x\_axis, apdm\_record\_t::gyro\_x\_axis\_si, apdm\_record\_t::gyro\_y\_axis, apdm\_record\_t::gyro\_y\_axis\_si, apdm\_record\_t::gyro\_z\_axis, apdm\_record\_t::gyro\_z\_axis\_si, apdm\_record\_t::mag\_common\_axis, apdm\_record\_t::mag\_x\_axis, apdm\_record\_t::mag\_x\_axis\_si, apdm\_record\_t::mag\_y\_axis, apdm\_record\_t::mag\_z\_axis, and apdm\_record\_t::temperature\_derivative\_si.

## 5.5 Monitor

### Functions

- APDM\_EXPORT apdm\_device\_handle\_t [apdm\\_sensor\\_allocate\\_and\\_open](#) (const uint32\_t sensor\_index)
- APDM\_EXPORT void [apdm\\_sensor\\_free\\_handle](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT apdm\_device\_handle\_t [apdm\\_sensor\\_allocate\\_handle](#) (void)
- APDM\_EXPORT int [apdm\\_sensor\\_open](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t device\_index)
- APDM\_EXPORT int [apdm\\_sensor\\_list\\_attached\\_sensors3](#) (uint32\_t \*serial\_number\_buffer, const uint32\_t buffer\_length, uint32\_t \*dest\_count)
- APDM\_DEPRECATED APDM\_EXPORT int [apdm\\_sensor\\_list\\_attached\\_sensors](#) (uint32\_t \*serial\_number\_buffer, const uint32\_t buffer\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_close](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_verify\\_supported\\_calibration\\_version](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_verify\\_supported\\_version](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_override\\_minimum\\_supported\\_version](#) (const char \*new\_version)
- APDM\_EXPORT int [apdm\\_sensor\\_comm\\_channel\\_verify\\_supported\\_version](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_initialize\\_device\\_info](#) (apdm\_device\_info\_t \*device\_info)
- APDM\_EXPORT int [apdm\\_sensor\\_apply\\_configuration](#) (apdm\_device\_handle\_t device\_handle, apdm\_device\_info\_t \*device\_info)
- APDM\_EXPORT int [apdm\\_device\\_extract\\_module\\_id\\_from\\_case\\_id\\_string](#) (const char \*case\_id, uint32\_t \*dest\_module\_id)
- APDM\_EXPORT int [apdm\\_sensor\\_get\\_device\\_id\\_list](#) (uint32\_t \*serial\_number\_buffer, const uint32\_t buffer\_length)

- APDM\_EXPORT int [apdm\\_halt\\_all\\_attached\\_sensors](#) (void)
- APDM\_EXPORT int **apdm\_sensor\_configure\_wireless** (apdm\_device\_handle\_t device\_handle, const enum APDMDeviceConfig wirelessConfigType, const uint32\_t value)
- APDM\_EXPORT int [apdm\\_sensor\\_populate\\_device\\_info](#) (apdm\_device\_handle\_t device\_handle, [apdm\\_device\\_info\\_t](#) \*dest)

## 5.5.1 Function Documentation

### 5.5.1.1 APDM\_EXPORT int apdm\_device\_extract\_module\_id - from\_case\_id\_string ( const char \* *case\_id*, uint32\_t \* *dest\_module\_id* )

Extracts the module ID from a monitor case ID string, such as "SI-000025" will find 25.

#### Parameters

- \****case\_id*** Case ID string from the motion monitor
- \****dest\_module\_id*** Destination into which to store the module ID

#### Returns

APDM\_OK on success, error code otherwise

### 5.5.1.2 APDM\_EXPORT int apdm\_halt\_all\_attached\_sensors ( void )

Halts all sensors that are connected to the host. Note, make sure all device handles and contexts have been closed prior to calling this function.

#### Returns

Zero on success, non-zero error code if failure.

References [apdm\\_log\\_debug\(\)](#), [apdm\\_log\\_error\(\)](#), [apdm\\_log\\_info\(\)](#), [apdm\\_sensor\\_allocate\\_handle\(\)](#), [apdm\\_sensor\\_close\(\)](#), [apdm\\_sensor\\_cmd\\_halt\(\)](#), [apdm\\_sensor\\_free\\_handle\(\)](#), [apdm\\_sensor\\_get\\_num\\_attached\\_dockingstations1\(\)](#), [apdm\\_sensor\\_open\(\)](#), and [apdm\\_strerror\(\)](#).

### 5.5.1.3 APDM\_EXPORT int apdm\_initialize\_device\_info ( apdm\_device\_info\_t \* *device\_info* )

Applies default configuration settings to a [apdm\\_device\\_info\\_t](#) structure

**Parameters**

\**device\_info* Pointer to the structure to be initialized

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_monitor\_decimation\_rate\_t\_to\_int(), apdm\_monitor\_output\_select\_rate\_t\_to\_int(), and apdm\_device\_info\_t::decimation\_factor.

#### 5.5.1.4 APDM\_EXPORT apdm\_device\_handle\_t apdm\_sensor\_allocate\_and\_open ( const uint32\_t *sensor\_index* )

Allocates and opens/connects to a sensor on the system

**Parameters**

*sensor\_index* The index of the sensor attached to the host

**Returns**

Zero handle on error, non-zero handle on success.

References apdm\_sensor\_allocate\_handle(), apdm\_sensor\_free\_handle(), and apdm\_sensor\_open().

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming().

#### 5.5.1.5 APDM\_EXPORT apdm\_device\_handle\_t apdm\_sensor\_allocate\_handle ( void )

Allocates memory and returns a new sensor handle.

**Returns**

Zero on failure, non-zero handle on success.

References apdm\_log\_debug(), apdm\_log\_error(), and apdm\_sensor\_free\_handle().

Referenced by apdm\_halt\_all\_attached\_sensors(), and apdm\_sensor\_allocate\_and\_open().



**5.5.1.6** **APDM\_EXPORT** int apdm\_sensor\_apply\_configuration (  
apdm\_device\_handle\_t *device\_handle*, apdm\_device\_info\_t \*  
*device\_info* )

**Parameters**

*device\_handle* The handle to the device to apply the configuration to.

*device\_info* The configuration to be applied to the device.

**Returns**

APDM\_OK on success, error code otherwise.

**5.5.1.7** **APDM\_EXPORT** int apdm\_sensor\_close (  
apdm\_device\_handle\_t *device\_handle* )

Closes the handle.

**Parameters**

*device\_handle* The handle to be closed

**Returns**

APDM\_OK on success.

Referenced by apdm\_ctx\_disconnect(), apdm\_halt\_all\_attached\_sensors(),  
apdm\_sensor\_close\_and\_free(), and apdm\_sensor\_get\_device\_id\_list().

**5.5.1.8** **APDM\_EXPORT** int apdm\_sensor\_comm\_channel\_verify\_  
supported\_version ( apdm\_device\_handle\_t *device\_handle*  
)

this function is used to verify that the given docking station handle has a version  
of firmware that is supported by the libraries.

**Parameters**

*device\_handle* The device handle

**Returns**

APDM\_OK if the version is OK, respective error code otherwise.

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming().

#### 5.5.1.9 APDM\_EXPORT void apdm\_sensor\_free\_handle ( apdm\_device\_handle\_t *device\_handle* )

Frees memory associated with a device handle

##### Parameters

***device\_handle*** The handle to be freed.

References apdm\_log\_debug().

Referenced by apdm\_halt\_all\_attached\_sensors(), apdm\_sensor\_allocate\_and\_open(), apdm\_sensor\_allocate\_handle(), and apdm\_sensor\_close\_and\_free().

#### 5.5.1.10 APDM\_EXPORT int apdm\_sensor\_get\_device\_id\_list ( uint32\_t \* *serial\_number\_buffer*, const uint32\_t *buffer\_length* )

Retrieves a list of device ID's attached to the host.

##### Parameters

***\*serial\_number\_buffer*** Pointer to an array of uint32\_t into which the serial numbers should be stored.

***buffer\_length*** The number of elements in the destination array.

##### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_info(), apdm\_sensor\_close(), apdm\_sensor\_cmd\_device\_id(), apdm\_sensor\_open(), and apdm\_strerror().

#### 5.5.1.11 APDM\_DEPRECATED APDM\_EXPORT int apdm\_sensor\_list\_attached\_sensors ( uint32\_t \* *serial\_number\_buffer*, const uint32\_t *buffer\_length* )

Fills the buffer pointed to by serial\_number\_buffer with a list of Motion Monitor ID numbers.

##### Deprecated

non-standard function semantics, see [apdm\\_sensor\\_list\\_attached\\_sensors3\(\)](#). Will be removed after March 2011.

**Parameters**

\****serial\_number\_buffer*** Destination array into which device IDs are to be populated

***buffer\_length*** The maximum number of entries in the serial\_number\_ - buffer buffer.

**Returns**

APDM\_OK on success.

References apdm\_sensor\_list\_attached\_sensors3().

**5.5.1.12 APDM\_EXPORT int apdm\_sensor\_list\_attached\_sensors3**  
( uint32\_t \* ***serial\_number\_buffer***, const uint32\_t  
***buffer\_length***, uint32\_t \* ***dest\_count*** )

Fills the buffer pointed to by serial\_number\_buffer with a list of Motion Monitor ID numbers.

**Parameters**

\****serial\_number\_buffer*** Destination array into which device IDs are to be populated

***buffer\_length*** The maximum number of entries in the serial\_number\_ - buffer buffer.

\****dest\_count*** The number of devices added to the buffer list

**Returns**

APDM\_OK on success.

Referenced by apdm\_sensor\_list\_attached\_sensors().

**5.5.1.13 APDM\_EXPORT int apdm\_sensor\_open (**  
**apdm\_device\_handle\_t *device\_handle*, const uint32\_t**  
***device\_index* )**

Opens a sensor by it's corresponding index number on the host computer.

**Parameters**

***device\_handle*** The handle to which the corresponding sensor is to be associated with

***device\_index*** The index of the device which is to be opened

**Returns**

APDM\_OK on success.

Referenced by `apdm_halt_all_attached_sensors()`, `apdm_sensor_allocate_and_open()`, and `apdm_sensor_get_device_id_list()`.

#### 5.5.1.14 **APDM\_EXPORT int apdm\_sensor\_override\_minimum\_supported\_version ( const char \* *new\_version* )**

Allows you to override the minimum motion sensor version number used to validate motion sensor versions.

**Parameters**

***new\_version*** Version number, e.g. "2010-09-02" Set this to NULL to use library default version number.

**Returns**

APDM\_OK on success, error code otherwise

#### 5.5.1.15 **APDM\_EXPORT int apdm\_sensor\_populate\_device\_info ( apdm\_device\_handle\_t *device\_handle*, apdm\_device\_info\_t \* *dest* )**

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

References `apdm_get_time_ms_64()`, `apdm_log_debug()`, `apdm_log_error()`, `apdm_sensor_cmd_calibration_data_blob()`, `apdm_sensor_cmd_calibration_version()`, `apdm_sensor_cmd_case_id()`, `apdm_sensor_cmd_config_get()`, `apdm_sensor_cmd_device_id()`, `apdm_sensor_cmd_hw_id()`, `apdm_sensor_cmd_protocol_version()`, `apdm_sensor_cmd_user_calibration_data_blob()`, `apdm_sensor_cmd_version_string_1()`, `apdm_sensor_cmd_version_string_2()`, `apdm_sensor_cmd_version_string_3()`, `apdm_sensor_config_get_label()`, `apdm_strerror()`, `apdm_device_info_t::protocol_version`, `apdm_device_info_t::wireless_addr_id`, `apdm_device_info_t::wireless_block0`, `apdm_device_info_t::wireless_block1`, `apdm_device_info_t::wireless_block2`, `apdm_device_`

info\_t::wireless\_block3, apdm\_device\_info\_t::wireless\_channel1, apdm\_device\_info\_t::wireless\_channel2, apdm\_device\_info\_t::wireless\_channel3, and apdm\_device\_info\_t::wireless\_timeslice.

#### 5.5.1.16 APDM\_EXPORT int apdm\_sensor\_verify\_supported\_calibration\_version ( apdm\_device\_handle\_t *device\_handle* )

this function is used to verify that the given device handle has a version of calibration data that is supported by the libraries.

##### Parameters

*device\_handle* The device handle

##### Returns

APDM\_OK if the version is OK, respective error code otherwise.

References apdm\_sensor\_cmd\_calibration\_version().

#### 5.5.1.17 APDM\_EXPORT int apdm\_sensor\_verify\_supported\_version ( apdm\_device\_handle\_t *device\_handle* )

this function is used to verify that the given device handle has a version of firmware that is supported by the libraries.

##### Parameters

*device\_handle* The device handle

##### Returns

APDM\_OK if the version is OK, respective error code otherwise.

References apdm\_log\_error(), and apdm\_sensor\_cmd\_version\_string\_2().

## 5.6 MonitorCommands

### Functions

- APDM\_EXPORT int [apdm\\_sensor\\_close\\_and\\_free](#) (apdm\_device\_handle\_t device\_handle)

- APDM\_EXPORT int [apdm\\_sensor\\_get\\_monitor\\_type](#) (const char \*case\_id\_string, apdm\_monitor\_type\_t \*dest)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_halt](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_reset](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_run](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_memory\\_crc16](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, const uint16\_t length, uint16\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_write\\_flash\\_block](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, uint8\_t \*data, const uint32\_t length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_time\\_set](#) (apdm\_device\_handle\_t device\_handle, uint32\_t year, uint32\_t month, uint32\_t day, uint32\_t hour, uint32\_t minute, uint32\_t second)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_time\\_set2](#) (apdm\_device\_handle\_t device\_handle, const time\_t epoch\_time)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_time\\_get](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*year, uint32\_t \*month, uint32\_t \*day, uint32\_t \*hour, uint32\_t \*minute, uint32\_t \*second)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_flash\\_block\\_set](#) (apdm\_device\_handle\_t device\_handle, uint32\_t block)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_flash\\_block\\_get](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*block)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_battery\\_voltage](#) (apdm\_device\_handle\_t device\_handle, uint16\_t \*voltage)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_battery\\_charge\\_rate](#) (apdm\_device\_handle\_t device\_handle, uint16\_t rate)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_calibration\\_version](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*calibration\_version)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_memory\\_dump](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t monitor\_memory\_address, const int num\_bytes\_to\_read, char \*BYTE\_ARRAY, const int dest\_buffer\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_peek](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, uint8\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_peek2](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, uint16\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_poke](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, uint8\_t new\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_poke2](#) (apdm\_device\_handle\_t device\_handle, const uint32\_t address, uint16\_t new\_value)

- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sync\\_get](#) (apdm\_device\_handle\_t device\_handle, uint64\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sync\\_dock\\_wait](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_led\\_pattern](#) (apdm\_device\_handle\_t device\_handle, uint8\_t interval, uint8\_t \*pattern, uint8\_t length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_led\\_reset](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_off\\_reason](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*reason)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_uptime\\_get](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*uptime)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_uptime\\_reset](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_last\\_uptime](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*uptime)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_last\\_standby\\_uptime](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*uptime)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_unlock\\_bootloader\\_flash](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_enter\\_bootloader](#) (apdm\_device\_handle\_t device\_handle, const char \*password, const int password\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_bootloader\\_version](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*dest\_version)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sample\\_start](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sample\\_get](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*dest\_buffer, const int buff\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sync\\_set](#) (apdm\_device\_handle\_t device\_handle, const uint64\_t new\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_sync\\_commit](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_config\\_commit](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_ping](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*mode)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_device\\_id](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_count](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*error\_count)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_name](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int length, uint16\_t error\_id)

- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_log\\_size](#) (apdm\_device\_handle\_t device\_handle, uint16\_t \*error\_log\_size)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_log\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t offset, uint16\_t \*error\_id)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_stats\\_size](#) (apdm\_device\_handle\_t device\_handle, uint16\_t \*stats\_size)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_stats\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t id, uint16\_t \*count)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_size](#) (apdm\_device\_handle\_t device\_handle, uint16\_t \*value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_max\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t id, uint16\_t \*max\_val)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_min\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t id, uint16\_t \*min\_val)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_count\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t id, uint16\_t \*count\_val)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_sum\\_get](#) (apdm\_device\_handle\_t device\_handle, const uint16\_t id, uint32\_t \*sum\_val)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_stats\\_clear](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_error\\_clear](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_battery\\_charge\\_status](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*current\_status)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_calibration\\_data\\_blob](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*dest, const int dest\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_user\\_calibration\\_data\\_blob](#) (apdm\_device\_handle\_t dev\_handle, uint8\_t \*dest, const int dest\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_calibration\\_data](#) (apdm\_device\_handle\_t device\_handle, [apdm\\_sensor\\_compensation\\_t](#) \*sensor\_comp)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_user\\_calibration\\_data](#) (apdm\_device\_handle\_t dev\_handle, [apdm\\_sensor\\_compensation\\_t](#) \*sensor\_comp)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_version\\_string\\_1](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int dest\_buff\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_version\\_string\\_2](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int dest\_buff\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_version\\_string\\_3](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int dest\_buff\_length)



- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_dock\\_status](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*status)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_config\\_get](#) (apdm\_device\_handle\_t device\_handle, const enum APDMDeviceConfig config\_type, uint32\_t \*value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_config\\_set](#) (apdm\_device\_handle\_t device\_handle, const enum APDMDeviceConfig config\_type, const uint32\_t value)
- APDM\_EXPORT int [apdm\\_sensor\\_config\\_set\\_label](#) (apdm\_device\_handle\_t device\_handle, const char label\_str[16], const int str\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_config\\_get\\_label](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int buff\_size)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_config\\_status](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*status)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_timer\\_adjust\\_get](#) (apdm\_device\_handle\_t device\_handle, uint16\_t \*value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_debug\\_set](#) (apdm\_device\_handle\_t device\_handle, uint8\_t id, uint32\_t data)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_debug\\_get](#) (apdm\_device\_handle\_t device\_handle, uint8\_t id, uint32\_t \*data)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_dock](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_undock](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_config\\_check](#) (apdm\_device\_handle\_t device\_handle, uint8\_t \*is\_valid)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_flash\\_format](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_standby](#) (apdm\_device\_handle\_t device\_handle)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_case\\_id](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int dest\_buff\_length)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_hw\\_id](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*current\_value)
- APDM\_EXPORT int [apdm\\_sensor\\_cmd\\_protocol\\_version](#) (apdm\_device\_handle\_t h, uint32\_t \*protocol\_version)

## 5.6.1 Function Documentation

### 5.6.1.1 APDM\_EXPORT int apdm\_sensor\_close\_and\_free (apdm\_device\_handle\_t device\_handle )

Closes and de-allocates a device handle.

**Parameters**

***device\_handle*** The handle to be closed and deallocated

**Returns**

APDM\_OK on success, other code on failure.

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming().

**5.6.1.2 APDM\_EXPORT int apdm\_sensor\_cmd\_battery\_charge\_rate ( apdm\_device\_handle\_t *device\_handle*, uint16\_t *rate* )**

Sets the battery charge rate

**Parameters**

***device\_handle*** The device handle.

***rate*** Units of milliamps, minimum = 100mA, max = 450mA;

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.3 APDM\_EXPORT int apdm\_sensor\_cmd\_battery\_charge\_status ( apdm\_device\_handle\_t *device\_handle*, uint8\_t \* *current\_status* )**

Retrieves the battery charge status

**Parameters**

***device\_handle*** The device handle.

**\**current\_status*** Destination into which to store the battery charge status, values are defined in "enum APDM\_Battery\_Charge\_Status" in [apdm\\_types.h](#).

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.4 APDM\_EXPORT int apdm\_sensor\_cmd\_battery\_voltage ( apdm\_device\_handle\_t *device\_handle*, uint16\_t \* *voltage* )**

#### Parameters

***device\_handle*** The device handle.

**\**voltage*** Destination into which to store the battery voltage, this is in units of the raw ADC value off the MCU.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.5 APDM\_EXPORT int apdm\_sensor\_cmd\_bootloader\_version ( apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *dest\_version* )**

This command is only supported on v1.1 or later monitors. Previous monitor version will result in a return code of APDM\_DEVICE\_RESPONSE\_ERROR\_INVALID\_COMMAND.

#### Parameters

***device\_handle*** The device handle.

**\**dest\_version*** The destination into which to store the bootloader version number

#### Returns

APDM\_OK on success, error code otherwise

**5.6.1.6 APDM\_EXPORT int apdm\_sensor\_cmd\_calibration\_data ( apdm\_device\_handle\_t *device\_handle*, apdm\_sensor\_compensation\_t \* *sensor\_comp* )**

This function will retrieve the sensor calibration data from the given devices via dev\_handle

#### Parameters

***device\_handle*** The device handle.

**\**sensor\_comp*** Destination into which to store sensor compensation data

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.7 APDM\_EXPORT** int apdm\_sensor\_cmd\_calibration\_data\_blob ( apdm\_device\_handle\_t *device\_handle*, uint8\_t \* *dest*, const int *dest\_length* )

Returns the packed binary representation of the motion monitor calibration data

#### Parameters

***device\_handle*** The device handle.

**\**dest*** Destination buffer into which to store the packed cal data.

***dest\_length*** The size of the destination buffer.

#### Returns

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.8 APDM\_EXPORT** int apdm\_sensor\_cmd\_calibration\_version ( apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *calibration\_version* )

Gets the version of calibration data currently on the motion monitor

#### Parameters

***device\_handle*** The device handle.

**\**calibration\_version*** Destination into which to store the calibration version number

#### Returns

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info(), and apdm\_sensor\_verify\_supported\_calibration\_version().

**5.6.1.9 APDM\_EXPORT** int apdm\_sensor\_cmd\_case\_id ( apdm\_device\_handle\_t *device\_handle*, char \* *BYTE\_ARRAY*, const int *dest\_buff\_length* )

Retrieves the case ID from the motion monitor

#### Parameters

***device\_handle*** The device handle.

\***BYTE\_ARRAY** Destination buffer into which to store the case ID  
**dest\_buff\_length** size of \*BYTE\_ARRAY

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_debug().

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.10 APDM\_EXPORT int apdm\_sensor\_cmd\_config\_check (**  
**apdm\_device\_handle\_t device\_handle, uint8\_t \* is\_valid )**

#### Parameters

**device\_handle** The device handle.

#### Returns

APDM\_OK on success, error code otherwise. Sets what is pointed to by is\_valid to 1 if the configuration is valid, sets it to 0 otherwise.

**5.6.1.11 APDM\_EXPORT int apdm\_sensor\_cmd\_config\_commit (**  
**apdm\_device\_handle\_t device\_handle )**

#### Parameters

**device\_handle** The device handle.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.12 APDM\_EXPORT int apdm\_sensor\_cmd\_config\_get (**  
**apdm\_device\_handle\_t device\_handle, const enum**  
**APDMDeviceConfig config\_type, uint32\_t \* value )**

Retrieves a specified configuration parameter type.

#### Parameters

**device\_handle** The device handle.

**config\_type** The configuration parameter type.

\***value** The destination into which to store the parameter value.

### Returns

APDM\_OK on success, error code otherwise.

Referenced by `apdm_sensor_config_get_label()`, and `apdm_sensor_populate_device_info()`.

**5.6.1.13** **APDM\_EXPORT** int **apdm\_sensor\_cmd\_config\_set** (  
    **apdm\_device\_handle\_t** *device\_handle*, const enum  
    **APDMDeviceConfig** *config\_type*, const uint32\_t *value* )

Sets a specified configuration parameter.

### Parameters

**device\_handle** The device handle.

**config\_type** The parameter which is to be set.

**value** The value to which it is to be set. There are enums in [apdm\\_types.h](#) to specify valid values for various parameter types.

### Returns

APDM\_OK on success, error code otherwise.

Referenced by `apdm_sensor_config_set_label()`.

**5.6.1.14** **APDM\_EXPORT** int **apdm\_sensor\_cmd\_config\_status** (  
    **apdm\_device\_handle\_t** *device\_handle*, uint8\_t \* *status* )

This command returns the current status of if the configuration has been committed or not.

### Parameters

**device\_handle** The device handle.

\***status** Destination into which to put the configuration status, 0 indicates the config has not been committed, 1 indicates that it has been committed.

### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.15** **APDM\_EXPORT** int apdm\_sensor\_cmd\_debug\_get (  
apdm\_device\_handle\_t *device\_handle*, uint8\_t *id*, uint32\_t \*  
*data* )

This command gets the debug value identified by the id parameter.

#### Parameters

***device\_handle*** The device handle.

***id*** The debug variable ID which is to be retrieved

***data*** The destination into which to store the debug value.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.16** **APDM\_EXPORT** int apdm\_sensor\_cmd\_debug\_set (  
apdm\_device\_handle\_t *device\_handle*, uint8\_t *id*, uint32\_t  
*data* )

This command sets the debug value identified by the id parameter.

#### Parameters

***device\_handle*** The device handle.

***id*** The debug variable ID which is to be set

***data*** The value to which it is to be set.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.17** **APDM\_EXPORT** int apdm\_sensor\_cmd\_device\_id (  
apdm\_device\_handle\_t *device\_handle*, uint32\_t \*  
*current\_value* )

Retrieves the device ID off the motion monitor

#### Parameters

***device\_handle*** The device handle.

**\**current\_value*** Destination into which to store the device id.

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by `apdm_sensor_get_device_id_list()`, and `apdm_sensor_populate_device_info()`.

**5.6.1.18 APDM\_EXPORT int apdm\_sensor\_cmd\_dock ( apdm\_device\_handle\_t *device\_handle* )****Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.19 APDM\_EXPORT int apdm\_sensor\_cmd\_dock\_status ( apdm\_device\_handle\_t *device\_handle*, uint8\_t \* *status* )****Parameters**

***device\_handle*** The device handle.

***\*status*** Values defined in enum `apdm_monitor_dock_status_t`

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.20 APDM\_EXPORT int apdm\_sensor\_cmd\_enter\_bootloader ( apdm\_device\_handle\_t *device\_handle*, const char \* *password*, const int *password\_length* )****Parameters**

***device\_handle*** The device handle.

***password*** The password, of length 8, to enter the bootloader (password differs based on monitor version)

***password\_length*** The length of the password, must be 8 bytes long.

**Returns**

APDM\_OK on success, error code otherwise.

References `apdm_log_error()`.



**5.6.1.21** `APDM_EXPORT int apdm_sensor_cmd_error_clear (`  
`apdm_device_handle_t device_handle )`

Clears all errors and error stats on the motion monitor.

**Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.22** `APDM_EXPORT int apdm_sensor_cmd_error_count (`  
`apdm_device_handle_t device_handle, uint32_t * error_count`  
`)`

Gets the number of errors on the motion monitor.

**Parameters**

**device\_handle** The device handle.

**\*error\_count** Destination into which to store the error count.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.23** `APDM_EXPORT int apdm_sensor_cmd_error_log_get (`  
`apdm_device_handle_t device_handle, const uint16_t offset,`  
`uint16_t * error_id )`

Retrieves the number of times the error at offset has occurred.

**Parameters**

**device\_handle** The device handle.

**offset** The error number offset to retrieve

**\*error\_id** Destination into which to store the error count for the specified offset.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.24** **APDM\_EXPORT** int apdm\_sensor\_cmd\_error\_log\_size  
( apdm\_device\_handle\_t *device\_handle*, uint16\_t \*  
*error\_log\_size* )

Retrieves the size of the error log on the motion monitor

**Parameters**

***device\_handle*** The device handle.

***\*error\_log\_size*** Destination into which to store the error log size.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.25** **APDM\_EXPORT** int apdm\_sensor\_cmd\_error\_name (  
apdm\_device\_handle\_t *device\_handle*, char \* **BYTE\_ARRAY**,  
const int *length*, uint16\_t *error\_id* )

Retrieves the name of the specified error ID.

**Parameters**

***device\_handle*** The device handle.

***\*BYTE\_ARRAY*** Destination string into which to store the name of the error

***length*** The size of the *\*BYTE\_ARRAY* string buffer

***error\_id*** The ID for which you want to retrieve the error name

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.26** **APDM\_EXPORT** int apdm\_sensor\_cmd\_error\_stats\_get (  
apdm\_device\_handle\_t *device\_handle*, const uint16\_t *id*,  
uint16\_t \* *count* )

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.27** **APDM\_EXPORT** int apdm\_sensor\_cmd\_error\_stats\_size (  
apdm\_device\_handle\_t *device\_handle*, uint16\_t \* *stats\_size*  
)

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.28** **APDM\_EXPORT** int apdm\_sensor\_cmd\_flash\_block\_get (  
apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *block* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.29** **APDM\_EXPORT** int apdm\_sensor\_cmd\_flash\_block\_set (  
apdm\_device\_handle\_t *device\_handle*, uint32\_t *block* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.30** **APDM\_EXPORT** int apdm\_sensor\_cmd\_flash\_format (  
apdm\_device\_handle\_t *device\_handle* )

Causes the motion monitor to re-format it's SD card when it is removed from the docking station or device cable.

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

#### 5.6.1.31 **APDM\_EXPORT int apdm\_sensor\_cmd\_halt (** **apdm\_device\_handle\_t device\_handle )**

When the motion monitor is removed from the dock, or disconnected from the cable, the motion monitor will halt.

##### **Parameters**

**device\_handle** The device handle.

##### **Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_halt\_all\_attached\_sensors().

#### 5.6.1.32 **APDM\_EXPORT int apdm\_sensor\_cmd\_hw\_id (** **apdm\_device\_handle\_t device\_handle, uint32\_t \*** **current\_value )**

Retrieves the hardware ID of the motion monitor.

##### **Parameters**

**device\_handle** The device handle.

**\*current\_value** The destination into which to store the hardware ID.

##### **Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info().

#### 5.6.1.33 **APDM\_EXPORT int apdm\_sensor\_cmd\_last\_standby\_uptime (** **apdm\_device\_handle\_t device\_handle, uint32\_t \* uptime )**

This command returns the last max uptime the device achieved while in standby mode. Mainly useful as a debugging command.

##### **Parameters**

**device\_handle** The device handle.

##### **Returns**

APDM\_OK on success, error code otherwise.

#### 5.6.1.34 APDM\_EXPORT int apdm\_sensor\_cmd\_last\_uptime ( apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *uptime* )

This command returns the last max uptime the device achieved while running before powering off or going into standby mode. Mainly useful as a debugging command.

##### Parameters

***device\_handle*** The device handle.

***\*uptime*** Destination into which to store the last uptime.

##### Returns

APDM\_OK on success, error code otherwise.

#### 5.6.1.35 APDM\_EXPORT int apdm\_sensor\_cmd\_led\_pattern ( apdm\_device\_handle\_t *device\_handle*, uint8\_t *interval*, uint8\_t \* *pattern*, uint8\_t *length* )

Led pattern is sent to the device as a character string which represents the led color pattern to display.

##### Parameters

***device\_handle*** The device handle.

***\*pattern***

##### Returns

APDM\_OK on success, error code otherwise.

#### 5.6.1.36 APDM\_EXPORT int apdm\_sensor\_cmd\_led\_reset ( apdm\_device\_handle\_t *device\_handle* )

Tells the device to go back to its normal led sequence (after having been overridden by [apdm\\_sensor\\_cmd\\_led\\_pattern\(\)](#))

##### Parameters

***device\_handle*** The device handle.

##### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.37** `APDM_EXPORT int apdm_sensor_cmd_memory_crc16 (`  
`apdm_device_handle_t device_handle, const uint32_t`  
`address, const uint16_t length, uint16_t * current_value )`

Does a CRC check on the given address and length of flash in the motion monitor

#### Parameters

**device\_handle** The device handle.

**address** The start address of the CRC check.

**length** The number of bytes to CRC

**\*current\_value** Destination into which to store the CRC value.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.38** `APDM_EXPORT int apdm_sensor_cmd_memory_dump (`  
`apdm_device_handle_t device_handle, const uint32_t`  
`monitor_memory_address, const int num_bytes_to_read,`  
`char * BYTE_ARRAY, const int dest_buffer_length )`

#### Parameters

**device\_handle** The device handle.

**monitor\_memory\_address** The start address in the monitors address space to start reading from

**num\_bytes\_to\_read** The number of bytes of memory to read from the device, must be > 0 and less then 32768

**\*BYTE\_ARRAY** The destination into which the memory dump is to be stored, must be non-null

**dest\_buffer\_length** The length of the destination buffer pointed to by \*BYTE\_ARRAY, must be > 0 and >= num\_bytes\_to\_read

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_log_error()`.

**5.6.1.39** `APDM_EXPORT int apdm_sensor_cmd_off_reason (`  
`apdm_device_handle_t device_handle, uint8_t * reason )`

#### Parameters

**device\_handle** The device handle.

**\*reason** Destination into which to store the reason code, reason is defined in 'enum apdm\_monitor\_off\_reason\_t' in [apdm\\_types.h](#)

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.40** `APDM_EXPORT int apdm_sensor_cmd_peek (`  
`apdm_device_handle_t device_handle, const uint32_t`  
`address, uint8_t * current_value )`

Peeks an 8-bit value in the motion monitor address space.

#### Parameters

**device\_handle** The device handle.

**address** The address to be peeked

**\*current\_value** The destination pointer into which to store the value.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.41** `APDM_EXPORT int apdm_sensor_cmd_peek2 (`  
`apdm_device_handle_t device_handle, const uint32_t`  
`address, uint16_t * current_value )`

Peeks an 16-bit value in the motion monitor address space.

#### Parameters

**device\_handle** The device handle.

**address** The address to be peeked

**\*current\_value** The destination pointer into which to store the value.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.42 APDM\_EXPORT int apdm\_sensor\_cmd\_ping (**  
**apdm\_device\_handle\_t device\_handle, uint8\_t \* mode )**

This command queries if the device is present and what its state is in regards to the bootloader (pre-bootloader/bootloader/post-bootloader).

**Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.43 APDM\_EXPORT int apdm\_sensor\_cmd\_poke (**  
**apdm\_device\_handle\_t device\_handle, const uint32\_t**  
**address, uint8\_t new\_value )**

Writes an 8-bit value into the address space of the motion monitor, note, writing to flash address space won't work.

**Parameters**

**device\_handle** The device handle.

**address** The address at which to write to

**new\_value** The value to be written

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.44 APDM\_EXPORT int apdm\_sensor\_cmd\_poke2 (**  
**apdm\_device\_handle\_t device\_handle, const uint32\_t**  
**address, uint16\_t new\_value )**

Writes an 16-bit value into the address space of the motion monitor, note, writing to flash address space won't work.

**Parameters**

**device\_handle** The device handle.

**address** The address at which to write to

**new\_value** The value to be written



**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.45 APDM\_EXPORT int apdm\_sensor\_cmd\_protocol\_version ( apdm\_device\_handle\_t h, uint32\_t \* protocol\_version )**

Retrieves the protocol version number from the monitor. This represents the binary packing and semantics used during wireless transmission.

**Parameters**

**device\_handle** The device handle.

**\*protocol\_version** Pointer to where the protocol version is to be stored

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.46 APDM\_EXPORT int apdm\_sensor\_cmd\_reset ( apdm\_device\_handle\_t device\_handle )**

Causes the monitor to reset.

**Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.47 APDM\_EXPORT int apdm\_sensor\_cmd\_run ( apdm\_device\_handle\_t device\_handle )**

Commands the device to enter run mode.

**Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

This command is used to instruct the device to into run mode.

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.48 APDM\_EXPORT int apdm\_sensor\_cmd\_sample\_get (**  
**apdm\_device\_handle\_t *device\_handle*, uint8\_t \* *dest\_buffer*,**  
**const int *buff\_length* )**

Gets the 1 second of data that was initiated by the sample\_start() command.

The format of the data returned is a set of 4x 512byte blocks with 25 sample sets each. There will be 12bytes of padding at the end of each 512byte block. This provides the host with 100 total samples. Each sample is a 16bit value with a sample set packed in the following order AX,AY,AZ,GX,GY,GZ,MX,MY,MZ,T. (T=temperature)

**Parameters**

***device\_handle*** The device handle.

***\*dest\_buffer*** Destination buffer into which to store samples taken.

***buff\_length*** The length of the buffer pointed to by \*dest\_buffer, must be  
>= 2048 bytes

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

**5.6.1.49 APDM\_EXPORT int apdm\_sensor\_cmd\_sample\_start (**  
**apdm\_device\_handle\_t *device\_handle* )**

Starts a 1-second cycle of the device sampling data on its internal sensors. Using the following settings:

- output rate 128
- decimation factor 5x2
- no mag set/reset
- all sensors enabled
- temperature from gyro

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.50** **APDM\_EXPORT** int apdm\_sensor\_cmd\_standby (  
apdm\_device\_handle\_t ***device\_handle*** )

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.51** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_clear (  
apdm\_device\_handle\_t ***device\_handle*** )

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.52** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_count\_get (  
apdm\_device\_handle\_t ***device\_handle***, const uint16\_t ***id***,  
uint16\_t \* ***count\_val*** )

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.53** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_max\_get ( apdm\_device\_handle\_t *device\_handle*, const uint16\_t *id*, uint16\_t \* *max\_val* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.54** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_min\_get ( apdm\_device\_handle\_t *device\_handle*, const uint16\_t *id*, uint16\_t \* *min\_val* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.55** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_size ( apdm\_device\_handle\_t *device\_handle*, uint16\_t \* *value* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.56** **APDM\_EXPORT** int apdm\_sensor\_cmd\_stats\_sum\_get ( apdm\_device\_handle\_t *device\_handle*, const uint16\_t *id*, uint32\_t \* *sum\_val* )

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.57 APDM\_EXPORT int apdm\_sensor\_cmd\_sync\_commit ( apdm\_device\_handle\_t device\_handle )**

Commits the sync value previously set by cmd\_sync\_set() thus causing the change to take effect.

**Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.58 APDM\_EXPORT int apdm\_sensor\_cmd\_sync\_dock\_wait ( apdm\_device\_handle\_t device\_handle )****Parameters**

**device\_handle** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.59 APDM\_EXPORT int apdm\_sensor\_cmd\_sync\_get ( apdm\_device\_handle\_t device\_handle, uint64\_t \* current\_value )**

Retrieves the sync value currently on the motion monitor.

**Parameters**

**device\_handle** The device handle.

**\*current\_value** The destination into which to store the current sync value on the motion monitor

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.60** **APDM\_EXPORT** int apdm\_sensor\_cmd\_sync\_set (  
apdm\_device\_handle\_t *device\_handle*, const uint64\_t  
*new\_value* )

Sets the sync value on the motion monitor, should call cmd\_sync\_commit()  
sometime after the sync value is set.

#### Parameters

***device\_handle*** The device handle.

***new\_value*** The new sync value to be set.

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_debug().

**5.6.1.61** **APDM\_EXPORT** int apdm\_sensor\_cmd\_time\_get (  
apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *year*,  
uint32\_t \* *month*, uint32\_t \* *day*, uint32\_t \* *hour*, uint32\_t \*  
*minute*, uint32\_t \* *second* )

Retrieves the time from the motion monitor.

#### Parameters

***device\_handle*** The device handle.

**\**year***

**\**month***

**\**day***

**\**hour***

**\**minute***

**\**second***

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_debug().

**5.6.1.62** **APDM\_EXPORT** int apdm\_sensor\_cmd\_time\_set (  
apdm\_device\_handle\_t *device\_handle*, uint32\_t *year*,  
uint32\_t *month*, uint32\_t *day*, uint32\_t *hour*, uint32\_t  
*minute*, uint32\_t *second* )

Sets the time on the motion monitor

#### Parameters

***device\_handle*** The device handle.

***year*** 0-9999

***month*** 1-12

***day*** 1-31

***hour*** 0-23

***minute*** 0-59

***second*** 0-59

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_debug().

Referenced by apdm\_sensor\_cmd\_time\_set2().

**5.6.1.63** **APDM\_EXPORT** int apdm\_sensor\_cmd\_time\_set2 (  
apdm\_device\_handle\_t *device\_handle*, const time\_t  
*epoch\_time* )

Sets the time on the Motion Monitor in terms of the epoch time (number of seconds since 1970)

#### Parameters

***device\_handle*** The device handle.

***epoch\_time*** Number of seconds since 1970.

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_sensor\_cmd\_time\_set().

**5.6.1.64** `APDM_EXPORT int apdm_sensor_cmd_timer_adjust_get ( apdm_device_handle_t device_handle, uint16_t * value )`

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.65** `APDM_EXPORT int apdm_sensor_cmd_undock ( apdm_device_handle_t device_handle )`

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.66** `APDM_EXPORT int apdm_sensor_cmd_unlock_bootloader_flash ( apdm_device_handle_t device_handle )`

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.67** `APDM_EXPORT int apdm_sensor_cmd_uptime_get ( apdm_device_handle_t device_handle, uint32_t * uptime )`

**Parameters**

*device\_handle* The device handle.

**Returns**

APDM\_OK on success, error code otherwise.



**5.6.1.68 APDM\_EXPORT int apdm\_sensor\_cmd\_uptime\_reset ( apdm\_device\_handle\_t *device\_handle* )**

This command resets the uptime counter on the device. Mainly useful as a debugging command.

**Parameters**

***device\_handle*** The device handle.

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.69 APDM\_EXPORT int apdm\_sensor\_cmd\_user\_calibration\_data ( apdm\_device\_handle\_t *dev\_handle*, apdm\_sensor\_compensation\_t \* *sensor\_comp* )**

This function will retrieve the user-overridden sensor calibration data from the given devices via *dev\_handle*. Requires monitor firmware versions newer than March 2011.

**Parameters**

***device\_handle*** The device handle.

**\**sensor\_comp*** Destination into which to store sensor compensation data

**Returns**

APDM\_OK on success, error code otherwise.

**5.6.1.70 APDM\_EXPORT int apdm\_sensor\_cmd\_user\_calibration\_data\_blob ( apdm\_device\_handle\_t *dev\_handle*, uint8\_t \* *dest*, const int *dest\_length* )**

Retrieves the user calibration data from the sensor (from re-calibration in the field)

**Parameters**

***dev\_handle*** The device handle of which you want user calibration from.

**\**dest*** The destination buffer into which binary data is to be stored.

***dest\_length*** The length of the destination buffer, which is not to be over-run

**Returns**

APDM\_OK on success, error code otherwise

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.71** **APDM\_EXPORT** int apdm\_sensor\_cmd\_version\_string\_1 ( apdm\_device\_handle\_t *device\_handle*, char \* *BYTE\_ARRAY*, const int *dest\_buff\_length* )

**Parameters**

***device\_handle*** The device handle.

***\*BYTE\_ARRAY*** Destination into which to store the version string.

***dest\_buff\_length*** length of the *\*BYTE\_ARRAY* array.

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.72** **APDM\_EXPORT** int apdm\_sensor\_cmd\_version\_string\_2 ( apdm\_device\_handle\_t *device\_handle*, char \* *BYTE\_ARRAY*, const int *dest\_buff\_length* )

**Parameters**

***device\_handle*** The device handle.

***\*BYTE\_ARRAY*** Destination into which to store the version string.

***dest\_buff\_length*** length of the *\*BYTE\_ARRAY* array.

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info(), and apdm\_sensor\_verify\_supported\_version().

**5.6.1.73** **APDM\_EXPORT** int apdm\_sensor\_cmd\_version\_string\_3 ( apdm\_device\_handle\_t *device\_handle*, char \* *BYTE\_ARRAY*, const int *dest\_buff\_length* )

**Parameters**

***device\_handle*** The device handle.

\***BYTE\_ARRAY** Destination into which to store the version string.

**dest\_buff\_length** length of the \*BYTE\_ARRAY array.

#### Returns

APDM\_OK on success, error code otherwise.

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.74 APDM\_EXPORT int apdm\_sensor\_cmd\_write\_flash\_block**  
( apdm\_device\_handle\_t **device\_handle**, const uint32\_t  
**address**, uint8\_t \* **data**, const uint32\_t **length** )

#### Parameters

**device\_handle** The device handle.

#### Returns

APDM\_OK on success, error code otherwise.

**5.6.1.75 APDM\_EXPORT int apdm\_sensor\_config\_get\_label** (  
apdm\_device\_handle\_t **device\_handle**, char \* **BYTE\_ARRAY**,  
const int **buff\_size** )

Wrapper function for getting the label config parameter

#### Parameters

**device\_handle** The device handle.

**BYTE\_ARRAY** 16 character array destination to store the label.

**buff\_size** The size of the buffer into which to store the label, must be at least 16

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_debug(), apdm\_log\_error(), and apdm\_sensor\_cmd\_config\_get().

Referenced by apdm\_sensor\_populate\_device\_info().

**5.6.1.76** **APDM\_EXPORT** int **apdm\_sensor\_config\_set\_label** (   
apdm\_device\_handle\_t *device\_handle*, const char   
*label\_str*[16], const int *str\_length* )

Wrapper function for setting the label config parameter

#### Parameters

***device\_handle*** The device handle.

***label\_str*** 16 character array source for label.

***str\_length*** The length of the buffer pointed to by *label\_str*

#### Returns

APDM\_OK on success, error code otherwise.

References `apdm_log_debug()`, `apdm_log_error()`, and `apdm_sensor_cmd_config_set()`.

**5.6.1.77** **APDM\_EXPORT** int **apdm\_sensor\_get\_monitor\_type** ( const   
char \* *case\_id\_string*, apdm\_monitor\_type\_t \* *dest* )

Parses a case ID string from a motion monitor and identifies which type of monitor it is (opal, emerald, sapphire)

#### Parameters

***case\_id\_string*** Case ID from the monitor

**\**dest*** Destination into which the monitor type will be stored

#### Returns

APDM\_OK on success, error code otherwise.

## 5.7 DockingStation

### Functions

- **APDM\_EXPORT** int [apdm\\_sensor\\_get\\_num\\_attached\\_dockingstations1](#) (uint32\_t \**dest\_num\_docks*)
- **APDM\_EXPORT** int [apdm\\_ds\\_override\\_minimum\\_supported\\_version](#) (const uint64\_t *new\_version*)

- APDM\_EXPORT int [apdm\\_ds\\_get\\_docked\\_module\\_id](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*dest)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_protocol\\_subversion](#) (apdm\_device\_handle\_t device\_handle, int64\_t \*dest\_protocol\_subversion)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_hardware\\_version](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*dest)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_firmware\\_version](#) (apdm\_device\_handle\_t device\_handle, uint64\_t \*dest)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_case\\_id](#) (apdm\_device\_handle\_t device\_handle, char \*BYTE\_ARRAY, const int dest\_buffer\_length)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_serial\\_number\\_by\\_index](#) (const int docking\_station\_index, uint32\_t \*serial\_number)
- APDM\_EXPORT int [apdm\\_ds\\_is\\_monitor\\_present](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*output\_flag)
- APDM\_EXPORT int [apdm\\_ds\\_is\\_monitor\\_data\\_forwarding\\_enabled](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*output\_flag)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_serial](#) (apdm\_device\_handle\_t device\_handle, uint32\_t \*serial\_number)
- APDM\_EXPORT int [apdm\\_ds\\_get\\_index\\_by\\_serial\\_number](#) (const uint32\_t serial\_number, uint32\_t \*docking\_station\_index)

## 5.7.1 Function Documentation

**5.7.1.1 APDM\_EXPORT int apdm\_ds\_get\_case\_id (**  
**apdm\_device\_handle\_t device\_handle, char \* BYTE\_ARRAY,**  
**const int dest\_buffer\_length )**

### Parameters

**device\_handle** The docking station handle  
**\*BYTE\_ARRAY** Destination into which to store the Case ID String  
**dest\_buffer\_length** The length of the buffer to which dest\_buffer is pointing

### Returns

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

**5.7.1.2 APDM\_EXPORT int apdm\_ds\_get\_docked\_module\_id (**  
**apdm\_device\_handle\_t device\_handle, uint32\_t \* dest )**

Gets the Module ID that the dock things is currently placed in the dock.

**Parameters**

***device\_handle*** The docking station handle

***\*dest*** Destination into which you want the module stored into.

**Returns**

APDM\_OK on success, error code otherwise

Referenced by apdm\_apply\_autoconfigure\_sensor\_config().

**5.7.1.3 APDM\_EXPORT int apdm\_ds\_get\_firmware\_version ( apdm\_device\_handle\_t device\_handle, uint64\_t \* dest )**

Note only works in firmware/bootloaders after Nov 8, 2010

**Parameters**

***device\_handle*** The docking station handle

***\*dest*** The destination into which to store the dock firmware version

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

**5.7.1.4 APDM\_EXPORT int apdm\_ds\_get\_hardware\_version ( apdm\_device\_handle\_t device\_handle, uint32\_t \* dest )**

Note this only works in firmware/bootloaders after Nov 8, 2010

**Parameters**

***device\_handle*** The docking station handle

***\*dest*** The destination into which to store the hardware revision number of the dock

**Returns**

APDM\_OK on success, error code otherwise.

**5.7.1.5 APDM\_EXPORT int apdm\_ds\_get\_index\_by\_serial\_number**  
( const uint32\_t *serial\_number*, uint32\_t \*  
*docking\_station\_index* )

This will return the index of the of the docking station with with the specified serial number.

#### Parameters

- serial\_number*** The serial number of the docking station for which you want the index of.
- \**docking\_station\_index*** The destination into which you want the index to be stored.

#### Returns

APDM\_OK on success, error code otherwise.

References apdm\_ds\_get\_serial\_number\_by\_index(), apdm\_log\_error(), and apdm\_sensor\_get\_num\_attached\_dockingstations1().

**5.7.1.6 APDM\_EXPORT int apdm\_ds\_get\_protocol\_subversion**  
( apdm\_device\_handle\_t *device\_handle*, int64\_t \*  
*dest\_protocol\_subversion* )

#### Parameters

- device\_handle*** The device handle
- \**dest\_protocol\_subversion*** The destination into which to store the protocol version

#### Returns

APDM\_OK on success, error code otherwise.

**5.7.1.7 APDM\_EXPORT int apdm\_ds\_get\_serial (**  
apdm\_device\_handle\_t *device\_handle*, uint32\_t \*  
*serial\_number* )

#### Parameters

- device\_handle*** The docking station handle for which you want the serial number
- \**serial\_number*** Destination into which to store the dock serial number

**Returns**

APDM\_OK on success, error code otherwise

References apdm\_log\_warning().

**5.7.1.8 APDM\_EXPORT int apdm\_ds\_get\_serial\_number\_by\_index (**  
**const int *docking\_station\_index*, uint32\_t \* *serial\_number* )**

Used to retrieve the serial number of a given docking station index number.

**Parameters**

***docking\_station\_index*** Index of the docking station for which you want the serial number

**\**serial\_number*** Destination into which the serial number will be stored

**Returns**

APDM\_OK on success, error code otherwise.

Referenced by apdm\_ds\_get\_index\_by\_serial\_number().

**5.7.1.9 APDM\_EXPORT int apdm\_ds\_is\_monitor\_data\_forwarding\_**  
**enabled ( apdm\_device\_handle\_t *device\_handle*, uint32\_t \***   
***output\_flag* )**

**Parameters**

***device\_handle*** The device handle

**\**output\_flag*** Destination into which to store the current status of weather or not data forwarding is enabled, zero indicates data is not being forwarded, non-zero indicates it is being forwarded

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_log\_error().

**5.7.1.10 APDM\_EXPORT int apdm\_ds\_is\_monitor\_present (**  
**apdm\_device\_handle\_t *device\_handle*, uint32\_t \* *output\_flag***  
**)**

**Parameters**

***device\_handle*** The docking station handle



\***output\_flag** The destination into which to store the indicator as to whether or not there is a monitor present in the dock, zero indicates dock is empty, non-zero indicates a monitor is present.

References apdm\_log\_error().

#### 5.7.1.11 APDM\_EXPORT int apdm\_ds\_override\_minimum\_supported\_version ( const uint64\_t *new\_version* )

Allows you to override the minimum docking station version number used to validate dock versions.

##### Parameters

**new\_version** Version number, e.g. 20100902170629 Set this to zero to use library default version number.

##### Returns

APDM\_OK on success, error code otherwise

#### 5.7.1.12 APDM\_EXPORT int apdm\_sensor\_get\_num\_attached\_dockingstations1 ( uint32\_t \* *dest\_num\_docks* )

##### Parameters

\***dest\_num\_docks** Destination into which to store the number of docking stations attached to the host

##### Returns

APDM\_OK on success, error code otherwise

Referenced by apdm\_ds\_get\_index\_by\_serial\_number(), and apdm\_halt\_all\_attached\_sensors().

## 5.8 DataHandling

### Functions

- APDM\_EXPORT uint64\_t [apdm\\_calculate\\_sync\\_value\\_age](#) (const uint64\_t sync1, const uint64\_t sync2)

- APDM\_EXPORT uint64\_t [apdm\\_epoch\\_access\\_point\\_to\\_epoch\\_second](#) (const uint64\_t sync\_value)
- APDM\_EXPORT uint64\_t [apdm\\_epoch\\_access\\_point\\_to\\_epoch\\_millisecond](#) (const uint64\_t sync\_value)
- APDM\_EXPORT int [apdm\\_epoch\\_access\\_point\\_to\\_epoch\\_microsecond](#) (const uint64\_t sync\_value, struct timeval \*dest)
- APDM\_EXPORT uint64\_t [apdm\\_epoch\\_second\\_to\\_epoch\\_access\\_point](#) (const uint64\_t epochSecond)
- APDM\_EXPORT int [apdm\\_recalibrate\\_magnetometers\\_from\\_h5](#) (char \*file, double local\_field\_magnitude, uint8\_t \*calibration\_block, double \*uncalibrated\_data, double \*calibrated\_data, int32\_t \*num\_samples)
- APDM\_EXPORT int [apdm\\_recalibrate\\_magnetometers](#) (uint16\_t \*mag\_data, double \*temperature, int nSamples, double \*original\_calibration, double \*recalibrated, [apdm\\_sensor\\_compensation\\_t](#) \*sensor\_comp, double local\_field\_magnitude)
- APDM\_EXPORT int [apdm\\_recalibrate\\_gyroscopes\\_from\\_h5](#) (char \*file, uint8\_t \*calibration\_block)
- APDM\_EXPORT int [apdm\\_recalibrate\\_gyroscopes](#) (uint16\_t \*gyro\_data, double \*temperature, int nSamples, [apdm\\_sensor\\_compensation\\_t](#) \*sensor\_comp, bool isDockRecal)
- int [apdm\\_extract\\_next\\_sample\\_set](#) (apdm\_ctx\_t context, const bool checked\_all\_aps)

## 5.8.1 Function Documentation

### 5.8.1.1 APDM\_EXPORT uint64\_t apdm\_calculate\_sync\_value\_age ( const uint64\_t *sync1*, const uint64\_t *sync2* )

#### Returns

The number of milliseconds delta between the two passed synced value.

Referenced by [apdm\\_ctx\\_get\\_wireless\\_reliability\\_value\(\)](#), and [apdm\\_extract\\_next\\_sample\\_set\(\)](#).

### 5.8.1.2 APDM\_EXPORT int apdm\_epoch\_access\_point\_to\_epoch\_microsecond ( const uint64\_t *sync\_value*, struct timeval \* *dest* )

Converts a sync value to an epoch second and microseconds (point in time, as since 1970, that the sample was taken).

#### Parameters

***sync\_value*** The sync value

\***dest** Destination timeval struct into which to store the time,

### Returns

The corresponding epoch second and microseconds for the passed sync value (point in time, since 1970, that the sample was taken).

#### 5.8.1.3 APDM\_EXPORT uint64\_t apdm\_epoch\_access\_point\_to\_epoch\_millisecond ( const uint64\_t *sync\_value* )

Converts a sync value to an epoch millisecond (point in time, as number of milliseconds since 1970, that the sample was taken).

### Parameters

***sync\_value*** The sync value

### Returns

The corresponding epoch millisecond for the passed sync value (point in time, as number of milliseconds since 1970, that the sample was taken).

#### 5.8.1.4 APDM\_EXPORT uint64\_t apdm\_epoch\_access\_point\_to\_epoch\_second ( const uint64\_t *sync\_value* )

Converts a sync value to an epoch second.

### Parameters

***sync\_value*** The sync value

### Returns

The corresponding epoch second for the passed sync value.

#### 5.8.1.5 APDM\_EXPORT uint64\_t apdm\_epoch\_second\_to\_epoch\_access\_point ( const uint64\_t *epochSecond* )

Helper function to convert an epoch second to a sync-value

**Parameters**

***epochSecond*** Number of seconds since 1970, unix time.

**Returns**

The system sync value that represents that point in time.

#### 5.8.1.6 **int apdm\_extract\_next\_sample\_set ( apdm\_ctx\_t context, const bool checked\_all\_aps )**

This function will inspect the head elements of the correlation fifos and assemble a set of samples all of which have the same sync value and store that into a context-specific data structure.

The resulting list may not necessarily contain a sample from all devices, as data may have been dropped due to wireless issues, or max latency thresholds could have been exceeded while waiting for a given sensor's data.

**Parameters**

***context***

**Returns**

APDM\_OK on success, error code otherwise.

References apdm\_record\_t::accl\_x\_axis, apdm\_calculate\_sync\_value\_age(), apdm\_ctx\_estimate\_now\_sync\_value(), apdm\_ctx\_get\_expected\_number\_of\_sensors2(), apdm\_log\_debug(), apdm\_log\_error(), apdm\_log\_warning(), apdm\_strerror(), apdm\_record\_t::device\_info\_serial\_number, apdm\_record\_t::source\_ap\_index, and apdm\_record\_t::sync\_val32\_low.

Referenced by apdm\_ctx\_get\_next\_access\_point\_record\_list().

#### 5.8.1.7 **APDM\_EXPORT int apdm\_recalibrate\_gyroscopes ( uint16\_t \* gyro\_data, double \* temperature, int nSamples, apdm\_sensor\_compensation\_t \* sensor\_comp, bool isDockRecal )**

Recalibrates the gyroscopes for bias shifts.

**Parameters**

***\*gyro\_data*** 3xN array containing raw gyroscope measurements

***\*temperature*** Nx1 array containing calibrated temperature measurements

***nSamples*** Number of samples in the previous arrays

***sensor\_comp*** `apdm_sensor_compensation_t` struct containing calibration data for this monitor. Will be updated on return with new magnetometer bias.

### Returns

APDM\_OK on success, error code otherwise

References `apdm_log_debug()`, `apdm_log_error()`, `calibration_v5_t::gyro_y_bias`, and `calibration_v5_t::gyro_z_bias`.

Referenced by `apdm_recalibrate_gyroscopes_from_h5()`.

#### 5.8.1.8 APDM\_EXPORT int apdm\_recalibrate\_gyroscopes\_from\_h5 ( char \* *file*, uint8\_t \* *calibration\_block* )

Recalibrates the gyroscopes for bias shifts.

### Parameters

***file*** HDF5 file containing raw and calibrated data during a period of up to 5 minutes of sitting still on a table.

***calibration\_block*** Byte array to be populated with the updated calibration data. Must be 2048 bytes.

### Returns

APDM\_OK on success, error code otherwise

References `apdm_get_hdf_dataset_shape()`, `apdm_get_hdf_device_list()`, `apdm_log_debug()`, `apdm_log_error()`, `apdm_read_hdf_calibration_data()`, `apdm_read_hdf_dataset()`, `apdm_recalibrate_gyroscopes()`, `calibration_v5_t::gyro_y_bias`, and `calibration_v5_t::gyro_z_bias`.

#### 5.8.1.9 APDM\_EXPORT int apdm\_recalibrate\_magnetometers ( uint16\_t \* *mag\_data*, double \* *temperature*, int *nSamples*, double \* *original\_calibration*, double \* *recalibrated*, `apdm_sensor_compensation_t` \* *sensor\_comp*, double *local\_field\_magnitude* )

Recalibrates the magnetometers for bias shifts due to magnetization of monitor components.

**Parameters**

- \*mag\_data** 3xN array containing raw magnetometer measurements
- \*temperature** Nx1 array containing calibrated temperature measurements
- nSamples** Number of samples in the previous arrays
- original\_calibration** Output array containing the raw data used during recalibration calibrated with the original calibration data. 3\*nSamples, column major ordering
- recalibrated** Output array containing the raw data used during recalibration calibrated with the updated calibration data. 3\*nSamples, column major ordering
- sensor\_comp** [apdm\\_sensor\\_compensation\\_t](#) struct containing calibration data for this monitor. Will be updated on return with new magnetometer bias.
- local\_field\_magnitude** Local magnetic field strength, usually obtained through a geomagnetic model like (<http://www.ngdc.noaa.gov/geomagmodels/IGRFWMM.jsp>). Set to 0 to keep current value.

**Returns**

APDM\_OK on success, error code otherwise

References [apdm\\_log\\_debug\(\)](#), [apdm\\_log\\_error\(\)](#), [apdm\\_strerror\(\)](#), [calibration\\_v5\\_t::mag\\_x\\_scale](#), [calibration\\_v5\\_t::mag\\_y\\_bias](#), and [calibration\\_v5\\_t::mag\\_z\\_bias](#).

**5.8.1.10 APDM\_EXPORT int apdm\_recalibrate\_magnetometers\_from\_h5 ( char \* file, double local\_field\_magnitude, uint8\_t \* calibration\_block, double \* uncalibrated\_data, double \* calibrated\_data, int32\_t \* num\_samples )**

Recalibrates the magnetometers for bias shifts due to magnetization of monitor components.

**Parameters**

- mag\_recalibration** Structure containing the fields below:
- file** HDF5 file containing raw and calibrated data during a period of up to 5 minutes of rotation covering as much of the orientation space as possible in a uniform magnetic field
- local\_field\_magnitude** Local magnetic field strength, usually obtained through a geomagnetic model like (

<http://www.ngdc.noaa.gov/geomagmodels/IGRFWMM.jsp>  
 ). Set to 0 to use the same value as last time the monitor was calibrated.

**calibration\_block** Byte array to be populated with the updated calibration data. Must be 2048 bytes.

**original\_calibration** Output array containing the raw data used during recalibration calibrated with the original calibration data. 3\*num\_samples, column major ordering

**recalibrated** Output array containing the raw data used during recalibration calibrated with the updated calibration data. 3\*num\_samples, column major ordering

**num\_samples** Number of samples in the calibrated data arrays (actual array is 3 times larger). Updated with the number of samples actually written to the arrays. Should be at least 38400 (5 minutes)

### Returns

APDM\_OK on success, error code otherwise

## 5.9 Logging

### Functions

- APDM\_EXPORT int [apdm\\_set\\_log\\_level](#) (int log\_level)
- APDM\_EXPORT const char \* [apdm\\_logging\\_level\\_t\\_str](#) (const apdm\_logging\_level\_t level)
- APDM\_EXPORT int [apdm\\_set\\_log\\_file](#) (const char \*filePath)
- APDM\_EXPORT int [apdm\\_close\\_log\\_file](#) (void)
- APDM\_EXPORT int [apdm\\_log](#) (const char \*format,...)
- APDM\_EXPORT int [apdm\\_logl](#) (const enum APDM\_Logging\_Level level, const char \*format,...)
- APDM\_EXPORT int [apdm\\_log\\_debug](#) (const char \*format,...)
- APDM\_EXPORT int [apdm\\_log\\_info](#) (const char \*format,...)
- APDM\_EXPORT int [apdm\\_log\\_warning](#) (const char \*format,...)
- APDM\_EXPORT int [apdm\\_log\\_error](#) (const char \*format,...)
- APDM\_EXPORT int [apdm\\_log\\_context](#) (apdm\_ctx\_t context, const enum APDM\_Logging\_Level level)

## 5.9.1 Function Documentation

### 5.9.1.1 APDM\_EXPORT int apdm\_close\_log\_file ( void )

Closes the apdm log file (if its open)

#### Returns

APDM\_OK on success

Referenced by apdm\_set\_log\_file().

### 5.9.1.2 APDM\_EXPORT int apdm\_log ( const char \* *format*, ... )

Adds log message to the apdm log stream at DEBUG level.

#### Parameters

*format* printf-style format string

... var-args for printf-style values

#### Returns

APDM\_OK on success

### 5.9.1.3 APDM\_EXPORT int apdm\_log\_context ( apdm\_ctx\_t *context*, const enum APDM\_Logging\_Level *level* )

Logs all the details about the passed context to the apdm logging stream

#### Parameters

*context* The context to be logged

*level* The logging severity level to log as

#### Returns

APDM\_OK on success, error code otherwise

References apdm\_logl(), apdm\_monitor\_decimation\_rate\_t\_str(), apdm\_output\_select\_rate\_t\_str(), apdm\_wireless\_mode\_t\_str(), apdm\_device\_info\_t::decimation\_factor, apdm\_device\_info\_t::dock\_id\_during\_configuration, apdm\_device\_info\_t::protocol\_version, apdm\_device\_info\_t::wireless\_addr\_id,



apdm\_device\_info\_t::wireless\_block0, apdm\_device\_info\_t::wireless\_block1, apdm\_device\_info\_t::wireless\_block2, apdm\_device\_info\_t::wireless\_block3, apdm\_device\_info\_t::wireless\_channel1, apdm\_device\_info\_t::wireless\_channel2, apdm\_device\_info\_t::wireless\_channel3, and apdm\_device\_info\_t::wireless\_timeslice.

Referenced by apdm\_ctx\_open\_all\_access\_points(), and apdm\_ctx\_sync\_record\_list\_head().

#### 5.9.1.4 APDM\_EXPORT int apdm\_log\_debug ( const char \* *format*, ... )

Adds log message to the apdm log stream at DEBUG level.

##### Parameters

*format* printf-style format string  
... var-args for printf-style values

##### Returns

APDM\_OK on success

Referenced by apdm\_ap\_connect(), apdm\_ap\_get\_id(), apdm\_ap\_get\_version\_string(), apdm\_apply\_autoconfigure\_sensor\_config(), apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming(), apdm\_ctx\_get\_next\_access\_point\_record(), apdm\_ctx\_open\_all\_access\_points(), apdm\_ctx\_sync\_record\_list\_head(), apdm\_extract\_next\_sample\_set(), apdm\_halt\_all\_attached\_sensors(), apdm\_read\_hdf\_calibration\_data(), apdm\_read\_hdf\_dataset(), apdm\_read\_hdf\_timestamps(), apdm\_read\_raw\_file\_info(), apdm\_recalibrate\_gyroscopes(), apdm\_recalibrate\_gyroscopes\_from\_h5(), apdm\_recalibrate\_magnetometers(), apdm\_send\_accesspoint\_cmd(), apdm\_sensor\_allocate\_handle(), apdm\_sensor\_cmd\_case\_id(), apdm\_sensor\_cmd\_sync\_set(), apdm\_sensor\_cmd\_time\_get(), apdm\_sensor\_cmd\_time\_set(), apdm\_sensor\_config\_get\_label(), apdm\_sensor\_config\_set\_label(), apdm\_sensor\_free\_handle(), apdm\_sensor\_populate\_device\_info(), and apdm\_write\_record\_hdf().

#### 5.9.1.5 APDM\_EXPORT int apdm\_log\_error ( const char \* *format*, ... )

Adds log message to the apdm log stream at ERROR level.

##### Parameters

*format* printf-style format string

... var-args for printf-style values

### Returns

APDM\_OK on success

Referenced by `apdm_ap_get_case_id()`, `apdm_ap_get_io_value()`, `apdm_ap_get_mode()`, `apdm_ap_get_monitor_latency()`, `apdm_ap_set_io_value()`, `apdm_ap_verify_supported_version()`, `apdm_apply_autoconfigure_sensor_config()`, `apdm_autoconfigure_devices_and_accesspoint_streaming()`, `apdm_configure_accesspoint()`, `apdm_ctx_allocate_new_context()`, `apdm_ctx_ap_get_io_value()`, `apdm_ctx_ap_set_io_value()`, `apdm_ctx_extract_next_sample()`, `apdm_ctx_free_context()`, `apdm_ctx_get_ap_id_for_ap_index()`, `apdm_ctx_get_device_index_by_id3()`, `apdm_ctx_get_monitor_latency()`, `apdm_ctx_get_next_access_point_record()`, `apdm_ctx_get_next_access_point_record_list()`, `apdm_ctx_get_next_record()`, `apdm_ctx_get_num_samples_collected()`, `apdm_ctx_get_num_samples_collected_from_device()`, `apdm_ctx_get_sensor_compensation_data()`, `apdm_ctx_is_more_data_immediately_available()`, `apdm_ctx_open_all_access_points()`, `apdm_ctx_set_sensor_compensation_data()`, `apdm_ctx_sync_record_list_head()`, `apdm_ds_get_case_id()`, `apdm_ds_get_firmware_version()`, `apdm_ds_get_index_by_serial_number()`, `apdm_ds_is_monitor_data_forwarding_enabled()`, `apdm_ds_is_monitor_present()`, `apdm_extract_next_sample_set()`, `apdm_halt_all_attached_sensors()`, `apdm_read_raw_file_info()`, `apdm_recalibrate_gyroscopes()`, `apdm_recalibrate_gyroscopes_from_h5()`, `apdm_recalibrate_magnetometers()`, `apdm_send_accesspoint_cmd()`, `apdm_sensor_allocate_handle()`, `apdm_sensor_cmd_enter_bootloader()`, `apdm_sensor_cmd_memory_dump()`, `apdm_sensor_cmd_sample_get()`, `apdm_sensor_config_get_label()`, `apdm_sensor_config_set_label()`, `apdm_sensor_populate_device_info()`, and `apdm_sensor_verify_supported_version()`.

#### 5.9.1.6 APDM\_EXPORT int apdm\_log\_info ( const char \* *format*, ... )

Adds log message to the apdm log stream at INFO level.

### Parameters

***format*** printf-style format string

... var-args for printf-style values

### Returns

APDM\_OK on success

Referenced by `apdm_autoconfigure_devices_and_accesspoint_streaming()`, `apdm_autoconfigure_mesh_sync()`, and `apdm_autoconfigure_mesh_sync2()`.

apdm\_configure\_accesspoint(), apdm\_ctx\_allocate\_new\_context(), apdm\_ctx\_open\_all\_access\_points(), apdm\_halt\_all\_attached\_sensors(), and apdm\_sensor\_get\_device\_id\_list().

#### 5.9.1.7 APDM\_EXPORT int apdm\_log\_warning ( const char \* *format*, ... )

Adds log message to the apdm log stream at WARNING level.

##### Parameters

*format* printf-style format string  
... var-args for printf-style values

##### Returns

APDM\_OK on success

Referenced by apdm\_ap\_connect(), apdm\_ap\_get\_case\_id(), apdm\_ctx\_get\_next\_access\_point\_record(), apdm\_ds\_get\_serial(), apdm\_extract\_next\_sample\_set(), and apdm\_free\_ap\_handle().

#### 5.9.1.8 APDM\_EXPORT const char\* apdm\_logging\_level\_t\_str ( const apdm\_logging\_level\_t *level* )

##### Parameters

*level* The level for which you want the string representation

##### Returns

Pointer to string for the given log level

#### 5.9.1.9 APDM\_EXPORT int apdm\_logl ( const enum APDM\_Logging\_Level *level*, const char \* *format*, ... )

Adds a log message to the apdm log stream at the given loglevel using the format and args passed in.

##### Parameters

*level* The log level that the message should be logged at.  
*format* printf-style format string

... var-args for printf-style values

### Returns

APDM\_OK on success

Referenced by `apdm_log_context()`.

#### 5.9.1.10 APDM\_EXPORT int apdm\_set\_log\_file ( const char \* *filePath* )

Sets and opens a log file to be used by APDM libraries for logging purposes

### Parameters

***filePath*** The file to which logging data should be saved

### Returns

APDM\_OK on success.

References `apdm_close_log_file()`.

#### 5.9.1.11 APDM\_EXPORT int apdm\_set\_log\_level ( int *log\_level* )

Sets the current log level to be used by APDM libraries. Valid values are:

### Parameters

***log\_level*** Valid values are: APDM\_LL\_ALL = 0, APDM\_LL\_DEBUG = 1, APDM\_LL\_INFO = 2, APDM\_LL\_WARNING = 3, APDM\_LL\_ERROR = 4, APDM\_LL\_NONE = 5

### Returns

APDM\_OK on success.

## 5.10 Misc

### Functions

- APDM\_EXPORT [apdm\\_device\\_info\\_t](#) \* [apdm\\_streaming\\_config\\_get\\_device\\_info](#) ([apdm\\_streaming\\_config\\_t](#) \*streaming\_config, int sensor\_index)

- APDM\_EXPORT const char \* [apdm\\_monitor\\_error\\_id\\_str](#) (const apdm\_monitor\_error\_id\_t error\_id)
- APDM\_EXPORT const char \* [apdm\\_get\\_library\\_version](#) (void)
- APDM\_EXPORT const char \* [apdm\\_get\\_library\\_build\\_datetime](#) (void)
- APDM\_EXPORT uint64\_t [apdm\\_get\\_time\\_ms\\_64](#) (struct timeval \*dest)
- APDM\_EXPORT const char \* [apdm\\_strerror](#) (const enum APDM\_Status status\_code)
- APDM\_EXPORT const char \* [apdm\\_output\\_select\\_rate\\_t\\_str](#) (const apdm\_monitor\_output\_select\_rate\_t rate)
- APDM\_EXPORT const char \* [apdm\\_monitor\\_decimation\\_rate\\_t\\_str](#) (const apdm\_monitor\_decimation\_rate\_t rate)
- APDM\_EXPORT uint32\_t [apdm\\_monitor\\_output\\_select\\_rate\\_t\\_to\\_int](#) (const apdm\_monitor\_output\_select\_rate\_t rate)
- APDM\_EXPORT uint32\_t [apdm\\_monitor\\_get\\_expected\\_sync\\_delta](#) (const apdm\_monitor\_output\_select\_rate\_t rate)
- APDM\_EXPORT uint32\_t [apdm\\_monitor\\_decimation\\_rate\\_t\\_to\\_int](#) (const apdm\_monitor\_decimation\_rate\_t rate)
- APDM\_EXPORT const char \* [apdm\\_wireless\\_mode\\_t\\_str](#) (const apdm\_wireless\_mode\_t mode)
- APDM\_EXPORT enum APDM\_Status\_Severity [apdm\\_error\\_severity](#) (const int status)
- APDM\_EXPORT void [apdm\\_usleep](#) (const uint64\_t microseconds)
- APDM\_EXPORT void [apdm\\_msleep](#) (const uint64\_t milliseconds)
- uint64\_t [apdm\\_get\\_now\\_sync\\_value\\_host](#) (void)

## 5.10.1 Function Documentation

### 5.10.1.1 APDM\_EXPORT enum APDM\_Status\_Severity [apdm\\_error\\_severity](#) ( const int *status* )

Helper function to get the severity level of a given apdm status code (APDM\_Status)

#### Parameters

***status*** The APDM\_Status status code in question

#### Returns

APDM\_SEVERITY\_ERROR, APDM\_SEVERITY\_WARNING or APDM\_SEVERITY\_INFO depending on the respective error severity.

#### 5.10.1.2 **APDM\_EXPORT** const char\* apdm\_get\_library\_build\_datetime ( void )

##### **Returns**

The date on which the libraries were built.

#### 5.10.1.3 **APDM\_EXPORT** const char\* apdm\_get\_library\_version ( void )

##### **Returns**

The version of the host libraries currently being used

#### 5.10.1.4 **uint64\_t** apdm\_get\_now\_sync\_value\_host ( void )

##### **Returns**

the sync value for 'now', based on the host computers current clock time.  
Note: this does not account for clock drift errors between the host computer and the access point.

References apdm\_get\_time\_ms\_64().

Referenced by apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming(), and apdm\_ctx\_estimate\_now\_sync\_value().

#### 5.10.1.5 **APDM\_EXPORT** uint64\_t apdm\_get\_time\_ms\_64 ( struct timeval \* *dest* )

##### **Returns**

the number of milliseconds elapsed since the UNIX epoch. Works on both windows and linux.

Referenced by apdm\_ap\_connect(), apdm\_ap\_get\_version\_string(), apdm\_autoconfigure\_devices\_and\_accesspoint\_streaming(), apdm\_ctx\_open\_all\_access\_points(), apdm\_ctx\_sync\_record\_list\_head(), apdm\_get\_now\_sync\_value\_host(), apdm\_send\_accesspoint\_cmd(), and apdm\_sensor\_populate\_device\_info().

**5.10.1.6** `APDM_EXPORT const char* apdm_monitor_decimation_rate_t_str ( const apdm_monitor_decimation_rate_t rate )`

#### Parameters

***rate*** The `apdm_monitor_decimation_rate_t` for which you want the string representation.

#### Returns

The string representation of the rate passed in.

Referenced by `apdm_log_context()`.

**5.10.1.7** `APDM_EXPORT uint32_t apdm_monitor_decimation_rate_t_to_int ( const apdm_monitor_decimation_rate_t rate )`

#### Parameters

***rate*** The `apdm_monitor_decimation_rate_t` for which you want the numerical decimation rate.

#### Returns

The decimation rate, numerical, for the specified rate, E.G. `APDM_DECIMATE_5x2` maps to 10

Referenced by `apdm_initialize_device_info()`.

**5.10.1.8** `APDM_EXPORT const char* apdm_monitor_error_id_str ( const apdm_monitor_error_id_t error_id )`

#### Parameters

***error\_id*** The error ID, of type, `apdm_motion_monitor_error_id_t`, for which you want a string representation

#### Returns

Const `char*` pointing to string representation of the given error ID.

**5.10.1.9 APDM\_EXPORT uint32\_t apdm\_monitor\_get\_expected\_sync\_delta ( const apdm\_monitor\_output\_select\_rate\_t *rate* )**

#### Parameters

***rate*** For a given output rate, determine the expected sync delta between any two samples.

#### Returns

The expected sync delta

References apdm\_monitor\_output\_select\_rate\_t\_to\_int().

**5.10.1.10 APDM\_EXPORT uint32\_t apdm\_monitor\_output\_select\_rate\_t\_to\_int ( const apdm\_monitor\_output\_select\_rate\_t *rate* )**

#### Parameters

***rate*** The apdm\_monitor\_output\_select\_rate\_t for which you want the numerical output rate.

#### Returns

The output sample rate of the specified apdm\_monitor\_output\_select\_rate\_t, e.g APDM\_OUTPUT\_SELECT\_RATE\_128 maps to 128

Referenced by apdm\_initialize\_device\_info(), and apdm\_monitor\_get\_expected\_sync\_delta().

**5.10.1.11 APDM\_EXPORT void apdm\_msleep ( const uint64\_t *milliseconds* )**

Platform independent version of msleep().

#### Parameters

***milliseconds*** Number of milliseconds to sleep for



#### 5.10.1.12 **APDM\_EXPORT** const char\* apdm\_output\_select\_rate\_t\_str ( const apdm\_monitor\_output\_select\_rate\_t *rate* )

##### Parameters

***rate*** The apdm\_monitor\_output\_select\_rate\_t for which you want the string representation

##### Returns

The string representation of the rate passed in.

Referenced by apdm\_log\_context().

#### 5.10.1.13 **APDM\_EXPORT** apdm\_device\_info\_t\* apdm\_streaming\_config\_get\_device\_info ( apdm\_streaming\_config\_t \* *streaming\_config*, int *sensor\_index* )

Helper method to retrieve a reference to an [apdm\\_device\\_info\\_t](#) structure. Useful for the Java SWIG binding.

##### Parameters

\****streaming\_config*** Pointer to [apdm\\_streaming\\_config\\_t](#) structure to be used

***int*** *sensor\_index* The index into the array of [apdm\\_device\\_info\\_t](#) structures

##### Returns

Pointer to the corresponding [apdm\\_device\\_info\\_t](#) structure.

References apdm\_streaming\_config\_t::device\_info\_cache.

#### 5.10.1.14 **APDM\_EXPORT** const char\* apdm\_strerror ( const enum APDM\_Status *status\_code* )

Helper function to convert an apdm status code to a string.

##### Parameters

***status\_code*** The status code for which you want the string representation.

##### Returns

Pointer to a char array with a string representation of the status code.

Referenced by `apdm_ap_connect()`, `apdm_autoconfigure_devices_and_accesspoint_streaming()`, `apdm_ctx_get_next_access_point_record()`, `apdm_ctx_get_next_access_point_record_list()`, `apdm_ctx_get_num_samples_collected()`, `apdm_ctx_get_num_samples_collected_from_device()`, `apdm_ctx_open_all_access_points()`, `apdm_extract_next_sample_set()`, `apdm_halt_all_attached_sensors()`, `apdm_recalibrate_magnetometers()`, `apdm_sensor_get_device_id_list()`, and `apdm_sensor_populate_device_info()`.

#### 5.10.1.15 APDM\_EXPORT void apdm\_usleep ( const uint64\_t *microseconds* )

Helper function for working with HDF5 files. Reads all of the annotations stored in the .h5 file.

##### Parameters

**file** The .h5 file to load data from

**annotations** Array of `apdm_annotation_t` structures containing the annotations. If NULL, only `nAnnotations` is set.

**nAnnotations** Number of annotations in the file.

##### Returns

APDM\_OK on success Platform independent version of `usleep()`.

##### Parameters

**microseconds** Number of microseconds to sleep for

Referenced by `apdm_ctx_sync_record_list_head()`.

#### 5.10.1.16 APDM\_EXPORT const char\* apdm\_wireless\_mode\_t\_str ( const apdm\_wireless\_mode\_t *mode* )

##### Parameters

**mode** The `apdm_wireless_mode_t` for which you want the string representation of.

##### Returns

The string representation of the specified mode.

Referenced by `apdm_log_context()`.

## Chapter 6

# Class Documentation

### 6.1 `__attribute__` Struct Reference

#### Public Member Functions

- union {  
    int64\_t **calibration\_version**  
    apdm\_calibration\_data\_v4\_t **v4**  
    apdm\_calibration\_data\_v5\_t **v5**  
    uint8\_t **raw** [256]  
} **\_\_attribute\_\_** ((\_\_packed\_\_)) data

#### Public Attributes

- int64\_t **accl\_x\_bias**
- int64\_t **accl\_y\_bias**
- int64\_t **accl\_z\_bias**
- int64\_t **accl\_x\_bias\_temp**
- int64\_t **accl\_y\_bias\_temp**
- int64\_t **accl\_z\_bias\_temp**
- int64\_t **accl\_x\_scale**
- int64\_t **accl\_y\_scale**
- int64\_t **accl\_z\_scale**
- int64\_t **accl\_x\_scale\_temp**
- int64\_t **accl\_y\_scale\_temp**
- int64\_t **accl\_z\_scale\_temp**
- int64\_t **accl\_xy\_sensitivity**

- `int64_t accl_xz_sensitivity`
- `int64_t accl_yz_sensitivity`
- `int64_t gyro_x_bias`
- `int64_t gyro_y_bias`
- `int64_t gyro_z_bias`
- `int16_t gyro_z_bias_temp [61]`
- `int16_t nothing [3]`
- `int64_t gyro_x_bias_temp`
- `int64_t gyro_x_bias_temp2`
- `int64_t gyro_y_bias_temp`
- `int64_t gyro_y_bias_temp2`
- `int64_t gyro_x_scale`
- `int64_t gyro_y_scale`
- `int64_t gyro_z_scale`
- `int64_t gyro_x_scale_temp`
- `int64_t gyro_y_scale_temp`
- `int64_t gyro_z_scale_temp`
- `int64_t gyro_xy_sensitivity`
- `int64_t gyro_xz_sensitivity`
- `int64_t gyro_yz_sensitivity`
- `int64_t gyro_accl_roll`
- `int64_t gyro_accl_pitch`
- `int64_t gyro_accl_yaw`
- `int64_t mag_xy_sensitivity`
- `int64_t mag_xz_sensitivity`
- `int64_t mag_yz_sensitivity`
- `int64_t mag_x_bias`
- `int64_t mag_y_bias`
- `int64_t mag_z_bias`
- `int64_t mag_x_scale`
- `int64_t mag_y_scale`
- `int64_t mag_z_scale`
- `int64_t mag_accl_roll`
- `int64_t mag_accl_pitch`
- `int64_t mag_accl_yaw`
- `int64_t temperature_bias`
- `int64_t temperature_scale`
- `int64_t accl_z_dtemp_scale`
- `int64_t temperature_bias_msp`
- `int64_t temperature_scale_msp`
- `int64_t cal_version`
- `uint16_t accl_x_bias [61]`
- `uint16_t accl_y_bias [61]`

- `uint16_t accl_z_bias` [61]
- `int16_t nothing0`
- `uint16_t gyro_x_bias` [61]
- `uint16_t gyro_y_bias` [61]
- `uint16_t gyro_z_bias` [61]
- `int16_t nothing1`
- `int64_t mag_x_scale_temp`
- `int64_t mag_y_scale_temp`
- `int64_t mag_z_scale_temp`
- `uint16_t mag_x_bias` [61]
- `uint16_t mag_y_bias` [61]
- `uint16_t mag_z_bias` [61]
- `int16_t nothing2`
- `int64_t mag_x_offset`
- `int64_t mag_y_offset`
- `int64_t mag_z_offset`
- `int64_t mag_conversion_gain`
- `uint32_t cal_version`
- `uint32_t device_id`
- `uint8_t retries`
- `uint16_t event_id`
- `union {`
  - `} packet`
- `uint16_t ax`
- `uint16_t ay`
- `uint16_t az`
- `uint16_t gx`
- `uint16_t gy`
- `uint16_t gz`
- `uint16_t mx`
- `uint16_t my`
- `uint16_t mz`
- `uint16_t t`
- `uint32_t data_a`
- `uint32_t data_b`
- `uint32_t data_c`
- `uint32_t data_d`
- `uint32_t data_e`
- `uint32_t data_f`
- `uint32_t error_id`
- `uint32_t error_count`
- `uint32_t sync_low_32`

- uint32\_t **sync\_high\_32**
- uint32\_t **pre\_block**
- uint32\_t **post\_block**
- uint32\_t **samples\_per\_block**
- uint32\_t **max\_latency**
- uint64\_t **pre\_sync**
- uint64\_t **post\_sync**

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

- apdm\_internal.h

## 6.2 apdm\_access\_point\_configuration\_t Struct Reference

### Public Attributes

- uint8\_t **radio1\_channel**
- uint8\_t **radio2\_channel**
- uint32\_t **address\_blockA**
- uint32\_t **address\_blockB**
- uint64\_t **sync\_value\_subtractor**
- uint32\_t **single\_ap\_mode**
- uint32\_t **id**
- uint32\_t **board\_version**
- uint32\_t **sensor\_group**
- uint16\_t **ap\_group**
- char **case\_id** [64]
- uint32\_t **pipe\_mappings** [NUM\_PIPES\_IN\_PIPE\_MAPPING]

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

- apdm\_internal.h

## 6.3 apdm\_access\_point\_handle Struct Reference

```
#include <apdm_internal.h>
```

## Public Attributes

- struct libusb\_device\_handle \* **devh**
- uint8\_t **usb\_protocol\_version**
- uint64\_t **usb\_protocol\_subversion**
- bool **has\_flushed\_data**
- bool **has\_skipped\_first\_sample**
- [apdm\\_bulk\\_in\\_buffer\\_t](#) **ep\_in\_buffer**
- [apdm\\_bulk\\_in\\_buffer\\_t](#) **ep\_in\_binary\_buffer**
- uint8\_t **sensor\_group**
- uint32\_t **num\_remaining\_samples**
- uint64\_t **current\_ap\_sync\_value\_64**
- int64\_t **sync\_value\_clock\_modifier**
- uint32\_t **current\_ap\_sample\_counter**
- apdm\_ap\_wireless\_streaming\_status\_t **current\_led\_streaming\_status**
- uint32\_t **sample\_number**
- uint32\_t **total\_samples\_collected**
- int32\_t **data\_array\_start\_idx**
- int32\_t **data\_array\_end\_idx**
- [apdm\\_record\\_t](#) **data\_array** [DA\_SIZE]
- int32\_t **sync\_data\_array\_head**
- int32\_t **sync\_data\_array\_tail**
- [apdm\\_external\\_sync\\_data\\_t](#) **sync\_data\_array** [SYNC\_DATA\_ARRAY\_SIZE]
- int32\_t **opal\_event\_data\_array\_head**
- int32\_t **opal\_event\_data\_array\_tail**
- apdm\_opal\_event\_packet\_t **opal\_event\_packet\_data\_array** [MONITOR\_EVENT\_ARRAY\_SIZE]
- struct timeval **last\_read\_time**
- uint64\_t **ap\_firmware\_ver**
- char **ap\_firmware\_version** [1024]
- [apdm\\_access\\_point\\_configuration\\_t](#) **ap\_configuration**

### 6.3.1 Detailed Description

This structure maps to the client programmer data type of: typedef void\* apdm\_ap\_handle\_t;

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

- apdm\_internal.h

## 6.4 apdm\_annotation\_t Struct Reference

### Public Attributes

- uint64\_t **time**
- uint32\_t **device\_id**
- char **text** [2048]

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

- apdm\_types.h

## 6.5 apdm\_bulk\_in\_buffer\_t Struct Reference

### Public Attributes

- char **temp\_buffer** [16384]
- [apdm\\_byte\\_array\\_ring\\_buffer\\_t](#) **ring\_buffer**

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

- apdm\_internal.h

## 6.6 apdm\_byte\_array\_ring\_buffer\_t Struct Reference

### Public Attributes

- uint32\_t **current\_head**
- uint32\_t **current\_tail**
- char **data** [APDM\_RING\_BUFFER\_SIZE]

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

- apdm\_ring\_buffer.h



## 6.7 apdm\_case\_id\_t Struct Reference

### Public Attributes

- char **id** [CASE\_ID\_SIZE]

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

- apdm\_types.h

## 6.8 apdm\_context\_t Struct Reference

```
#include <apdm_internal.h>
```

### Public Attributes

- uint32\_t **num\_configured\_aps**
- [apdm\\_access\\_point\\_handle](#) **ap\_handle\_list** [APDM\_MAXIMUM\_NUM\_ACCESS\_POINTS]
- uint64\_t **last\_correlation\_fifo\_population\_time\_ms**
- [per\\_device\\_info\\_t](#) **sensor\_list** [APDM\_MAX\_NUMBER\_OF\_SENSORS]
- uint32\_t **num\_compenstation\_entries**
- int32\_t **temp**
- enum APDMErrHandlingBehavior **error\_handling\_behavior**
- uint16\_t **max\_data\_delay\_seconds**
- uint32\_t **expected\_sync\_delta**
- uint32\_t **total\_sample\_lists\_collected**
- bool **has\_returned\_full\_sample\_set\_flag**
- uint8\_t **temp\_buff** [DEFAULT\_READ\_SIZE \*2]
- apdm\_wireless\_mode\_t **wireless\_configuration\_mode**
- uint32\_t **initial\_sample\_retrieval\_count**
- uint32\_t **num\_omitted\_sample\_sets**
- uint32\_t **total\_omitted\_sample\_sets**
- uint32\_t **num\_omitted\_samples**
- uint32\_t **total\_omitted\_samples**
- bool **more\_data\_available\_flag**
- uint64\_t **last\_found\_sync\_value**
- uint64\_t **last\_returned\_sample\_list\_sync\_value**
- [apdm\\_device\\_sample\\_buffer\\_row\\_t](#) **most\_recent\_list**

### 6.8.1 Detailed Description

This structure maps to the client programmer data type of: `typedef void* apdm_ctx_t;`

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

- `apdm_internal.h`

## 6.9 `apdm_device_dtemp_filter_state_t` Struct Reference

### Public Attributes

- double **state** [2]
- double [state\\_transition\\_matrix](#) [4]
- double [process\\_noise\\_matrix](#) [4]
- double [measurement](#) [1]
- double [measurement\\_matrix](#) [2]
- double [measurement\\_noise\\_matrix](#) [1]
- double [error\\_covariance\\_matrix](#) [4]
- double [filtered\\_measurement](#) [1]

### 6.9.1 Member Data Documentation

**6.9.1.1** `double apdm_device_dtemp_filter_state_t::error_covariance_matrix[4]`

**6.9.1.2** `double apdm_device_dtemp_filter_state_t::filtered_measurement[1]`

**6.9.1.3** `double apdm_device_dtemp_filter_state_t::measurement[1]`

**6.9.1.4** `double apdm_device_dtemp_filter_state_t::measurement_matrix[2]`

initialize to [0;0;0;0;0;0;0];

6.9.1.5 **double** apdm\_device\_dtemp\_filter\_state\_t::measurement\_noise\_matrix[1]

6.9.1.6 **double** apdm\_device\_dtemp\_filter\_state\_t::process\_noise\_matrix[4]

initialize to [0 0; 0 1]

6.9.1.7 **double** apdm\_device\_dtemp\_filter\_state\_t::state\_transition\_matrix[4]

initialize to [0 0];

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

- apdm\_internal.h

## 6.10 apdm\_device\_fifo\_t Struct Reference

### Public Attributes

- int32\_t **head**
- int32\_t **tail**
- [apdm\\_record\\_t](#) **fifo\_data** [MAX\_SAMPLES\_THAT\_A\_DEVICE\_CAN\_BUFFER]

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

- apdm\_internal.h

## 6.11 apdm\_device\_info\_t Struct Reference

### Public Attributes

- bool **decimation\_bypass\_flag**
- bool **time\_good\_flag**
- bool **accelerometer\_full\_scale\_flag**
- bool **accelerometer\_enabled\_flag**
- bool **gyroscope\_enabled\_flag**
- bool **magnetometer\_enabled\_flag**

- bool **sd\_card\_enabled\_flag**
- bool **always\_off\_flag**
- bool **erase\_sd\_card\_after\_undocking**
- bool **enable\_button**
- uint8\_t **button\_mode**
- apdm\_monitor\_spin\_mode\_t **spin\_mode**
- uint8\_t **selected\_temperature\_sensor**
- apdm\_monitor\_decimation\_rate\_t **decimation\_rate**
- apdm\_monitor\_output\_select\_rate\_t **output\_select\_rate**
- uint16\_t **sample\_rate**
- uint32\_t [decimation\\_factor](#)
- int32\_t [timezone](#)
- char **device\_label** [DEVICE\_LABEL\_SIZE]
- uint8\_t **calibration\_binary\_blob** [CALIBRATION\_DATA\_BUFFER\_SIZE]
- uint32\_t **calibration\_version\_number**
- uint8\_t **user\_calibration\_binary\_blob** [CALIBRATION\_DATA\_BUFFER\_SIZE]
- uint32\_t **user\_calibration\_version\_number**
- uint32\_t **device\_id**
- uint32\_t **hardware\_id**
- char **sd\_file\_version** [9]
- char **firmware\_version\_string1** [VERSION\_STRING\_SIZE]
- char **firmware\_version\_string2** [VERSION\_STRING\_SIZE]
- int64\_t **firmware\_version\_string2\_number**
- char **firmware\_version\_string3** [VERSION\_STRING\_SIZE]
- char **case\_id** [CASE\_ID\_SIZE]
- apdm\_config\_mag\_set\_reset\_t **magnetometer\_set\_reset**
- apdm\_monitor\_recording\_mode\_t **recording\_mode**
- apdm\_monitor\_data\_mode\_t **data\_mode**
- bool **enable\_wireless**
- apdm\_wireless\_mode\_t **wireless\_protocol**
- uint8\_t [wireless\\_timeslice](#)
- uint8\_t [wireless\\_addr\\_id](#)
- uint32\_t [protocol\\_version](#)
- uint8\_t **wireless\_channel0**
- uint32\_t [wireless\\_block0](#)
- uint8\_t [wireless\\_channel1](#)
- uint32\_t [wireless\\_block1](#)
- uint8\_t [wireless\\_channel2](#)
- uint32\_t [wireless\\_block2](#)
- uint8\_t [wireless\\_channel3](#)
- uint32\_t [wireless\\_block3](#)
- uint32\_t [dock\\_id\\_during\\_configuration](#)
- uint32\_t **dock\_hardware\_version\_during\_configuration**

## 6.11.1 Detailed Description

## 6.11.2 Member Data Documentation

### 6.11.2.1 uint32\_t apdm\_device\_info\_t::decimation\_factor

E.G. 128, can be directly derived from output\_select\_rate, this should be set with results from [apdm\\_monitor\\_output\\_select\\_rate\\_t\\_to\\_int\(\)](#)

Referenced by apdm\_ctx\_get\_next\_access\_point\_record(), apdm\_initialize\_device\_info(), and apdm\_log\_context().

### 6.11.2.2 uint32\_t apdm\_device\_info\_t::dock\_id\_during\_configuration

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm\_log\_context().

### 6.11.2.3 uint32\_t apdm\_device\_info\_t::protocol\_version

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value. Defines the pipe-number that data will come in on for the monitor.

Referenced by apdm\_ctx\_get\_next\_access\_point\_record(), apdm\_log\_context(), and apdm\_sensor\_populate\_device\_info().

### 6.11.2.4 int32\_t apdm\_device\_info\_t::timezone

E.G. 10, can be directly derived from decimation\_rate, this should be set with results from [apdm\\_monitor\\_decimation\\_rate\\_t\\_to\\_int\(\)](#)

### 6.11.2.5 uint8\_t apdm\_device\_info\_t::wireless\_addr\_id

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm\_log\_context(), and apdm\_sensor\_populate\_device\_info().

#### 6.11.2.6 uint32\_t apdm\_device\_info\_t::wireless\_block0

CAUTION: modifying this can cause unpredictable behavior, allow `autoconfigure()` or other similar functions to set this value.

Referenced by `apdm_log_context()`, and `apdm_sensor_populate_device_info()`.

#### 6.11.2.7 uint32\_t apdm\_device\_info\_t::wireless\_block1

CAUTION: modifying this can cause unpredictable behavior, allow `autoconfigure()` or other similar functions to set this value.

Referenced by `apdm_log_context()`, and `apdm_sensor_populate_device_info()`.

#### 6.11.2.8 uint32\_t apdm\_device\_info\_t::wireless\_block2

CAUTION: modifying this can cause unpredictable behavior, allow `autoconfigure()` or other similar functions to set this value.

Referenced by `apdm_log_context()`, and `apdm_sensor_populate_device_info()`.

#### 6.11.2.9 uint32\_t apdm\_device\_info\_t::wireless\_block3

CAUTION: modifying this can cause unpredictable behavior, allow `autoconfigure()` or other similar functions to set this value.

Referenced by `apdm_log_context()`, and `apdm_sensor_populate_device_info()`.

#### 6.11.2.10 uint8\_t apdm\_device\_info\_t::wireless\_channel1

CAUTION: modifying this can cause unpredictable behavior, allow `autoconfigure()` or other similar functions to set this value.

Referenced by `apdm_autoconfigure_devices_and_accesspoint_streaming()`, `apdm_log_context()`, and `apdm_sensor_populate_device_info()`.

### 6.11.2.11 uint8\_t apdm\_device\_info\_t::wireless\_channel2

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm\_log\_context(), and apdm\_sensor\_populate\_device\_info().

### 6.11.2.12 uint8\_t apdm\_device\_info\_t::wireless\_channel3

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm\_log\_context(), and apdm\_sensor\_populate\_device\_info().

### 6.11.2.13 uint8\_t apdm\_device\_info\_t::wireless\_timeslice

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm\_log\_context(), and apdm\_sensor\_populate\_device\_info().

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

- apdm\_types.h

## 6.12 apdm\_device\_sample\_buffer\_row\_t Struct Reference

### Public Attributes

- bool **is\_full\_flag**
- bool **is\_partially\_populated\_flag**
- uint64\_t **sync\_val\_for\_this\_line**
- [apdm\\_record\\_t](#) **data\_records** [APDM\_MAX\_NUMBER\_OF\_SENSORS]

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

- apdm\_internal.h

## 6.13 apdm\_device\_state\_data\_t Struct Reference

### Public Attributes

- uint32\_t **device\_id**
- time\_t **last\_received\_data\_timestamp**
- apdm\_opal\_event\_packet\_t **last\_sync\_event\_received**
- uint64\_t **last\_received\_data\_sync\_value**
- uint32\_t **last\_received\_sample\_count**
- double **battery\_level**
- double **last\_temperature**
- double **last\_temperature\_diff**
- uint64\_t **last\_processed\_data\_sync\_value**
- bool **first\_sample**
- double **temperature\_derivative\_buffer** [NUM\_TEMPERATURE\_READINGS\_FOR\_AVERAGING]
- int **temperature\_derivative\_buffer\_index**
- uint64\_t **sync\_buffer** [NUM\_TEMPERATURE\_READINGS\_FOR\_AVERAGING]
- double **differentiator\_buffer** [NUM\_TEMPERATURE\_READINGS\_FOR\_DIFFERENTIATOR]
- int **differentiator\_buffer\_index**
- double **mag\_x\_buffer** [7]
- double **mag\_y\_buffer** [7]
- double **mag\_z\_buffer** [7]
- int **mag\_buffer\_index**
- int **calibration\_data\_validated**
- [apdm\\_orientation\\_info\\_t](#) **orientation\_info**
- int32\_t **retry\_count\_history** [APDM\_RETRY\_HISTORY\_LENGTH]
- uint32\_t **retry\_count\_history\_head\_index**

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

- apdm\_internal.h

## 6.14 apdm\_device\_status\_t Struct Reference

```
#include <apdm_types.h>
```



## Public Attributes

- int **result\_code**
- uint8\_t **gyro\_recalibration\_block** [CALIBRATION\_DATA\_BUFFER\_SIZE]
- enum APDM\_Status **gyro\_recalibration\_result**
- uint32\_t **sd\_mbytes\_total**
- uint32\_t **sd\_mbytes\_used**

### 6.14.1 Detailed Description

FIXME document this

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

- apdm\_types.h

## 6.15 apdm\_disk\_ll\_t Struct Reference

### Public Attributes

- FILE \* **file\_handle**
- int32\_t **current\_length**
- int32\_t **tail\_largest\_idx**
- int32\_t **head\_smallest\_idx**
- uint8\_t **free\_sample\_list** [APDM\_FREE\_SAMPLE\_LIST\_ARRAY\_SIZE]

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

- apdm\_internal.h

## 6.16 apdm\_external\_sync\_data\_t Struct Reference

### Public Attributes

- uint8\_t **data**
- uint8\_t **data\_type**

- uint64\_t **sync\_value**
- uint32\_t **ap\_id**

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

- apdm\_types.h

## 6.17 apdm\_mag\_dechop\_state\_t Struct Reference

### Public Attributes

- double **state** [2]
- double [state\\_transition\\_matrix](#) [4]
- double [process\\_noise\\_matrix](#) [4]
- double [measurement](#) [6]
- double [measurement\\_matrix](#) [12]
- double [measurement\\_noise\\_matrix](#) [6 \*6]
- double [error\\_covariance\\_matrix](#) [4]
- double [filtered\\_measurement](#) [6]
- double [stepResponse](#) [10]
- [apdm\\_mag\\_step\\_response\\_state\\_t](#) [stepResponseEstimate](#)
- int **set\_reset\_flag**
- int [polarity](#)
- int [iSample](#)

### 6.17.1 Detailed Description

### 6.17.2 Member Data Documentation

**6.17.2.1** `double apdm_mag_dechop_state_t::error_covariance_matrix[4]`

**6.17.2.2** `double apdm_mag_dechop_state_t::filtered_measurement[6]`

**6.17.2.3** `int apdm_mag_dechop_state_t::iSample`

**6.17.2.4** `double apdm_mag_dechop_state_t::measurement[6]`

**6.17.2.5** `double apdm_mag_dechop_state_t::measurement_matrix[12]`

initialize to [0;0;0;0;0;0;0];

**6.17.2.6** `double apdm_mag_dechop_state_t::measurement_noise_matrix[6 * 6]`

**6.17.2.7** `int apdm_mag_dechop_state_t::polarity`

**6.17.2.8** `double apdm_mag_dechop_state_t::process_noise_matrix[4]`

initialize to [0 0; 0 1]

**6.17.2.9** `double apdm_mag_dechop_state_t::state_transition_matrix[4]`

initialize to [0 0];

**6.17.2.10** `double apdm_mag_dechop_state_t::stepResponse[10]`

**6.17.2.11** `apdm_mag_step_response_state_t  
apdm_mag_dechop_state_t::stepResponseEstimate`

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

- `apdm_types.h`

## 6.18 apdm\_mag\_opt\_data\_t Struct Reference

### Public Attributes

- double \* **samples**
- double \* **temperature**
- double \* **cal\_samples**
- int **n**
- [calibration\\_v5\\_t](#) **sensor\_comp**
- nlopt\_opt **opt**

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

- apdm\_internal.h

## 6.19 apdm\_mag\_step\_response\_state\_t Struct Reference

### Public Attributes

- double **state** [2]
- double [state\\_transition\\_matrix](#) [4]
- double [process\\_noise\\_matrix](#) [4]
- double [measurement](#) [2]
- double [measurement\\_matrix](#) [4]
- double [measurement\\_noise\\_matrix](#) [4]
- double [error\\_covariance\\_matrix](#) [4]
- double [filtered\\_measurement](#) [2]

### 6.19.1 Member Data Documentation

6.19.1.1 `double apdm_mag_step_response_state_t::error_covariance_matrix[4]`

6.19.1.2 `double apdm_mag_step_response_state_t::filtered_measurement[2]`

6.19.1.3 `double apdm_mag_step_response_state_t::measurement[2]`

6.19.1.4 `double apdm_mag_step_response_state_t::measurement_matrix[4]`

initialize to [0;0;0;0;0;0;0];

6.19.1.5 `double apdm_mag_step_response_state_t::measurement_noise_matrix[4]`

6.19.1.6 `double apdm_mag_step_response_state_t::process_noise_matrix[4]`

initialize to [1 0 0; 0 1 0; 0 0 1]

6.19.1.7 `double apdm_mag_step_response_state_t::state_transition_matrix[4]`

initialize to [0 0 0];

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

- `apdm_types.h`

## 6.20 apdm\_magnetometer\_recalibration\_t Struct Reference

### Public Attributes

- `char * file`
- `double local_field_magnitude`
- `uint8_t calibration_block [2048]`

- double **original\_calibrated\_data** [115200]
- double **updated\_calibrated\_data** [115200]
- int **num\_samples**

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

- apdm\_types.h

## 6.21 apdm\_monitor\_error\_stat\_t Struct Reference

### Public Attributes

- apdm\_monitor\_error\_id\_t **error\_id**
- uint32\_t **error\_count**
- uint64\_t **sync\_value**

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

- apdm\_types.h

## 6.22 apdm\_monitor\_label\_t Struct Reference

### Public Attributes

- char **label** [DEVICE\_LABEL\_SIZE]

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

- apdm\_types.h

## 6.23 apdm\_orientation\_info\_t Struct Reference

### Public Attributes

- double **x** [6]
- double **y** [6]

- apdm\_orientation\_model\_t **model**
- double **state\_transition\_matrix** [36]
- double **process\_noise\_matrix** [36]
- double **error\_covariance\_matrix** [36]
- double **measurement\_matrix** [36]
- double **measurement\_covariance\_matrix** [36]
- double **filtered\_measurement** [6]
- int **state\_dimension**
- int **measurement\_dimension**
- double **gyro\_bias** [3]
- double **mag\_mag**
- double **acc\_mag**
- double **acc\_var**
- double **gyro\_var**
- double **mag\_var**
- double **mag\_inclination**
- double **inclination\_var**
- double **fs**
- int **iSample**
- double **q\_current** [4]
- double **q\_old** [4]

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

- apdm\_internal.h

## 6.24 apdm\_orientation\_info\_ukf\_t Struct Reference

### Public Attributes

- [apdm\\_orientation\\_ukf\\_fdata\\_t](#) **fdata**
- [apdm\\_orientation\\_ukf\\_hdata\\_t](#) **hdata**
- [apdm\\_ukf\\_state\\_t](#) **ukf\_state**

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

- apdm\_internal.h

## 6.25 apdm\_orientation\_ukf\_fdata\_t Struct Reference

### Public Attributes

- double **fs**
- double **gyro** [3]

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

- apdm\_internal.h

## 6.26 apdm\_orientation\_ukf\_hdata\_t Struct Reference

### Public Attributes

- int **iDelayBuffer**
- int **iSample**
- double **old\_gyro** [ORIENTATION\_BUFFER\_LENGTH \*3]
- double **old\_acc** [ORIENTATION\_BUFFER\_LENGTH \*3]
- double **old\_mag** [ORIENTATION\_BUFFER\_LENGTH \*3]
- apdm\_orientation\_model\_t **model**
- double **gyro\_bias** [3]
- double **gyro\_scale** [3]
- double **acc\_bias** [3]
- double **mag\_mag**
- double **acc\_mag**
- double **acc\_var**
- double **gyro\_var**
- double **mag\_var**
- double **mag\_inclination**
- double **inclination\_var**
- double **fs**
- double **acc** [3]
- double **gyro** [3]
- double **mag** [3]
- double **q\_current** [4]
- double **q\_old** [4]



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

- apdm\_internal.h

## 6.27 apdm\_progress\_t Struct Reference

### Public Attributes

- char **task** [128]
- int **num\_tasks**
- int **task\_index**
- double **percent\_complete**

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

- apdm\_types.h

## 6.28 apdm\_record\_ll\_disk\_data\_t Struct Reference

### Public Attributes

- int32\_t **idx**
- int32\_t **prev\_larger\_idx**
- int32\_t **next\_smaller\_idx**
- [apdm\\_record\\_t](#) **data**

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

- apdm\_internal.h

## 6.29 apdm\_record\_t Struct Reference

```
#include <apdm_types.h>
```

## Public Attributes

- uint64\_t **sync\_val64**
- uint32\_t [sync\\_val32\\_low](#)
- uint32\_t **sync\_val32\_high**
- uint8\_t **nRF\_pipe**
- uint8\_t [num\\_retrys](#)
- int32\_t [source\\_ap\\_index](#)
- uint16\_t [accl\\_x\\_axis](#)
- uint16\_t [accl\\_y\\_axis](#)
- uint16\_t [accl\\_z\\_axis](#)
- bool [accl\\_full\\_scale\\_mode](#)
- bool [accl\\_isPopulated](#)
- uint16\_t [gyro\\_x\\_axis](#)
- uint16\_t [gyro\\_y\\_axis](#)
- uint16\_t [gyro\\_z\\_axis](#)
- bool [gyro\\_isPopulated](#)
- uint16\_t [mag\\_x\\_axis](#)
- uint16\_t [mag\\_y\\_axis](#)
- uint16\_t [mag\\_z\\_axis](#)
- uint16\_t [mag\\_common\\_axis](#)
- bool [mag\\_isPopulated](#)
- uint8\_t [flag\\_accel\\_enabled](#)
- uint8\_t **flag\_gyro\_enabled**
- uint8\_t **flag\_mag\_enabled**
- uint8\_t **flag\_full\_scale\_enabled**
- uint8\_t **flag\_sync\_lock**
- uint8\_t **flag\_sync\_reset**
- uint8\_t **flag\_temp\_select**
- uint16\_t **flags**
- bool [gyro\\_temperature\\_sensor\\_selected](#)
- uint32\_t **optional\_data**
- uint8\_t [opt\\_select](#)
- uint32\_t [debug\\_data](#)
- uint16\_t **debug\_flags**
- double **temperature**
- double **temperature\_average**
- double **temperature\_diff**
- bool **temperature\_isPopulated**
- uint32\_t **batt\_voltage**
- bool [batt\\_voltage\\_isPopulated](#)
- uint32\_t [device\\_info\\_serial\\_number](#)
- uint8\_t [device\\_info\\_wireless\\_channel\\_id](#)

- uint8\_t [device\\_info\\_wireless\\_address](#)
- bool [device\\_info\\_isPopulated](#)
- uint8\_t [events\\_num\\_events](#)
- uint8\_t [events\\_event\\_list](#) [MAX\_APDM\_EVENTS]
- uint8\_t [button\\_status](#)
- uint32\_t [tag\\_data](#)
- bool [tag\\_data\\_isPopulated](#)
- double [accl\\_x\\_axis\\_si](#)
- double [accl\\_y\\_axis\\_si](#)
- double [accl\\_z\\_axis\\_si](#)
- double [gyro\\_x\\_axis\\_si](#)
- double [gyro\\_y\\_axis\\_si](#)
- double [gyro\\_z\\_axis\\_si](#)
- double [mag\\_x\\_axis\\_si](#)
- double [mag\\_y\\_axis\\_si](#)
- double [mag\\_z\\_axis\\_si](#)
- double [orientation\\_quaternion0](#)
- double [orientation\\_quaternion1](#)
- double [orientation\\_quaternion2](#)
- double [orientation\\_quaternion3](#)
- double [temperature\\_si](#)
- double [temperature\\_derivative\\_si](#)
- double [battery\\_level](#)

### 6.29.1 Detailed Description

APDM Sensor Sample

### 6.29.2 Member Data Documentation

#### 6.29.2.1 bool apdm\_record\_t::accl\_full\_scale\_mode

raw ADC readings

#### 6.29.2.2 bool apdm\_record\_t::accl\_isPopulated

True indicates accelerometers are in 6G mode, false indicates 2G mode

**6.29.2.3 uint16\_t apdm\_record\_t::accl\_x\_axis**

Index of the AP that the sample came in on.

Referenced by `apdm_extract_next_sample_set()`, and `apdm_write_record_hdf()`.

**6.29.2.4 uint16\_t apdm\_record\_t::accl\_y\_axis**

raw ADC readings

Referenced by `apdm_write_record_hdf()`.

**6.29.2.5 double apdm\_record\_t::accl\_y\_axis\_si**

meters per second<sup>2</sup>

Referenced by `apdm_write_record_hdf()`.

**6.29.2.6 uint16\_t apdm\_record\_t::accl\_z\_axis**

raw ADC readings

Referenced by `apdm_write_record_hdf()`.

**6.29.2.7 double apdm\_record\_t::accl\_z\_axis\_si**

meters per second<sup>2</sup>

Referenced by `apdm_write_record_hdf()`.

**6.29.2.8 bool apdm\_record\_t::batt\_voltage\_isPopulated**

raw ADC readings

**6.29.2.9 double apdm\_record\_t::battery\_level**

degrees C per sec

**6.29.2.10 uint32\_t apdm\_record\_t::debug\_data**

Indicates what type of data is in optional\_data, see enum apdm\_raw\_opt\_select\_t

**6.29.2.11 bool apdm\_record\_t::device\_info\_isPopulated****6.29.2.12 uint32\_t apdm\_record\_t::device\_info\_serial\_number**

Indicates that the battery voltage data is populated

Referenced by apdm\_ctx\_extract\_data\_by\_device\_id(), apdm\_ctx\_get\_next\_access\_point\_record(), and apdm\_extract\_next\_sample\_set().

**6.29.2.13 uint8\_t apdm\_record\_t::device\_info\_wireless\_address****6.29.2.14 uint8\_t apdm\_record\_t::device\_info\_wireless\_channel\_id**

Device ID

**6.29.2.15 uint8\_t apdm\_record\_t::events\_num\_events****6.29.2.16 uint8\_t apdm\_record\_t::flag\_accel\_enabled**

Indicates that the mag data is populated

**6.29.2.17 bool apdm\_record\_t::gyro\_isPopulated**

raw ADC readings

**6.29.2.18 bool apdm\_record\_t::gyro\_temperature\_sensor\_selected**

Flags packed binary structure, used to derive the flag\_XXXX field values.

**6.29.2.19 uint16\_t apdm\_record\_t::gyro\_x\_axis**

Indicates that the accel data is populated

Referenced by apdm\_write\_record\_hdf().

**6.29.2.20 double apdm\_record\_t::gyro\_x\_axis\_si**

meters per second<sup>2</sup>

Referenced by apdm\_write\_record\_hdf().

**6.29.2.21 uint16\_t apdm\_record\_t::gyro\_y\_axis**

raw ADC readings

Referenced by apdm\_write\_record\_hdf().

**6.29.2.22 double apdm\_record\_t::gyro\_y\_axis\_si**

radians per second

Referenced by apdm\_write\_record\_hdf().

**6.29.2.23 uint16\_t apdm\_record\_t::gyro\_z\_axis**

raw ADC readings

Referenced by apdm\_write\_record\_hdf().

**6.29.2.24 double apdm\_record\_t::gyro\_z\_axis\_si**

radians per second

Referenced by apdm\_write\_record\_hdf().

**6.29.2.25 uint16\_t apdm\_record\_t::mag\_common\_axis**

raw ADC readings

Referenced by apdm\_write\_record\_hdf().

**6.29.2.26 bool apdm\_record\_t::mag\_isPopulated**

only used with device protocol version 0. raw ADC readings

**6.29.2.27 uint16\_t apdm\_record\_t::mag\_x\_axis**

Indicates that the gyro data is populated

Referenced by apdm\_write\_record\_hdf().

**6.29.2.28 double apdm\_record\_t::mag\_x\_axis\_si**

radians per second

Referenced by apdm\_write\_record\_hdf().

**6.29.2.29 uint16\_t apdm\_record\_t::mag\_y\_axis**

raw ADC readings

Referenced by apdm\_write\_record\_hdf().

**6.29.2.30 uint16\_t apdm\_record\_t::mag\_z\_axis**

raw ADC readings

Referenced by apdm\_write\_record\_hdf().

**6.29.2.31 uint8\_t apdm\_record\_t::num\_retrys**

Internal use only

Referenced by apdm\_ctx\_get\_next\_access\_point\_record().

**6.29.2.32 uint8\_t apdm\_record\_t::opt\_select**

Optional and varying data from the monitor

Referenced by apdm\_ctx\_get\_next\_access\_point\_record().

**6.29.2.33 int32\_t apdm\_record\_t::source\_ap\_index**

Number of retry before this sample was received by the AP.

Referenced by apdm\_ctx\_get\_next\_access\_point\_record(), and apdm\_extract\_next\_sample\_set().

### 6.29.2.34 `uint32_t apdm_record_t::sync_val32_low`

Full 64 bit sync value

Referenced by `apdm_ctx_get_next_access_point_record()`, and `apdm_extract_next_sample_set()`.

### 6.29.2.35 `double apdm_record_t::temperature_derivative_si`

degrees celcius

Referenced by `apdm_write_record_hdf()`.

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

- `apdm_types.h`

## 6.30 `apdm_recording_info_t` Struct Reference

### Public Attributes

- [`apdm\_device\_info\_t`](#) `device_info`
- `uint64_t` `start_sync_count`
- `uint64_t` `end_sync_count`
- `int` `num_samples`

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

- `apdm_types.h`

## 6.31 `apdm_sensor_cmd` Struct Reference

### Public Member Functions

- `__attribute__((packed))` union
- payload `__attribute__((packed))`

### Public Attributes

- `uint8_t` `cmd_number`



- uint16\_t **payload\_size**
- uint16\_t **crc16**

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

- apdm\_internal.h

## 6.32 apdm\_sensor\_compensation\_t Struct Reference

### Public Attributes

- uint32\_t **converted\_calibration\_version**
- uint32\_t **raw\_calibration\_version**
- union {
  - [calibration\\_v4\\_t](#) **v4**
  - [calibration\\_v5\\_t](#) **v5**
- **data**

### 6.32.1 Detailed Description

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

- apdm\_types.h

## 6.33 apdm\_sensor\_device\_handle\_t Struct Reference

```
#include <apdm_internal.h>
```

### Public Attributes

- bool **is\_opal\_attached**
- struct libusb\_device\_handle \* **devh**
- [apdm\\_bulk\\_in\\_buffer\\_t](#) **ep\_ds\_opal\_in\_buffer**
- [apdm\\_bulk\\_in\\_buffer\\_t](#) **ep\_ds\_in\_buffer**

- uint8\_t **usb\_protocol\_version**
- uint64\_t **usb\_protocol\_subversion**
- uint32\_t **dock\_id**
- [apdm\\_device\\_info\\_t](#) **device\_info**
- [apdm\\_device\\_status\\_t](#) **offset\_test\_results**

### 6.33.1 Detailed Description

This structure maps to the client programmer data type of: typedef void\* apdm\_device\_handle\_t;

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

- apdm\_internal.h

## 6.34 apdm\_sensor\_response Struct Reference

### Public Attributes

- uint8\_t **response\_number**
- uint16\_t **payload\_size**
- union {
  - uint8\_t **in\_uint8\_t**
  - uint16\_t **in\_uint16\_t**
  - uint32\_t **in\_uint32\_t**
  - uint64\_t **in\_uint64\_t**
  - uint8\_t **ping\_mode**
  - uint8\_t **peek\_value**
  - uint16\_t **peek2\_value**
  - uint16\_t **memory\_crc16**
  - uint16\_t **status\_register**
  - uint32\_t **bootloader\_version**
  - uint8\_t **memory\_dump** [UINT16\_MAX]
  - uint32\_t **device\_id**
  - uint8\_t **binary\_blob** [2048]
  - char **version\_string\_1** [1024]
  - char **version\_string\_2** [1024]
  - char **version\_string\_3** [1024]
  - char **label\_0** [DEVICE\_LABEL\_SIZE]
  - char **label\_1** [DEVICE\_LABEL\_SIZE]
  - char **label\_2** [DEVICE\_LABEL\_SIZE]
  - char **label\_3** [DEVICE\_LABEL\_SIZE]

```

uint8_t mode
uint8_t dock_status
uint8_t battery_charge_status
uint16_t battery_voltage
uint64_t sync_value
uint8_t off_reason
uint32_t uptime_get_value
uint32_t last_uptime_value
uint32_t last_standby_uptime_value
uint32_t config_get_value
uint8_t config_status
uint32_t error_count
char error_name [1024]
uint16_t error_log_size
uint16_t error_log_get_error_id
uint16_t error_stats_size
uint16_t error_stats_get_count
uint16_t stats_size
uint16_t stats_max_value
uint16_t stats_min_value
uint16_t stats_count_value
uint32_t stats_sum_value
uint32_t flash_block_get_value
struct {
    uint16_t year
    uint8_t month
    uint8_t day
    uint8_t hour
    uint8_t min
    uint8_t sec
} time_get
uint32_t calibration_version
uint32_t debug_get_value
uint32_t protocol_version
apdm_calibration_data_t calibration_data
uint32_t hw_id
char case_id [16]
uint8_t sample_get [2048]
} response_data

```

- uint16\_t **crc16**

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

- apdm\_internal.h

## 6.35 apdm\_streaming\_config\_t Struct Reference

### Public Attributes

- uint8\_t **wireless\_channel\_number**
- bool [enable\\_sd\\_card](#)
- bool [erase\\_sd\\_card](#)
- bool [accel\\_full\\_scale\\_mode](#)
- bool [enable\\_accel](#)
- bool [enable\\_gyro](#)
- bool [enable\\_mag](#)
- bool [apply\\_new\\_sensor\\_modes](#)
- bool [set\\_configuration\\_on\\_device](#)
- apdm\_monitor\_decimation\_rate\_t [decimation\\_rate](#)
- apdm\_monitor\_output\_select\_rate\_t **output\_select\_rate**
- bool [button\\_enable](#)
- [apdm\\_device\\_info\\_t](#) [device\\_info\\_cache](#) [APDM\_MAX\_NUMBER\_OF\_SENSORS]

### 6.35.1 Member Data Documentation

#### 6.35.1.1 bool apdm\_streaming\_config\_t::accel\_full\_scale\_mode

Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

Referenced by `apdm_configure_all_attached_sensors()`, and `apdm_init_streaming_config()`.

#### 6.35.1.2 bool apdm\_streaming\_config\_t::apply\_new\_sensor\_modes

Enable the magnetometers

Referenced by `apdm_init_streaming_config()`.

#### 6.35.1.3 bool apdm\_streaming\_config\_t::button\_enable

select output rate

Referenced by `apdm_init_streaming_config()`.

**6.35.1.4 apdm\_monitor\_decimation\_rate\_t apdm\_streaming\_config\_t::decimation\_rate**

Allows you to disable the setting of the configuration on the device so that it can be done later in a threaded/concurrent manor by the application.

Referenced by apdm\_init\_streaming\_config().

**6.35.1.5 apdm\_device\_info\_t apdm\_streaming\_config\_t::device\_info\_cache[APDM\_MAX\_NUMBER\_OF\_SENSORS]**

Enable monitor button accessory

Referenced by apdm\_streaming\_config\_get\_device\_info().

**6.35.1.6 bool apdm\_streaming\_config\_t::enable\_accel**

If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode

Referenced by apdm\_configure\_all\_attached\_sensors(), and apdm\_init\_streaming\_config().

**6.35.1.7 bool apdm\_streaming\_config\_t::enable\_gyro**

Enable the accelerometers

Referenced by apdm\_configure\_all\_attached\_sensors(), and apdm\_init\_streaming\_config().

**6.35.1.8 bool apdm\_streaming\_config\_t::enable\_mag**

Enable the gyros

Referenced by apdm\_configure\_all\_attached\_sensors(), and apdm\_init\_streaming\_config().

**6.35.1.9 bool apdm\_streaming\_config\_t::enable\_sd\_card**

The base wireless channel number to use

Referenced by apdm\_configure\_all\_attached\_sensors(), and apdm\_init\_streaming\_config().

#### 6.35.1.10 `bool apdm_streaming_config_t::erase_sd_card`

Boolean indicating weather or not data should be logged to the SD card on the device.

Referenced by `apdm_configure_all_attached_sensors()`, and `apdm_init_streaming_config()`.

#### 6.35.1.11 `bool apdm_streaming_config_t::set_configuration_on_device`

If set to true, flags are carried thru to the `device_info` structure

Referenced by `apdm_init_streaming_config()`.

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

- `apdm_types.h`

### 6.36 `apdm_ukf_state_t` Struct Reference

#### Public Attributes

- `int nStates`
- `int nMeasurements`
- `double * x`
- `double * z`
- `double * zf`
- `double * Q`
- `double * R`
- `double * P`
- `int(* f)(double *, double *, void *)`
- `int(* h)(double *, double *, void *)`

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

- `kalman_filter.h`

### 6.37 `apdm_usb_device_list_t` Struct Reference

#### Public Attributes

- `libusb_device ** deviceListPtr`

- int **num\_elements**
- [apdm\\_usb\\_sorted\\_device\\_element\\_t](#) \* **sorted\_element\_list**

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

- apdm\_usb.h

## 6.38 apdm\_usb\_sorted\_device\_element\_t Struct Reference

### Public Attributes

- uint16\_t **vid**
- uint16\_t **pid**
- uint32\_t **bus\_number**
- uint32\_t **device\_address**
- uint16\_t **bcd\_device**
- int **iSerialNumber**
- int **libusb\_index\_number**

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

- apdm\_usb.h

## 6.39 calibration\_v4\_t Struct Reference

### Public Attributes

- double **accl\_x\_bias**
- double [accl\\_y\\_bias](#)
- double [accl\\_z\\_bias](#)
- double [accl\\_x\\_bias\\_temp](#)
- double [accl\\_y\\_bias\\_temp](#)
- double [accl\\_z\\_bias\\_temp](#)
- double [accl\\_z\\_bias\\_dtemp](#)
- double [accl\\_x\\_scale](#)
- double [accl\\_y\\_scale](#)
- double [accl\\_z\\_scale](#)
- double [accl\\_x\\_scale\\_temp](#)

- double [accl\\_y\\_scale\\_temp](#)
- double [accl\\_z\\_scale\\_temp](#)
- double [accl\\_xy\\_sensitivity](#)
- double [accl\\_xz\\_sensitivity](#)
- double [accl\\_yz\\_sensitivity](#)
- double [accl\\_error\\_matrix](#) [3 \*3]
- double **gyro\_x\_bias**
- double [gyro\\_y\\_bias](#)
- double [gyro\\_z\\_bias](#)
- double [gyro\\_x\\_bias\\_temp](#)
- double [gyro\\_x\\_bias\\_temp2](#)
- double [gyro\\_y\\_bias\\_temp](#)
- double [gyro\\_y\\_bias\\_temp2](#)
- double [gyro\\_z\\_bias\\_temp](#) [61]
- double [gyro\\_x\\_scale](#)
- double [gyro\\_y\\_scale](#)
- double [gyro\\_z\\_scale](#)
- double [gyro\\_x\\_scale\\_temp](#)
- double [gyro\\_y\\_scale\\_temp](#)
- double [gyro\\_z\\_scale\\_temp](#)
- double [gyro\\_xy\\_sensitivity](#)
- double [gyro\\_xz\\_sensitivity](#)
- double [gyro\\_yz\\_sensitivity](#)
- double [gyro\\_accl\\_roll](#)
- double [gyro\\_accl\\_pitch](#)
- double [gyro\\_accl\\_yaw](#)
- double [gyro\\_error\\_matrix](#) [3 \*3]
- double **mag\_x\_bias**
- double [mag\\_y\\_bias](#)
- double [mag\\_z\\_bias](#)
- double [mag\\_x\\_scale](#)
- double **mag\_y\_scale**
- double **mag\_z\_scale**
- double **mag\_xy\_sensitivity**
- double **mag\_xz\_sensitivity**
- double **mag\_yz\_sensitivity**
- double **mag\_accl\_roll**
- double **mag\_accl\_pitch**
- double **mag\_accl\_yaw**
- double **mag\_error\_matrix** [3 \*3]
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_x\_state**
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_y\_state**
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_z\_state**



- double [temperature\\_bias](#)
- double [temperature\\_scale](#)
- double [temperature\\_bias\\_msp](#)
- double [temperature\\_scale\\_msp](#)



## 6.39.1 Member Data Documentation

6.39.1.1 double calibration\_v4\_t::accl\_error\_matrix[3 \*3]

6.39.1.2 double calibration\_v4\_t::accl\_x\_bias\_temp

6.39.1.3 double calibration\_v4\_t::accl\_x\_scale

6.39.1.4 double calibration\_v4\_t::accl\_x\_scale\_temp

6.39.1.5 double calibration\_v4\_t::accl\_xy\_sensitivity

6.39.1.6 double calibration\_v4\_t::accl\_xz\_sensitivity

6.39.1.7 double calibration\_v4\_t::accl\_y\_bias

6.39.1.8 double calibration\_v4\_t::accl\_y\_bias\_temp

6.39.1.9 double calibration\_v4\_t::accl\_y\_scale

6.39.1.10 double calibration\_v4\_t::accl\_y\_scale\_temp

6.39.1.11 double calibration\_v4\_t::accl\_yz\_sensitivity

6.39.1.12 double calibration\_v4\_t::accl\_z\_bias

6.39.1.13 double calibration\_v4\_t::accl\_z\_bias\_dtemp

6.39.1.14 double calibration\_v4\_t::accl\_z\_bias\_temp

6.39.1.15 double calibration\_v4\_t::accl\_z\_scale

6.39.1.16 double calibration\_v4\_t::accl\_z\_scale\_temp

6.39.1.17 double calibration\_v4\_t::gyro\_accl\_pitch

6.39.1.18 double calibration\_v4\_t::gyro\_accl\_roll

6.39.1.19 double calibration\_v4\_t::gyro\_accl\_yaw

6.39.1.20 double calibration\_v4\_t::gyro\_error\_matrix[3 \*3]

6.39.1.21 double calibration\_v4\_t::gyro\_x\_bias\_temp

6.39.1.22 double calibration\_v4\_t::gyro\_x\_bias\_temp2

6.39.1.23 double calibration\_v4\_t::gyro\_x\_scale

6.39.1.24 double calibration\_v4\_t::gyro\_x\_scale\_temp

6.39.1.25 double calibration\_v4\_t::gyro\_xy\_sensitivity

6.39.1.26 double calibration\_v4\_t::gyro\_xz\_sensitivity

- `apdm_types.h`

## 6.40 `calibration_v5_t` Struct Reference

### Public Attributes

- `uint16_t` **`accl_x_bias`** [61]
- `uint16_t` `accl_y_bias` [61]
- `uint16_t` `accl_z_bias` [61]
- `double` `accl_z_bias_dtemp`
- `double` `accl_x_scale`
- `double` `accl_y_scale`
- `double` `accl_z_scale`
- `double` `accl_x_scale_temp`
- `double` `accl_y_scale_temp`
- `double` `accl_z_scale_temp`
- `double` `accl_xy_sensitivity`
- `double` `accl_xz_sensitivity`
- `double` `accl_yz_sensitivity`
- `double` `accl_error_matrix` [3 \*3]
- `uint16_t` **`gyro_x_bias`** [61]
- `uint16_t` `gyro_y_bias` [61]
- `uint16_t` `gyro_z_bias` [61]
- `double` `gyro_x_scale`
- `double` `gyro_y_scale`
- `double` `gyro_z_scale`
- `double` `gyro_x_scale_temp`
- `double` `gyro_y_scale_temp`
- `double` `gyro_z_scale_temp`
- `double` `gyro_xy_sensitivity`
- `double` `gyro_xz_sensitivity`
- `double` `gyro_yz_sensitivity`
- `double` `gyro_accl_roll`
- `double` `gyro_accl_pitch`
- `double` `gyro_accl_yaw`
- `double` `gyro_error_matrix` [3 \*3]
- `uint16_t` **`mag_x_bias`** [61]
- `uint16_t` `mag_y_bias` [61]
- `uint16_t` `mag_z_bias` [61]
- `double` `mag_x_scale`
- `double` **`mag_y_scale`**

- double **mag\_z\_scale**
- double **mag\_x\_scale\_temp**
- double **mag\_y\_scale\_temp**
- double **mag\_z\_scale\_temp**
- double **mag\_xy\_sensitivity**
- double **mag\_xz\_sensitivity**
- double **mag\_yz\_sensitivity**
- double **mag\_accl\_roll**
- double **mag\_accl\_pitch**
- double **mag\_accl\_yaw**
- double **mag\_x\_offset**
- double **mag\_y\_offset**
- double **mag\_z\_offset**
- double **mag\_conversion\_gain**
- double **mag\_error\_matrix** [3 \*3]
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_x\_state**
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_y\_state**
- [apdm\\_mag\\_dechop\\_state\\_t](#) **mag\_z\_state**
- double [temperature\\_bias](#)
- double [temperature\\_scale](#)
- double [temperature\\_bias\\_msp](#)
- double [temperature\\_scale\\_msp](#)

### 6.40.1 Member Data Documentation

6.40.1.1 double calibration\_v5\_t::accl\_error\_matrix[3 \*3]

6.40.1.2 double calibration\_v5\_t::accl\_x\_scale

6.40.1.3 double calibration\_v5\_t::accl\_x\_scale\_temp

6.40.1.4 double calibration\_v5\_t::accl\_xy\_sensitivity

6.40.1.5 double calibration\_v5\_t::accl\_xz\_sensitivity

6.40.1.6 uint16\_t calibration\_v5\_t::accl\_y\_bias[61]

Temperature dependant bias covering the range [-10,50] C

**6.40.1.7 double calibration\_v5\_t::accl\_y\_scale**

**6.40.1.8 double calibration\_v5\_t::accl\_y\_scale\_temp**

**6.40.1.9 double calibration\_v5\_t::accl\_yz\_sensitivity**

**6.40.1.10 uint16\_t calibration\_v5\_t::accl\_z\_bias[61]**

Temperature dependant bias covering the range [-10,50] C

**6.40.1.11 double calibration\_v5\_t::accl\_z\_bias\_dtemp**

Temperature dependant bias covering the range [-10,50] C

**6.40.1.12 double calibration\_v5\_t::accl\_z\_scale**

**6.40.1.13 double calibration\_v5\_t::accl\_z\_scale\_temp**

**6.40.1.14 double calibration\_v5\_t::gyro\_accl\_pitch**

**6.40.1.15 double calibration\_v5\_t::gyro\_accl\_roll**

**6.40.1.16 double calibration\_v5\_t::gyro\_accl\_yaw**

**6.40.1.17 double calibration\_v5\_t::gyro\_error\_matrix[3 \* 3]**

**6.40.1.18 double calibration\_v5\_t::gyro\_x\_scale**

**6.40.1.19 double calibration\_v5\_t::gyro\_x\_scale\_temp**

**6.40.1.20 double calibration\_v5\_t::gyro\_xy\_sensitivity**

**6.40.1.21 double calibration\_v5\_t::gyro\_xz\_sensitivity**

**6.40.1.22 uint16\_t calibration\_v5\_t::gyro\_y\_bias[61]**

Referenced by `apdm_recalibrate_gyroscopes()`, and `apdm_recalibrate_gyroscopes_from_h5()`.

**6.40.1.23 double calibration\_v5\_t::gyro\_y\_scale**

**6.40.1.24 double calibration\_v5\_t::gyro\_y\_scale\_temp**

**6.40.1.25 double calibration\_v5\_t::gyro\_yz\_sensitivity**

**6.40.1.26 uint16\_t calibration\_v5\_t::gyro\_z\_bias[61]**

Referenced by `apdm_recalibrate_gyroscopes()`, and `apdm_recalibrate_gyroscopes_from_h5()`.

**6.40.1.27 double calibration\_v5\_t::gyro\_z\_scale**

**6.40.1.28 double calibration\_v5\_t::gyro\_z\_scale\_temp**

**6.40.1.29 double calibration\_v5\_t::mag\_x\_scale**

Referenced by `apdm_recalibrate_magnetometers()`.

**6.40.1.30 uint16\_t calibration\_v5\_t::mag\_y\_bias[61]**

Referenced by `apdm_recalibrate_magnetometers()`.

**6.40.1.31 apdm\_mag\_dechop\_state\_t calibration\_v5\_t::mag\_y\_state**

**6.40.1.32 uint16\_t calibration\_v5\_t::mag\_z\_bias[61]**

Referenced by `apdm_recalibrate_magnetometers()`.

**6.40.1.33 apdm\_mag\_dechop\_state\_t calibration\_v5\_t::mag\_z\_state**

**6.40.1.34 double calibration\_v5\_t::temperature\_bias**

**6.40.1.35 double calibration\_v5\_t::temperature\_bias\_msp**

**6.40.1.36 double calibration\_v5\_t::temperature\_scale**

**6.40.1.37 double calibration\_v5\_t::temperature\_scale\_msp**

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

- [apdm\\_types.h](#)

## 6.41 `per_device_info_t` Struct Reference

### Public Attributes

- [apdm\\_sensor\\_device\\_handle\\_t](#) **sensor\_handle**
- [apdm\\_sensor\\_compensation\\_t](#) **compensation\_data**
- `uint32_t` **device\_last\_sync\_val\_received**
- `uint32_t` **total\_samples\_received**
- [apdm\\_device\\_state\\_data\\_t](#) **discovered\_device\_id\_data**
- `uint32_t` **user\_meta\_data\_uint32\_list**
- `uint8_t` **sensor\_has\_sync\_lock**
- `char` **user\_meta\_data\_strings** [USER\_META\_DATA\_STRING\_SIZE]
- [apdm\\_monitor\\_error\\_stat\\_t](#) **sensor\_error\_counts** [APDM\_MAX\_SENSOR\_ERROR\_COUNTERS]
- [apdm\\_disk\\_ll\\_t](#) \* **sensor\_sample\_list\_ptr**

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

- [apdm\\_internal.h](#)

## 6.42 `tekhex_t` Struct Reference

### Public Attributes

- `unsigned char *` **data**
- `unsigned char *` **data\_default**
- `unsigned int` **data\_size**

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

- [tekhex.h](#)

## 6.43 `WIRELESS_PACKET` Union Reference

### Public Attributes

- [wp\\_raw\\_t](#) **raw**



- [wp\\_sync\\_t](#) **sync**
- [wp\\_data\\_t](#) **data**
- [wp\\_event\\_t](#) **event**
- [WP\\_CONFIG](#) **config**
- [WP\\_CONFIG\\_ACK](#) **config\_ack**

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

- apdm\_l1\_ap.h

## 6.44 WP\_CONFIG Struct Reference

### Public Attributes

- [uint8\\_t](#) **type**
- [uint8\\_t](#) **cmd**
- [uint32\\_t](#) **device\_id**
- [uint8\\_t](#) **args** [26]

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

- apdm\_l1\_ap.h

## 6.45 WP\_CONFIG\_ACK Struct Reference

### Public Attributes

- [uint8\\_t](#) **type**
- [uint8\\_t](#) **error**
- [uint32\\_t](#) **device\_id**
- [uint8\\_t](#) **args** [26]

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

- apdm\_l1\_ap.h

## 6.46 WP\_DATA Struct Reference

### Public Attributes

- uint8\_t **type**
- uint8\_t **retrys**
- uint16\_t **flags**
- uint16\_t **mx**
- uint16\_t **my**
- uint16\_t **mz**
- uint16\_t **mc**
- uint16\_t **gx**
- uint16\_t **gy**
- uint16\_t **gz**
- uint16\_t **ax**
- uint16\_t **ay**
- uint16\_t **az**
- uint32\_t **opt\_data**
- uint32\_t **sync\_val**

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

- apdm\_l1\_ap.h

## 6.47 WP\_EVENT Struct Reference

### Public Attributes

- uint8\_t **type**
- uint8\_t **retrys**
- uint16\_t **event\_id**
- uint32\_t **data\_a**
- uint32\_t **data\_b**
- uint32\_t **data\_c**
- uint32\_t **data\_d**
- uint32\_t **data\_e**
- uint32\_t **data\_f**

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

- apdm\_l1\_ap.h

## 6.48 WP\_RAW Struct Reference

### Public Attributes

- `uint8_t type`
- `uint8_t data [31]`

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

- `apdm_l1_ap.h`

## 6.49 WP\_SYNC Struct Reference

### Public Attributes

- `uint8_t type`
- `uint8_t id`
- `uint32_t sync_time [2]`
- `uint16_t requested_device_state`
- `uint16_t max_latency`

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

- `apdm_l1_ap.h`