# FAQ - Frequently Asked Questions

## Table of Contents

# General Questions

## What is CalDAV?

CalDAV (Calendaring Extensions to WebDAV) is an Internet standard allowing clients to access calendar information on a remote server. It's defined in RFC 4791 and widely supported by calendar servers like Nextcloud, ownCloud, Google Calendar, and Apple iCloud.

## Why use CalDAV on ESP32?

CalDAV enables ESP32 devices to:

- Retrieve upcoming events for display on smart displays
- Trigger actions based on calendar events (home automation)
- Synchronize schedules with cloud services
- Create calendar-aware IoT devices

## Which CalDAV servers are supported?

The library works with any RFC 4791 compliant CalDAV server, including:

- Nextcloud
- ownCloud
- Radicale
- Baikal
- SOGo
- Apple Calendar Server
- Google Calendar (with CalDAV API)

# Configuration & Setup

## How do I find my CalDAV server URL?

The server URL format varies by provider:

**Nextcloud/ownCloud:**

```
https://your-server.com/remote.php/dav/calendars/username
```

**Radicale:**

```
https://your-server.com/username
```

**Baikal:**

```
https://your-server.com/cal.php/calendars/username
```

For other servers, consult their documentation or check the calendar app settings.

## What credentials do I need?

You need:

- Username (usually your login name or email)
- Password or app-specific password

| | |
|---|---|
| **NOTE** | If your server uses two-factor authentication (2FA), you'll need to generate an app-specific password. |

## How do I generate an app-specific password?

**Nextcloud:**

1. Go to Settings → Security
2. Scroll to "Devices & sessions"
3. Enter an app name and click "Create new app password"
4. Copy the generated password

**Google Calendar:**

1. Go to Google Account settings
2. Security → 2-Step Verification → App passwords
3. Generate a new app password for "Other (custom name)"

## Can I use a self-signed certificate?

The library uses ESP-IDF's certificate bundle by default. For self-signed certificates, you need to:

1. Disable certificate verification (not recommended for production)
2. Or add your certificate to the bundle
3. Or implement custom certificate validation

Example (disable verification - use only for testing):

```
// In caldav_client.cpp, modify HTTP config:
_CalDAV_HTTP_Config.skip_cert_common_name_check = true;
_CalDAV_HTTP_Config.crt_bundle_attach = NULL;
```

# Connection Issues

## I get CALDAV_ERROR_CONNECTION. What should I do?

Check these common issues:

1. **Network connectivity:** Ensure Wi-Fi is connected
2. **DNS resolution:** Verify the server hostname resolves correctly
3. **Firewall:** Check if firewall blocks outbound HTTPS (port 443)
4. **Server availability:** Test the URL in a web browser

Debug steps:

```
// Enable verbose logging
esp_log_level_set("CalDAV-Client", ESP_LOG_DEBUG);
esp_log_level_set("HTTP_CLIENT", ESP_LOG_DEBUG);
```

## I get CALDAV_ERROR_HTTP with status 401

This indicates authentication failure. Check:

- Username and password are correct
- Account is not locked
- Two-factor authentication app password is used (if enabled)
- Server URL points to the correct endpoint

## I get CALDAV_ERROR_TIMEOUT

The request is taking too long. Try:

- Increasing timeout: `config.TimeoutMs = 30000;` (30 seconds)
- Checking network latency
- Reducing the query time range for events
- Verifying server performance

## The connection works in a browser but not on ESP32

This usually indicates:

- Certificate validation issues
- Missing or incorrect certificate bundle
- Server using older TLS version not supported by ESP32

Enable certificate bundle in sdkconfig:

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

# Calendar & Event Queries

## No calendars are returned

If `CalDAV_Calendars_List()` returns 0 calendars:

1. Verify the server URL is correct
2. Check user has at least one calendar
3. Ensure user has permission to access calendars
4. Enable debug logging to inspect server response

## No events are returned

If `CalDAV_Calendar_Events_List()` returns 0 events:

1. Verify events exist in the specified time range
2. Check calendar path is correct (use path from `CalDAV_Calendars_List()`)
3. Ensure time format is correct (YYYYMMDDTHHmmssZ)
4. Try expanding the time range

## How do I query all future events?

Use a far future end date:

```
CalDAV_Calendar_Events_List(
    client,
    &events,
    &event_count,
    calendar_path,
    "20250101T000000Z",  // Current or past date
    "20501231T235959Z"   // Far future date
);
```

## Can I search for specific events?

The library doesn't support text search directly. Retrieve all events and filter locally:

```
for (size_t i = 0; i < event_count; i++) {
    if (events[i].Summary && strstr(events[i].Summary, "Meeting")) {
        // Found matching event
    }
}
```

## What date/time format is required?

Use iCalendar format:

- **UTC time:** `YYYYMMDDTHHmmssZ` (Z indicates UTC)
- **Local time:** `YYYYMMDDTHHmmss` (no Z suffix)

Examples:

```
"20250315T143000Z"  = March 15, 2025, 14:30:00 UTC
"20250101T000000Z"  = January 1, 2025, 00:00:00 UTC
```

## How do I handle recurring events?

CalDAV servers typically expand recurring events within the queried time range. Each occurrence appears as a separate event with its own start/end time.

## Events show weird times. Why?

The library returns times as received from the server. You may need to:

1. Parse the iCalendar format
2. Convert to local timezone
3. Handle `VALUE=DATE` for all-day events

Example parsing:

```
// For VALUE=DATE format (e.g., "20250210")
if (strlen(events[i].StartTime) == 8) {
    // All-day event
    int year, month, day;
    sscanf(events[i].StartTime, "%4d%2d%2d", &year, &month, &day);
}
```

# Memory Issues

## I get CALDAV_ERROR_NO_MEM errors

The system is out of memory. Solutions:

1. **Enable PSRAM:**

   ```
   CONFIG_ESP32_CALDAV_USEPSRAM=y
   ```

2. **Increase heap size:** Adjust partition table
3. **Reduce query range:** Fetch events in smaller time windows
4. **Free resources promptly:**

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

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

5. **Check for memory leaks:** Ensure all allocated resources are freed

## How much memory does the library use?

Typical usage:

- Client handle: ~100 bytes
- Per calendar: ~200 bytes + string data
- Per event: ~300 bytes + string data
- HTTP buffers: 4 KB (expandable)

For a calendar with 50 events averaging 100 bytes each, expect ~20 KB total.

## Can I limit memory usage?

Yes, several strategies:

1. Query shorter time ranges
2. Process events in batches and free between queries
3. Limit the number of calendars queried
4. Use PSRAM for large allocations

# Performance & Optimization

## How long does a typical query take?

Approximate times (depends on network and server):

- Connection test: 1-2 seconds
- List calendars: 2-5 seconds
- Fetch events (10-50 events): 3-10 seconds

## Can I make queries faster?

1. **Reduce timeout** (but risk timeouts on slow networks)
2. **Cache results** and refresh periodically
3. **Use narrower time ranges**
4. **Minimize calendar/event parsing**

## How often should I sync?

Depends on your use case:

- Smart display: Every 5-15 minutes
- Home automation: Every hour or event-triggered
- Status indicators: Every 30 minutes

Balance freshness needs with server load and power consumption.

## Should I use FreeRTOS tasks?

Yes, for periodic synchronization:

```
void caldav_sync_task(void *pvParameters)
{
    CalDAV_Client_t *client = (CalDAV_Client_t *)pvParameters;

    while (1) {
        // Sync calendars and events
        // ...

        vTaskDelay(pdMS_TO_TICKS(300000)); // 5 minutes
    }
}

xTaskCreate(caldav_sync_task, "caldav", 8192, client, 5, NULL);
```

# Security

## Is my password transmitted securely?

Yes, the library uses HTTPS (TLS/SSL) for all communications. Passwords are sent using HTTP Basic Authentication over the encrypted connection.

## Should I hardcode credentials?

No! Use secure storage:

1. **NVS (Non-Volatile Storage):** For encrypted credential storage
2. **Secure Boot:** Protect firmware from tampering
3. **Flash Encryption:** Encrypt the entire flash

## Can I use OAuth instead of passwords?

The current library version supports only HTTP Basic Authentication. OAuth support may be added in future versions.

For OAuth-enabled servers, generate an app-specific password as a workaround.

## How do I store credentials securely?

Example using NVS:

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

esp_err_t save_credentials(const char *user, const char *pass)
{
    nvs_handle_t handle;
    esp_err_t err = nvs_open("caldav", NVS_READWRITE, &handle);
    if (err != ESP_OK) return err;

    nvs_set_str(handle, "user", user);
    nvs_set_str(handle, "pass", pass);
    nvs_commit(handle);
    nvs_close(handle);

    return ESP_OK;
}
```

# Troubleshooting

## How do I enable debug logging?

```
#include <esp_log.h>

// In app_main() or before CalDAV calls:
esp_log_level_set("CalDAV-Client", ESP_LOG_DEBUG);
esp_log_level_set("HTTP_CLIENT", ESP_LOG_VERBOSE);
```

## The library crashes. What should I check?

1. **Stack overflow:** Increase task stack size
2. **Heap corruption:** Enable heap tracing
3. **Null pointers:** Check all returned pointers

4. **Memory leaks:** Ensure `_Free()` functions are called

Enable heap debugging:

```
CONFIG_HEAP_POISONING_COMPREHENSIVE=y
CONFIG_HEAP_TRACING=y
```

## Server returns HTML instead of XML

This usually means:

- Wrong server URL (hitting a web page instead of CalDAV endpoint)
- Server redirecting to login page
- Authentication required but not provided

The library detects HTML and returns `CALDAV_ERROR_HTTP`.

## Events are duplicated

Check if:

- You're querying overlapping time ranges
- The same event occurs in multiple calendars
- Server expands recurring events into the same period multiple times

## Strange characters in event titles

This indicates encoding issues:

1. Ensure server uses UTF-8 encoding
2. Check terminal/display supports UTF-8
3. Server may be sending escaped characters

# Development & Integration

## Can I use this with Arduino framework?

Yes! The library works with PlatformIO and Arduino-ESP32 framework:

```
platform = espressif32
framework = arduino
lib_deps =
    https://github.com/Kampi/ESP32-CalDAV.git
```

## How do I integrate with LVGL?

The library provides data; you display it with LVGL:

```
void display_events_on_lvgl(CalDAV_Calendar_Event_t *events, size_t count)
{
    lv_obj_t *list = lv_list_create(lv_scr_act());

    for (size_t i = 0; i < count; i++) {
        lv_obj_t *item = lv_list_add_btn(list, LV_SYMBOL_CALENDAR,
                                         events[i].Summary);
    }
}
```

## Can I modify the library?

Yes! The library is licensed under GPLv3:

- You can modify and redistribute
- You must keep the same license
- You must provide source code for modifications

## How do I contribute?

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Submit a pull request

Or report issues and suggestions via GitHub Issues.

## Is there example code?

Yes, check the `examples/` directory in the repository for:

- Basic connection test
- Calendar listing
- Event synchronization
- Full application examples

# Platform-Specific

## Does it work on ESP32-S2/S3/C3?

Yes, the library is compatible with:

- ESP32
- ESP32-S2
- ESP32-S3
- ESP32-C3

All variants with sufficient memory can run the library.

## What about memory-constrained devices?

For devices with limited RAM:

1. Enable PSRAM if available
2. Query smaller time ranges
3. Process one calendar at a time
4. Free resources immediately after use

Minimum recommended: 100 KB free heap

## Can I use it without Wi-Fi?

CalDAV requires network connectivity. Alternatives:

- Ethernet adapter
- LTE/cellular modem
- LoRaWAN (with custom gateway)

The library works with any network interface supported by ESP-IDF.

# Support & Resources

## Where can I get help?

- Email: DanielKampert@kampis-elektroecke.de
- GitHub Issues: https://github.com/Kampi/ESP32-CalDAV/issues
- ESP-IDF Forum: https://esp32.com

## Where is the source code?

GitHub: https://github.com/Kampi/ESP32-CalDAV

# How do I report bugs?

Open an issue on GitHub with:

1. ESP-IDF version
2. Hardware (ESP32 variant)
3. CalDAV server type and version
4. Complete error messages
5. Minimal code to reproduce

# Is commercial support available?

Contact DanielKampert@kampis-elektroecke.de for commercial support inquiries.