Examples

Table of Contents

1. Basic Connection Test . . . 1
2. List All Calendars . . . 2
3. Fetch Events from Calendar . . . 3
4. Upcoming Events This Month . . . 4
5. Complete Application Example . . . 5
6. Error Handling Example . . . 8
7. Tips for Production Use . . . 10
    7.1. 1. Use Environment Variables or Secure Storage . . . 10
    7.2. 2. Implement Exponential Backoff . . . 11
    7.3. 3. Monitor Memory Usage . . . 12

1. Basic Connection Test

This example demonstrates how to initialize the client and test the connection to a CalDAV server.

```
#include "caldav_client.h"
#include <esp_log.h>

static const char *TAG = "CalDAV-Test";

void test_connection(void)
{
    // Configure client
    CalDAV_Config_t config = {
        .ServerURL = "https://cloud.example.com/remote.php/dav/calendars/john",
        .Username = "john",
        .Password = "secure_password_123",
        .TimeoutMs = 10000
    };

    // Initialize client
    CalDAV_Client_t *client = CalDAV_Client_Init(&config);
    if (client == NULL) {
        ESP_LOGE(TAG, "Failed to initialize CalDAV client");
        return;
    }

    // Test connection
    CalDAV_Error_t err = CalDAV_Test_Connection(client);
    if (err == CALDAV_ERROR_OK) {
        ESP_LOGI(TAG, "✓ Connection successful!");
```

```
    } else {
        ESP_LOGE(TAG, "❌ Connection failed with error: %d", err);
    }

    // Clean up
    CalDAV_Client_Deinit(client);
}
```

2. List All Calendars

Retrieve and display all available calendars from the server.

```
#include "caldav_client.h"
#include <esp_log.h>

static const char *TAG = "CalDAV-List";

void list_calendars(CalDAV_Client_t *client)
{
    CalDAV_Calendar_List_t calendars;

    CalDAV_Error_t err = CalDAV_Calendars_List(client, &calendars);
    if (err != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "Failed to list calendars: %d", err);
        return;
    }

    ESP_LOGI(TAG, "Found %d calendar(s):", calendars.Length);

    for (size_t i = 0; i < calendars.Length; i++) {
        CalDAV_Calendar_t *cal = &calendars.Calendar[i];

        ESP_LOGI(TAG, "\nCalendar %d:", i + 1);
        ESP_LOGI(TAG, "  Name: %s", cal->Name ? cal->Name : "N/A");
        ESP_LOGI(TAG, "  Display Name: %s",
                 cal->DisplayName ? cal->DisplayName : "N/A");
        ESP_LOGI(TAG, "  Path: %s", cal->Path ? cal->Path : "N/A");

        if (cal->Description) {
            ESP_LOGI(TAG, "  Description: %s", cal->Description);
        }

        if (cal->Color) {
            ESP_LOGI(TAG, "  Color: %s", cal->Color);
        }
    }

    // Free allocated memory
    CalDAV_Calendars_Free(&calendars);
```

```
}
```

3. Fetch Events from Calendar

Retrieve events from a specific calendar within a time range.

```
#include "caldav_client.h"
#include <esp_log.h>
#include <time.h>

static const char *TAG = "CalDAV-Events";

void fetch_events(CalDAV_Client_t *client, const char *calendar_path)
{
    CalDAV_Calendar_Event_t *events = NULL;
    size_t event_count = 0;

    // Fetch events for the year 2025
    CalDAV_Error_t err = CalDAV_Calendar_Events_List(
        client,
        &events,
        &event_count,
        calendar_path,
        "20250101T000000Z",  // Jan 1, 2025
        "20251231T235959Z"   // Dec 31, 2025
    );

    if (err != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "Failed to fetch events: %d", err);
        return;
    }

    ESP_LOGI(TAG, "Found %d event(s):", event_count);

    for (size_t i = 0; i < event_count; i++) {
        CalDAV_Calendar_Event_t *evt = &events[i];

        ESP_LOGI(TAG, "\nEvent %d:", i + 1);
        ESP_LOGI(TAG, "  Title: %s", evt->Summary ? evt->Summary : "N/A");
        ESP_LOGI(TAG, "  Start: %s", evt->StartTime ? evt->StartTime : "N/A");
        ESP_LOGI(TAG, "  End: %s", evt->EndTime ? evt->EndTime : "N/A");

        if (evt->Location) {
            ESP_LOGI(TAG, "  Location: %s", evt->Location);
        }

        if (evt->Description) {
            ESP_LOGI(TAG, "  Description: %s", evt->Description);
        }
```

```
        if (evt->UID) {
            ESP_LOGI(TAG, "  UID: %s", evt->UID);
        }
    }

    // Free allocated memory
    CalDAV_Events_Free(events, event_count);
}
```

4. Upcoming Events This Month

Get events for the current month.

```
#include "caldav_client.h"
#include <esp_log.h>
#include <time.h>
#include <stdio.h>

static const char *TAG = "CalDAV-Upcoming";

void get_current_month_events(CalDAV_Client_t *client, const char *calendar_path)
{
    time_t now = time(NULL);
    struct tm timeinfo;
    localtime_r(&now, &timeinfo);

    // Start of current month
    char start_time[20];
    snprintf(start_time, sizeof(start_time), "%04d%02d01T000000Z",
             timeinfo.tm_year + 1900, timeinfo.tm_mon + 1);

    // End of current month (approximate)
    char end_time[20];
    int last_day = 31;
    if (timeinfo.tm_mon == 1) { // February
        last_day = 28; // Simplified, doesn't handle leap years
    } else if (timeinfo.tm_mon == 3 || timeinfo.tm_mon == 5 ||
               timeinfo.tm_mon == 8 || timeinfo.tm_mon == 10) {
        last_day = 30;
    }

    snprintf(end_time, sizeof(end_time), "%04d%02d%02dT235959Z",
             timeinfo.tm_year + 1900, timeinfo.tm_mon + 1, last_day);

    ESP_LOGI(TAG, "Fetching events from %s to %s", start_time, end_time);

    CalDAV_Calendar_Event_t *events = NULL;
    size_t event_count = 0;
```

```
    CalDAV_Error_t err = CalDAV_Calendar_Events_List(
        client,
        &events,
        &event_count,
        calendar_path,
        start_time,
        end_time
    );

    if (err != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "Failed to fetch events: %d", err);
        return;
    }

    ESP_LOGI(TAG, "This month's events: %d", event_count);

    for (size_t i = 0; i < event_count; i++) {
        ESP_LOGI(TAG, "  %s", events[i].Summary ? events[i].Summary : "Untitled");
    }

    CalDAV_Events_Free(events, event_count);
}
```

5. Complete Application Example

Full application with Wi-Fi connection and calendar synchronization.

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "caldav_client.h"

#define WIFI_SSID      "YourWiFiSSID"
#define WIFI_PASSWORD  "YourWiFiPassword"

#define CALDAV_SERVER  "https://cloud.example.com/remote.php/dav/calendars/user"
#define CALDAV_USER    "user"
#define CALDAV_PASS    "password"

static const char *TAG = "CalDAV-App";
static EventGroupHandle_t wifi_event_group;
```

```
static const int WIFI_CONNECTED_BIT = BIT0;

// Wi-Fi event handler
static void wifi_event_handler(void *arg, esp_event_base_t event_base,
                               int32_t event_id, void *event_data)
{
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED) {
        ESP_LOGI(TAG, "Disconnected from AP, retrying...");
        esp_wifi_connect();
    } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
        ip_event_got_ip_t *event = (ip_event_got_ip_t *)event_data;
        ESP_LOGI(TAG, "Got IP: " IPSTR, IP2STR(&event->ip_info.ip));
        xEventGroupSetBits(wifi_event_group, WIFI_CONNECTED_BIT);
    }
}

// Initialize Wi-Fi
static void wifi_init(void)
{
    wifi_event_group = xEventGroupCreate();

    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID,
                                               &wifi_event_handler, NULL));
    ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP,
                                               &wifi_event_handler, NULL));

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASSWORD,
        },
    };

    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI(TAG, "Waiting for Wi-Fi connection...");
    xEventGroupWaitBits(wifi_event_group, WIFI_CONNECTED_BIT, false, true,
portMAX_DELAY);
}
```

```c
// CalDAV synchronization task
static void caldav_sync_task(void *pvParameters)
{
    CalDAV_Config_t config = {
        .ServerURL = CALDAV_SERVER,
        .Username = CALDAV_USER,
        .Password = CALDAV_PASS,
        .TimeoutMs = 15000
    };

    CalDAV_Client_t *client = CalDAV_Client_Init(&config);
    if (client == NULL) {
        ESP_LOGE(TAG, "Failed to initialize CalDAV client");
        vTaskDelete(NULL);
        return;
    }

    while (1) {
        ESP_LOGI(TAG, "Starting calendar synchronization...");

        // Test connection
        if (CalDAV_Test_Connection(client) != CALDAV_ERROR_OK) {
            ESP_LOGE(TAG, "Connection test failed");
            vTaskDelay(pdMS_TO_TICKS(60000)); // Retry after 1 minute
            continue;
        }

        // List calendars
        CalDAV_Calendar_List_t calendars;
        if (CalDAV_Calendars_List(client, &calendars) == CALDAV_ERROR_OK) {
            ESP_LOGI(TAG, "Found %d calendars", calendars.Length);

            // Fetch events from each calendar
            for (size_t i = 0; i < calendars.Length; i++) {
                CalDAV_Calendar_Event_t *events = NULL;
                size_t event_count = 0;

                ESP_LOGI(TAG, "Fetching events from: %s",
                         calendars.Calendar[i].DisplayName
                         ? calendars.Calendar[i].DisplayName
                         : calendars.Calendar[i].Name);

                CalDAV_Error_t err = CalDAV_Calendar_Events_List(
                    client,
                    &events,
                    &event_count,
                    calendars.Calendar[i].Path,
                    "20250101T000000Z",
                    "20251231T235959Z"
                );
```

```
                if (err == CALDAV_ERROR_OK) {
                    ESP_LOGI(TAG, "  Found %d events", event_count);

                    // Process events here (display, store, etc.)
                    for (size_t j = 0; j < event_count; j++) {
                        ESP_LOGI(TAG, "    - %s",
                                 events[j].Summary ? events[j].Summary : "Untitled");
                    }

                    CalDAV_Events_Free(events, event_count);
                } else {
                    ESP_LOGE(TAG, "  Failed to fetch events: %d", err);
                }
            }

            CalDAV_Calendars_Free(&calendars);
        }

        ESP_LOGI(TAG, "Synchronization complete. Waiting 5 minutes...");
        vTaskDelay(pdMS_TO_TICKS(300000)); // Sync every 5 minutes
    }

    CalDAV_Client_Deinit(client);
    vTaskDelete(NULL);
}

void app_main(void)
{
    // Initialize NVS
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);

    ESP_LOGI(TAG, "CalDAV Client Application Starting");

    // Connect to Wi-Fi
    wifi_init();

    // Start CalDAV sync task
    xTaskCreate(caldav_sync_task, "caldav_sync", 8192, NULL, 5, NULL);
}
```

6. Error Handling Example

Comprehensive error handling for robust applications.

```c
#include "caldav_client.h"
#include <esp_log.h>

static const char *TAG = "CalDAV-Robust";

const char *caldav_error_to_string(CalDAV_Error_t err)
{
    switch (err) {
        case CALDAV_ERROR_OK:              return "Success";
        case CALDAV_ERROR_INVALID_ARG:     return "Invalid Argument";
        case CALDAV_ERROR_NO_MEM:          return "Out of Memory";
        case CALDAV_ERROR_FAIL:            return "General Failure";
        case CALDAV_ERROR_NOT_INITIALIZED: return "Not Initialized";
        case CALDAV_ERROR_CONNECTION:      return "Connection Error";
        case CALDAV_ERROR_HTTP:            return "HTTP Error";
        case CALDAV_ERROR_TIMEOUT:         return "Timeout";
        default:                           return "Unknown Error";
    }
}

void robust_calendar_fetch(void)
{
    CalDAV_Config_t config = {
        .ServerURL = "https://cloud.example.com/remote.php/dav/calendars/user",
        .Username = "user",
        .Password = "password",
        .TimeoutMs = 10000
    };

    CalDAV_Client_t *client = CalDAV_Client_Init(&config);
    if (client == NULL) {
        ESP_LOGE(TAG, "Failed to initialize client");
        return;
    }

    // Test connection with retries
    int max_retries = 3;
    CalDAV_Error_t err;

    for (int i = 0; i < max_retries; i++) {
        err = CalDAV_Test_Connection(client);
        if (err == CALDAV_ERROR_OK) {
            break;
        }

        ESP_LOGW(TAG, "Connection attempt %d failed: %s",
                 i + 1, caldav_error_to_string(err));

        if (i < max_retries - 1) {
            vTaskDelay(pdMS_TO_TICKS(2000)); // Wait 2 seconds before retry
```

```
        }
    }

    if (err != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "All connection attempts failed");
        CalDAV_Client_Deinit(client);
        return;
    }

    // Fetch calendars with error handling
    CalDAV_Calendar_List_t calendars;
    err = CalDAV_Calendars_List(client, &calendars);

    if (err != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "Failed to list calendars: %s", caldav_error_to_string(err));
        CalDAV_Client_Deinit(client);
        return;
    }

    if (calendars.Length == 0) {
        ESP_LOGW(TAG, "No calendars found");
        CalDAV_Calendars_Free(&calendars);
        CalDAV_Client_Deinit(client);
        return;
    }

    // Process calendars...
    ESP_LOGI(TAG, "Successfully retrieved %d calendars", calendars.Length);

    // Clean up
    CalDAV_Calendars_Free(&calendars);
    CalDAV_Client_Deinit(client);
}
```

7. Tips for Production Use

7.1. 1. Use Environment Variables or Secure Storage

Never hardcode credentials in your source code:

```
#include "nvs_flash.h"
#include "nvs.h"

esp_err_t load_caldav_config(CalDAV_Config_t *config)
{
    nvs_handle_t nvs_handle;
    esp_err_t err;
```

```
    err = nvs_open("caldav", NVS_READONLY, &nvs_handle);
    if (err != ESP_OK) return err;

    size_t server_len, user_len, pass_len;

    // Get sizes
    nvs_get_str(nvs_handle, "server", NULL, &server_len);
    nvs_get_str(nvs_handle, "user", NULL, &user_len);
    nvs_get_str(nvs_handle, "pass", NULL, &pass_len);

    // Allocate and read
    char *server = malloc(server_len);
    char *user = malloc(user_len);
    char *pass = malloc(pass_len);

    nvs_get_str(nvs_handle, "server", server, &server_len);
    nvs_get_str(nvs_handle, "user", user, &user_len);
    nvs_get_str(nvs_handle, "pass", pass, &pass_len);

    config->ServerURL = server;
    config->Username = user;
    config->Password = pass;
    config->TimeoutMs = 10000;

    nvs_close(nvs_handle);
    return ESP_OK;
}
```

7.2. 2. Implement Exponential Backoff

For network errors, use exponential backoff:

```
int retry_delay_ms = 1000;
const int max_delay_ms = 60000;

for (int retry = 0; retry < max_retries; retry++) {
    err = CalDAV_Test_Connection(client);
    if (err == CALDAV_ERROR_OK) break;

    vTaskDelay(pdMS_TO_TICKS(retry_delay_ms));
    retry_delay_ms = (retry_delay_ms * 2 < max_delay_ms)
                     ? retry_delay_ms * 2
                     : max_delay_ms;
}
```

7.3. 3. Monitor Memory Usage

```
void log_memory_stats(void)
{
    ESP_LOGI(TAG, "Free heap: %d bytes", esp_get_free_heap_size());
    ESP_LOGI(TAG, "Min free heap: %d bytes", esp_get_minimum_free_heap_size());
}
```