ESP32 CalDAV Client

Table of Contents

1. Introduction

The ESP32 CalDAV Client is a lightweight CalDAV protocol implementation for ESP32 microcontrollers. It enables ESP32-based devices to communicate with CalDAV servers (such as Nextcloud, ownCloud, Radicale, etc.) to retrieve calendar information and events.

1.1. Features

- **CalDAV Protocol Support**: Implements core CalDAV protocol operations
- **SSL/TLS Support**: Secure connections using ESP-IDF certificate bundle
- **Calendar Discovery**: Automatically discover available calendars on the server
- **Event Retrieval**: Fetch events from calendars with time-range filtering
- **Authentication**: HTTP Basic Authentication support
- **Memory Efficient**: Optimized for ESP32 resource constraints
- **Comprehensive API**: Easy-to-use C/C++ API with clear error handling

1.2. Supported Servers

The library has been tested with the following CalDAV servers:

- Nextcloud
- ownCloud
- Radicale
- Baikal

Other RFC 4791 compliant CalDAV servers should work as well.

2. Requirements

2.1. Hardware

- ESP32, ESP32-S2, ESP32-S3, or ESP32-C3 microcontroller
- Minimum 4 MB flash recommended
- Network connectivity (Wi-Fi or Ethernet)

2.2. Software

- ESP-IDF v4.4 or later
- CMake build system or PlatformIO

3. Installation

3.1. ESP-IDF Component

Add the component to your project's `components` directory or use the IDF Component Manager:

```
idf.py add-dependency "esp32-caldav"
```

Or manually add to your `idf_component.yml`:

```
dependencies:
  esp32-caldav:
    git: https://github.com/Kampi/ESP32-CalDAV.git
```

3.2. PlatformIO

Add to your `platformio.ini`:

```
lib_deps =
    https://github.com/Kampi/ESP32-CalDAV.git
```

4. Quick Start

4.1. Basic Example

Here's a minimal example to get started:

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

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

void app_main(void)
{
    // Initialize Wi-Fi first (not shown here)
    // ...

    // Configure CalDAV client
    CalDAV_Config_t config = {
        .ServerURL = "https://cloud.example.com/remote.php/dav/calendars/username",
        .Username = "username",
        .Password = "password",
        .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
    if (CalDAV_Test_Connection(client) != CALDAV_ERROR_OK) {
        ESP_LOGE(TAG, "Connection test failed");
        CalDAV_Client_Deinit(client);
        return;
    }

    ESP_LOGI(TAG, "Successfully connected to CalDAV server");

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

        for (size_t i = 0; i < calendars.Length; i++) {
            ESP_LOGI(TAG, "Calendar %d: %s", i + 1,
                     calendars.Calendar[i].DisplayName
                     ? calendars.Calendar[i].DisplayName
                     : calendars.Calendar[i].Name);
        }

        // Retrieve events from first calendar
        if (calendars.Length > 0) {
            CalDAV_Calendar_Event_t *events = NULL;
            size_t event_count = 0;
```

```
            CalDAV_Error_t err = CalDAV_Calendar_Events_List(
                client,
                &events,
                &event_count,
                calendars.Calendar[0].Path,
                "20250101T000000Z",  // Start time
                "20251231T235959Z"   // End time
            );

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

                for (size_t i = 0; i < event_count; i++) {
                    ESP_LOGI(TAG, "Event: %s", events[i].Summary);
                    ESP_LOGI(TAG, "  Start: %s", events[i].StartTime);
                    ESP_LOGI(TAG, "  End: %s", events[i].EndTime);
                }

                CalDAV_Events_Free(events, event_count);
            }
        }

        CalDAV_Calendars_Free(&calendars);
    }

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

5. API Reference

5.1. Data Types

5.1.1. CalDAV_Error_t

Error codes returned by library functions:

Error Code	Description
`CALDAV_ERROR_OK`	Operation completed successfully
`CALDAV_ERROR_INVALID_ARG`	Invalid argument provided
`CALDAV_ERROR_NO_MEM`	Out of memory
`CALDAV_ERROR_FAIL`	General failure
`CALDAV_ERROR_NOT_INITIALIZED`	CalDAV client not initialized

Error Code	Description
`CALDAV_ERROR_CONNECTION`	Network connection error
`CALDAV_ERROR_HTTP`	HTTP protocol error
`CALDAV_ERROR_TIMEOUT`	Operation timeout

5.1.2. CalDAV_Config_t

Client configuration structure:

```
typedef struct {
    const char *ServerURL;    // CalDAV server URL
    const char *Username;     // Username for authentication
    const char *Password;     // Password for authentication
    uint32_t TimeoutMs;       // Timeout in milliseconds
} CalDAV_Config_t;
```

Parameters:

- `ServerURL`: Full URL to the CalDAV server (must include protocol)
- `Username`: Account username
- `Password`: Account password
- `TimeoutMs`: HTTP request timeout in milliseconds (recommended: 10000)

5.1.3. CalDAV_Calendar_t

Calendar information structure:

```
typedef struct {
    char *Name;           // Calendar name
    char *Path;           // Calendar path on server
    char *DisplayName;    // Display name for UI
    char *Description;    // Calendar description (optional)
    char *Color;          // Calendar color (optional)
} CalDAV_Calendar_t;
```

5.1.4. CalDAV_Calendar_Event_t

Calendar event structure:

```
typedef struct {
    char *UID;            // Unique event identifier
    char *Summary;        // Event title/summary
    char *Description;    // Event description (optional)
    char *StartTime;      // Start time (iCalendar format)
```

```
    char *EndTime;        // End time (iCalendar format)
    char *Location;       // Event location (optional)
} CalDAV_Calendar_Event_t;
```

5.2. Functions

5.2.1. CalDAV_Client_Init

```
CalDAV_Client_t *CalDAV_Client_Init(const CalDAV_Config_t *p_Config);
```

Initializes a new CalDAV client with the provided configuration.

Parameters:

- `p_Config`: Pointer to configuration structure (must not be NULL)

Returns:

- Pointer to initialized client handle on success
- `NULL` on failure

Example:

```
CalDAV_Config_t config = {
    .ServerURL = "https://cloud.example.com/remote.php/dav/calendars/user",
    .Username = "user",
    .Password = "pass",
    .TimeoutMs = 10000
};

CalDAV_Client_t *client = CalDAV_Client_Init(&config);
if (client == NULL) {
    // Handle error
}
```

5.2.2. CalDAV_Client_Deinit

```
void CalDAV_Client_Deinit(CalDAV_Client_t *p_Client);
```

Deinitializes and frees resources associated with the CalDAV client.

Parameters:

- `p_Client`: CalDAV client handle

Example:

```
CalDAV_Client_Deinit(client);
```

5.2.3. CalDAV_Test_Connection

```
CalDAV_Error_t CalDAV_Test_Connection(CalDAV_Client_t *p_Client);
```

Tests the connection to the CalDAV server and validates credentials.

Parameters:

- `p_Client`: CalDAV client handle (must not be NULL)

Returns:

- `CALDAV_ERROR_OK`: Connection successful
- `CALDAV_ERROR_CONNECTION`: Network error
- `CALDAV_ERROR_HTTP`: Authentication failed or other HTTP error
- Other error codes as appropriate

Example:

```
if (CalDAV_Test_Connection(client) == CALDAV_ERROR_OK) {
    ESP_LOGI(TAG, "Connection successful");
} else {
    ESP_LOGE(TAG, "Connection failed");
}
```

5.2.4. CalDAV_Calendars_List

```
CalDAV_Error_t CalDAV_Calendars_List(CalDAV_Client_t *p_Client,
                                     CalDAV_Calendar_List_t *p_Calendars);
```

Retrieves a list of all available calendars from the server.

Parameters:

- `p_Client`: CalDAV client handle (must not be NULL)
- `p_Calendars`: Pointer to calendar list structure (will be populated)

Returns:

- `CALDAV_ERROR_OK`: Success
- Error code on failure

Note: Caller must free the calendar list using `CalDAV_Calendars_Free()`.

Example:

```
CalDAV_Calendar_List_t calendars;
if (CalDAV_Calendars_List(client, &calendars) == CALDAV_ERROR_OK) {
    for (size_t i = 0; i < calendars.Length; i++) {
        printf("Calendar: %s\n", calendars.Calendar[i].DisplayName);
    }
    CalDAV_Calendars_Free(&calendars);
}
```

5.2.5. CalDAV_Calendar_Events_List

```
CalDAV_Error_t CalDAV_Calendar_Events_List(CalDAV_Client_t *p_Client,
                                           CalDAV_Calendar_Event_t **p_Events,
                                           size_t *p_Length,
                                           const char *p_CalendarPath,
                                           const char *p_StartTime,
                                           const char *p_EndTime);
```

Retrieves events from a specific calendar within a time range.

Parameters:

- `p_Client`: CalDAV client handle (must not be NULL)
- `p_Events`: Pointer to event array pointer (will be allocated)
- `p_Length`: Pointer to store the number of events found
- `p_CalendarPath`: Path to the calendar (from `CalDAV_Calendar_t.Path`)
- `p_StartTime`: Start time in iCalendar format (e.g., "20250101T000000Z")
- `p_EndTime`: End time in iCalendar format (e.g., "20251231T235959Z")

Returns:

- `CALDAV_ERROR_OK`: Success
- Error code on failure

Note: Caller must free the event array using `CalDAV_Events_Free()`.

Example:

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

CalDAV_Error_t err = CalDAV_Calendar_Events_List(
    client,
```

```
    &events,
    &event_count,
    "/calendars/user/personal/",
    "20250101T000000Z",
    "20251231T235959Z"
);

if (err == CALDAV_ERROR_OK) {
    for (size_t i = 0; i < event_count; i++) {
        printf("%s: %s\n", events[i].Summary, events[i].StartTime);
    }
    CalDAV_Events_Free(events, event_count);
}
```

5.2.6. CalDAV_Calendars_Free

```
void CalDAV_Calendars_Free(CalDAV_Calendar_List_t *p_Calendars);
```

Frees memory allocated for calendar list.

Parameters:

- `p_Calendars`: Pointer to calendar list structure

Example:

```
CalDAV_Calendars_Free(&calendars);
```

5.2.7. CalDAV_Events_Free

```
void CalDAV_Events_Free(CalDAV_Calendar_Event_t *p_Events, size_t Length);
```

Frees memory allocated for event array.

Parameters:

- `p_Events`: Event array to free
- `Length`: Number of events in the array

Example:

```
CalDAV_Events_Free(events, event_count);
```

6. Configuration

6.1. Kconfig Options

The library provides several Kconfig options for customization:

```
CONFIG_ESP32_CALDAV_USEPSRAM
    Use PSRAM for memory allocation
    Default: n
```

To enable PSRAM support, add to your project's `sdkconfig`:

```
CONFIG_ESP32_CALDAV_USEPSRAM=y
```

7. Advanced Usage

7.1. Time Format

The library uses iCalendar time format for date/time specifications:

- **Basic format**: `YYYYMMDDTHHmmss` (local time)
- **UTC format**: `YYYYMMDDTHHmmssZ` (recommended)

Examples:

- `20250101T000000Z` = January 1, 2025, 00:00:00 UTC
- `20251231T235959Z` = December 31, 2025, 23:59:59 UTC
- `20250315T120000` = March 15, 2025, 12:00:00 local time

7.2. Server URL Format

The server URL format depends on your CalDAV server:

Nextcloud/ownCloud:

```
https://cloud.example.com/remote.php/dav/calendars/username
```

Radicale:

```
https://radicale.example.com/username
```

Baikal:

```
https://baikal.example.com/cal.php/calendars/username
```

7.3. Error Handling

Always check return values and handle errors appropriately:

```
CalDAV_Error_t err = CalDAV_Test_Connection(client);
switch (err) {
    case CALDAV_ERROR_OK:
        ESP_LOGI(TAG, "Connection successful");
        break;
    case CALDAV_ERROR_CONNECTION:
        ESP_LOGE(TAG, "Network connection failed");
        break;
    case CALDAV_ERROR_HTTP:
        ESP_LOGE(TAG, "Authentication failed or HTTP error");
        break;
    case CALDAV_ERROR_TIMEOUT:
        ESP_LOGE(TAG, "Request timeout");
        break;
    default:
        ESP_LOGE(TAG, "Unknown error: %d", err);
        break;
}
```

7.4. Memory Management

The library allocates memory dynamically for calendars and events. Always free allocated resources:

```
// After using calendars
CalDAV_Calendars_Free(&calendars);

// After using events
CalDAV_Events_Free(events, event_count);

// Before exiting application
CalDAV_Client_Deinit(client);
```

7.5. HTTPS Certificate Validation

The library uses ESP-IDF's certificate bundle for SSL/TLS verification. Ensure the certificate bundle is enabled in your project:

```
CONFIG_MBEDTLS_CERTIFICATE_BUNDLE=y
CONFIG_MBEDTLS_CERTIFICATE_BUNDLE_DEFAULT_FULL=y
```

8. Troubleshooting

8.1. Common Issues

8.1.1. Connection Timeout

Problem: `CALDAV_ERROR_TIMEOUT` errors

Solutions:

- Increase timeout value in `CalDAV_Config_t.TimeoutMs`
- Check network connectivity
- Verify server is reachable

8.1.2. Authentication Failed

Problem: HTTP 401 errors or `CALDAV_ERROR_HTTP`

Solutions:

- Verify username and password
- Check if two-factor authentication is enabled (may require app-specific password)
- Ensure the server URL is correct

8.1.3. No Calendars Found

Problem: `CalDAV_Calendars_List()` returns 0 calendars

Solutions:

- Verify the server URL points to the correct calendar home
- Check user permissions on the server
- Enable debug logging to inspect server response

8.1.4. Out of Memory

Problem: `CALDAV_ERROR_NO_MEM` errors

Solutions:

- Enable PSRAM support with `CONFIG_ESP32_CALDAV_USEPSRAM`
- Reduce the time range when fetching events

- Free resources promptly after use

8.2. Debug Logging

Enable verbose logging to troubleshoot issues:

```
esp_log_level_set("CalDAV-Client", ESP_LOG_DEBUG);
```

9. Examples

Complete examples can be found in the `examples/` directory:

- `basic_example.c` - Basic connection and calendar listing
- `event_sync.c` - Periodic event synchronization
- `display_events.c` - Display upcoming events on LCD

10. Performance Considerations

10.1. Memory Usage

Typical memory usage:

- Client structure: ~100 bytes
- Per calendar: ~200 bytes + string lengths
- Per event: ~300 bytes + string lengths

For systems with many events, consider:

- Using PSRAM for large allocations
- Fetching events in smaller time ranges
- Processing events in batches

10.2. Network Bandwidth

- Calendar list request: ~1-5 KB response
- Event query: varies by number of events (typical: 1-50 KB)

11. License

```
Copyright (C) Daniel Kampert, 2026
Website: www.kampis-elektroecke.de

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

12. Support

For bug reports, feature requests, and questions:

- Email: DanielKampert@kampis-elektroecke.de
- GitHub Issues: https://github.com/Kampi/ESP32-CalDAV/issues

13. References

- RFC 4791 - CalDAV: Calendaring Extensions to WebDAV
- RFC 5545 - iCalendar
- ESP-IDF Documentation