EtherNet/IP++ – Feature Documentation

Platform: Android (minSdk 26 / compileSdk 35)

Technology stack: Kotlin 2.0 · Jetpack Compose · Material 3 · Room · Coroutines

Protocol: EtherNet/IP (CIP Explicit Messaging, TCP Port 44818)

Table of Contents

1. App Overview & Getting Started

EtherNet/IP++ is a professional EtherNet/IP diagnostics and control app for Android. It implements the CIP Explicit Messaging protocol (ODVA standard) entirely in Kotlin without external libraries and is aimed at automation engineers, electricians, and developers.

Supported connection type:

Type	Description
EtherNet/IP (Explicit)	TCP connection, port 44818 (default)

Supported CIP services:

Service	Code	Description

Get Attribute Single	0x0E	Read a single attribute
Set Attribute Single	0x10	Write a single attribute
Get Attribute All	0x01	Read all attributes of an object
Read Tag	0x4C	Read Logix tag (Allen-Bradley ControlLogix/CompactLogix)
Write Tag	0x4D	Write Logix tag

App structure at launch:

The app opens the **Dashboard** as the start screen. All functions are accessible from there via the bottom navigation bar.

2. Navigation & Menu Structure

Navigation is provided via the **Bottom Navigation Bar** with five tabs, as well as Floating Action Buttons (FAB, "+" symbol) and icons in the top bar.

```
App launch
└── Dashboard (MainScreen)                    ← Bottom Nav: Home icon
    │   ├── [▶ Play]    → Start polling
    │   ├── [↻ Refresh] → One-time read
    │   └── [Tab select] → Switch device (with multiple devices)
    │
├── Devices (DeviceListScreen)                ← Bottom Nav: List icon
    │   ├── [+ FAB]     → New device (DeviceFormScreen)
    │   ├── [✏ Pencil] → Edit device (DeviceFormScreen)
    │   ├── [🗑 Trash]  → Delete device (confirmation dialog)
    │   └── [Card tap] → Device detail (DeviceDetailScreen)
    │       ├── [+ FAB]   → New tag (TagFormScreen)
    │       ├── [✏ Pencil] → Edit tag (TagFormScreen)
    │       └── [🗑 Trash] → Delete tag (confirmation dialog)
    │
├── Log (LogScreen)                           ← Bottom Nav: History icon
├── Alarms (AlarmScreen)                      ← Bottom Nav: Bell icon (badge)
└── Settings (SettingsScreen)                 ← Bottom Nav: Gear icon
    └── [Diagnostics button] → DiagScreen
```

3. Dashboard (MainScreen)

How to reach: Launch the app – Dashboard appears automatically.

The Dashboard shows all configured tags of the active device in real time. Values are displayed as cards.

3.1 Switch Device

Path: Dashboard → Tab bar (appears only with more than one device) → Tap desired tab

- When switching tabs, displayed values are reset
- The last active device is remembered for the session

3.2 One-Time Read

Path: Dashboard → ⟳ **button** (top right in the TopAppBar) → tap

Step	Control	Description
1	Device active in Dashboard	(Tab of the desired device selected)
2	Tap ⟳ button	All tags of the active device are read once
—	Tag cards	Values are updated immediately

3.3 Start/Stop Polling

Path: Dashboard → ▶ **button** (top right) → Polling active → ■ **button** to stop

Step	Control	Description
1	Tap ▶ button	Cyclic reading starts (interval from Settings)
—	Button changes to ■	Indicates active polling state
2	Tap ■ button	Polling stops, current session ends

While polling, values are automatically recorded to the database.

3.4 Error Messages

On connection problems, a **red banner** appears below the tab bar with the exact error message (e.g., "Connection refused", "Timeout").

3.5 Empty State

If no devices are configured yet, a hint text with a direct **"Add Device"** button is shown.

4. Device List (DeviceListScreen)

How to reach: Bottom Navigation → **List icon**

Shows all configured EtherNet/IP devices as cards. Each card shows the name, IP address, port, and number of configured tags.

4.1 Add New Device

Path: Device list → **"+"** button (bottom right) → DeviceFormScreen

4.2 Open Device (View Tags)

Path: Device list → Tap device card → DeviceDetailScreen

4.3 Edit Device

Path: Device list → ✏ **icon** (right on the device card) → DeviceFormScreen

4.4 Delete Device

Path: Device list → 🗑 **icon** (right on the device card) → Confirmation dialog → "Delete"

> *Deleting a device also removes all associated tags. Log recordings remain in the database.*

5. Device Detail (DeviceDetailScreen)

How to reach: Device list → Tap device card

Shows all configured tags of a device and allows their management.

5.1 Add Tag

Path: Device detail → **"+"** button (bottom right)

- In the **free version**, the FAB is locked (🔒 icon) once 5 tags are reached
- Tapping the locked FAB opens the **Pro upgrade dialog**

5.2 Tag Counter

The **counter** (e.g., `3 / 5`) in the top right of the device header shows progress:

- **Green background:** Space available
- **Red background:** Limit reached (free version)
- Pro version: No counter visible

5.3 Edit Tag

Path: Device detail → ✏ **icon** (right on the tag card) → TagFormScreen

5.4 Delete Tag

Path: Device detail → 🗑 **icon** (right on the tag card) → Confirmation dialog → "Delete"

5.5 Device Information

The device header below the TopAppBar shows:

- IP address and port
- Backplane slot (only if > 0)
- Description (if present)

6. Add/Edit Device (DeviceFormScreen)

How to reach (new): Device list → "+" button

How to reach (edit): Device list → ⋮ icon on device card

or: Device detail → ⋮ icon in the TopAppBar

Field	Required	Description
Name	✓	Device label
Description		Free-text note
IP Address	✓	IP address of the EtherNet/IP device
Port		TCP port (default: 44818)
Backplane Slot		Slot number in the chassis (0 = direct connection, no routing)

> ***Slot note:** For ControlLogix systems, the slot specifies which chassis slot the CPU is in. For directly addressable devices (e.g., drives, fieldbus adapters), leave slot at 0.*

Save: "Save" button (bottom) – only active when Name and IP are filled.

Discard: Back arrow (←) navigates back without saving.

6.2 Write NFC Tag

Path: Add/Edit Device → **NFC button** (chip icon, right of "Save") → Dialog "Hold device near an NFC tag" → tap tag

> *The NFC button only appears on devices with NFC hardware. It is only active when Name and IP are filled.*

Steps:

Step	Control	Description
1	Tap **NFC button**	Waiting dialog appears
2	Hold Android device near NFC tag	Tag is detected automatically
3	—	App writes profile to tag (NDEF, MIME `application/io.github.erginmusa.eip`)
4	—	Dialog closes; result banner appears (green = success, red = error)
—	**Cancel** button	Abort the write operation

Stored NFC profile:

```
name        – Device name
description – Description
ip          – IP address
port        – TCP port
slot        – Backplane slot
```

6.3 Read Device Profile from NFC Tag

When the app is in the foreground and a written tag is held near the device, the form fields are filled in automatically. This also works when the app is launched by tapping the tag.

7. Add/Edit Tag (TagFormScreen)

How to reach (new): Device detail → "+" button

How to reach (edit): Device detail → ⋮ icon on tag card

A **tag** represents a single CIP data point (attribute, Logix tag, etc.).

7.1 Fields

Field	Required	Description
Name	✓	Tag label
Description		Free-text note
CIP Path	✓	Addressing the data point (see 7.2)
CIP Service	✓	Read or write operation (dropdown)
Data Type	✓	How raw data is interpreted (dropdown)
Scale		Multiplier: display value = (raw × scale) + offset
Offset		Addition after scaling
Unit		Displayed text (e.g., °C, bar, rpm)
Warning threshold		Above this value → warning alarm
Alarm threshold		Above this value → critical alarm
Write value		Only visible for write services

7.2 CIP Path Syntax

Format	Example	Meaning
`Class.Instance.Attribute`	`1.1.1`	Identity Object, instance 1, attribute 1 (Vendor ID)

`Class.Instance`	`1.1`	Identity Object, all attributes (with Get Attribute All)
`TagName`	`MyMotorSpeed`	Logix symbolic tag (Allen-Bradley)

Common CIP objects:

Class	Hex	Description
Identity	0x01 (1)	Device info: vendor, product name, revision
Message Router	0x02 (2)	CIP message router
Connection Manager	0x06 (6)	Connection management
Assembly	0x04 (4)	I/O data assemblies
Parameter	0x0F (15)	Device parameters

7.3 CIP Services

Service	Direction	Suitable for
Get Attribute Single	Read	Single known attribute
Get Attribute All	Read	All attributes of an object at once
Set Attribute Single	Write	Set a single attribute
Read Tag	Read	Allen-Bradley Logix tag name
Write Tag	Write	Allen-Bradley Logix tag name

7.4 Data Types

Type	CIP Code	Size	Description
BOOL	0xC1	1 byte	Boolean (0/1)
SINT	0xC2	1 byte	8-bit signed
INT	0xC3	2 bytes	16-bit signed
DINT	0xC4	4 bytes	32-bit signed
LINT	0xC5	8 bytes	64-bit signed
USINT	0xC6	1 byte	8-bit unsigned
UINT	0xC7	2 bytes	16-bit unsigned
UDINT	0xC8	4 bytes	32-bit unsigned
REAL	0xCA	4 bytes	IEEE 754 Float32
LREAL	0xCB	8 bytes	IEEE 754 Float64
STRING	0xD0	variable	CIP string (4-byte length + data)

RAW	—	variable	Hex raw data

Save: "Save" button (bottom) – only active when Name and CIP Path are filled.

8. Log (LogScreen)

How to reach: Bottom Navigation → **History icon**

Shows all recorded polling sessions. Recording starts automatically with polling and ends when stopped.

8.1 Session Overview

Each session card shows:

- Device name
- Start date and time
- End date and time (when finished)
- Number of stored entries

8.2 View Entries (Expand)

Path: LogScreen → Tap session card

- Card expands and shows the **last 5 entries per tag** (timestamp + value)
- If more than 5 entries exist, a hint "… X more entries (use CSV export for all)" is shown
- Tapping again collapses the card

8.3 CSV Export

Path: LogScreen → Session card → **Share icon** (↗) → System share dialog → Choose target app

CSV format:

```
Timestamp,TagId,TagName,Value
2026-05-20 14:23:05.123,uuid-123,"Temperature","23.5 °C"
2026-05-20 14:23:05.124,uuid-456,"Speed","1450 rpm"
```

The share button is disabled (greyed out) when the session has no entries.

8.4 Delete Session

Path: LogScreen → 🗑 **icon** on the session card → Confirmation dialog shows date → "Delete"

Deleting also removes all associated entries from the database.

9. Alarms (AlarmScreen)

How to reach: Bottom Navigation → **Bell icon** (with red badge for unacknowledged alarms)

9.1 Alarm Overview

Shows all alarm events in descending chronological order. Each alarm card shows:

- Device name and tag name
- Alarm type (WARN / ALARM) with colored icon
- Triggered value and threshold
- Date and time
- Acknowledgement status

Alarm types:

Type	Icon	Trigger
ALARM	Red error icon	Alarm threshold exceeded
WARN	Yellow warning icon	Warning threshold exceeded

9.2 Acknowledge Alarm

Path: AlarmScreen → Alarm card → ✓ **icon** (green checkmark on the right)

- Alarm remains in the list
- Badge counter in the bottom navigation decreases
- Checkmark icon changes to a greyed-out outline icon

9.3 Delete All Alarms

Path: AlarmScreen → 🗑 **icon** (top right in the toolbar) → tap

> *Only visible when entries are present.*

10. Diagnostics (DiagScreen)

How to reach: Settings (SettingsScreen) → Diagnostics button

Protocol analysis and statistics for all EtherNet/IP communication. Records automatically when enabled. The screen is split into two tabs.

10.1 Start/Stop Recording

Path: DiagScreen → ▶/⏸ **button** in the toolbar

- When recording is active: red **"REC"** badge visible in the title
- Ring buffer stores the last **500 frames**
- State is persisted via `AppPreferences` (survives app restart)

10.2 Tab "Frames" – View and Filter Frames

Path: DiagScreen → tap tab **"Frames (n)"**

Step	Control	Description
1	Filter chip **"All"**	Show all recorded frames
1	Filter chip **"OK"**	Show successful frames only
1	Filter chip **"Errors"**	Show error frames only
2	Tap frame card	Expands → shows hex dump of request + response

Each frame card shows:

- IP address and port
- Timestamp (HH:mm:ss.SSS)
- Latency badge in milliseconds
- Success icon (green) or error icon (red) with message
- Expanded: Wireshark-style hex dump (16 bytes/row with offset and ASCII panel)

10.3 Tab "Statistics" – Analysis

Path: DiagScreen → tap tab **"Statistics"**

Displays automatically:

- **Overview:** Total transactions · Successes · Errors · Success rate (%)
- **Error breakdown:** Timeouts / no response · Other I/O errors
- **Latency sparkline:** graphical trend line with 200 ms threshold marker
- **Latency metrics:** Min · Avg · P95 · Max (in ms)

10.4 Export PCAP File (for Wireshark)

Path: DiagScreen → **PCAP icon** in the toolbar → Share dialog → choose target app

> *The exported `.pcap` file contains valid Ethernet + IPv4 + TCP + CIP headers and can be opened directly in Wireshark.*

10.5 Clear Frame Buffer

Path: DiagScreen → 🗑 **icon** in the toolbar → confirmation dialog → "Delete"

11. Settings (SettingsScreen)

How to reach: Bottom Navigation → **Gear icon**

11.1 Change Color Scheme

Path: Settings → "Appearance" section → "Color Scheme" dropdown

Option	Description
System	Follows the Android system setting (default)

Light	Light Material 3 theme (primary color: teal)
Dark	Dark Material 3 theme

Changes take effect immediately without restart.

11.2 Change Polling Interval

Path: Settings → "Polling Interval" section → Move slider

- Range: 0.5 s to 30 s
- Current value is displayed in seconds above the slider
- Applies to all devices

11.3 Configure MQTT Integration

Path: Settings → "MQTT Integration" section

The MQTT integration automatically forwards measurement values from polling to an MQTT broker.

Field	Description
MQTT enabled	Toggle to enable/disable the entire MQTT feature
Broker Address	Hostname or IP address of the MQTT broker (e.g. `192.168.1.10`)
Broker Port	TCP port (default: 1883)
Username	Optional; only if the broker requires authentication
Password	Optional; input is hidden (👁 icon to reveal)
Topic Prefix	Prefix for all topics (default: `eip`)
QoS Level	0 = At most once, 1 = At least once

Topic format: `{prefix}/{deviceName}/{tagName}`

Example: `eip/PLC1/MotorTemperature`

Spaces in device and tag names are automatically replaced with underscores.

Connection control:

Status	Display	Action
Connected	Green badge "Connected"	"Disconnect" button
Connecting…	Yellow badge	— (wait)
Disconnected	Grey badge	"Connect" button
Error	Red badge "Connection error"	"Connect" button

MQTT credentials are stored exclusively locally on the device. No data is transmitted to servers operated by the app developer.

11.4 License Status

Path: Settings → "License" section

- **Pro active:** Green verified icon + text "Pro version active"
- **Free version:** Info text + "Upgrade to Pro" button

11.5 About

Shows version number, developer, and implemented protocol details.

12. License Model

EtherNet/IP++ uses a **freemium model** via Google Play In-App Purchases.

12.1 Free Version

Feature	Free
Number of devices	Unlimited
Tags per device	Max. 5
Polling	✓
Recording	✓
CSV export	✓
Alarms	✓
Diagnostics tracer	✓

12.2 Pro Version

Feature	Pro
Tags per device	Unlimited
All free features	✓

12.3 Upgrade

Path: Settings → "License" section → **"Upgrade to Pro"** button

or: Device detail → 🔒 FAB → Pro dialog → **"Upgrade to Pro"**

The purchase is processed via **Google Play Billing**. Product ID: `ethernetplusplus_pro`.

12.4 Restore Purchase

The purchase is automatically restored on app launch when the Google account is linked to the original purchase.

13. Data Models

EipDevice

```
id           - UUID (auto-generated)
name         - Device label
description  - Description
ip           - IP address (default: 192.168.1.1)
port         - TCP port (default: 44818)
slot         - Backplane slot (default: 0)
tags         - List of EipTag entries
```

EipTag

```
id             - UUID
name           - Label
description    - Description
cipPath        - CIP path (e.g. "1.1.1" or Logix tag name)
serviceCode    - CIP service (GET_ATTRIBUTE_SINGLE etc.)
dataType       - Data type (BOOL, INT, DINT, REAL, STRING ...)
ip             - IP override (empty = use device IP)
port           - Port override (0 = use device port)
scale          - Scale factor (default: 1.0)
offset         - Offset after scaling (default: 0.0)
unit           - Unit text
widgetType     - AUTO / VALUE / LED / GAUGE / SPARKLINE
gaugeMin/Max   - Range for bar display
thresholdWarn  - Warning threshold (optional)
thresholdAlarm - Alarm threshold (optional)
alarmOnChange  - Alarm on value change
writeValue     - Pre-filled write value
```

14. EtherNet/IP Protocol Implementation

14.1 Encapsulation Layer (`EipEncapsulation.kt`)

The encapsulation layer wraps CIP messages according to the ODVA EtherNet/IP specification (Vol. 2).

Header format (24 bytes):

```
Byte  0- 1: Command (e.g. 0x0065 = Register Session)
Byte  2- 3: Length (data length excluding header)
Byte  4- 7: Session Handle (set after RegisterSession)
Byte  8-11: Status (0x00000000 in requests)
Byte 12-19: Sender Context (8 bytes, echoed back)
Byte 20-23: Options (0x00000000)
```

Transaction sequence:

```
1. Open TCP connection to IP:44818 (timeout: 4 s)
2. Send Register Session (Command 0x0065, Protocol Version = 1)
3. Extract Session Handle from response
4. Send Send RR Data (Command 0x0065) with CIP request in CPF wrapper
```

```
5. Read response → extract CIP response from CPF
6. Send Unregister Session (Command 0x0066)
7. Close TCP connection
```

Each tag read opens its own TCP connection. This is standard for Explicit Messaging without persistent connections.

Common Packet Format (CPF):

```
Interface Handle (4 bytes, always 0x00000000 for CIP)
Timeout (2 bytes)
Item Count (2 bytes, always 2)
  Address Item: Type 0x0000 (Null Address), Length 0
  Data Item:    Type 0x00B2 (Unconnected Data), Length + CIP bytes
```

14.2 CIP Layer (`CipBuilder.kt`)

CIP request format:

```
Byte 0:    Service Code (e.g. 0x0E = Get Attribute Single)
Byte 1:    Path Size in Words
Byte 2..n: EPATH (CIP path)
Byte n+1..: Request Data (write services only)
```

CIP response format:

```
Byte 0:    Service Code | 0x80 (mirrored service with bit 7 set)
Byte 1:    Reserved (0x00)
Byte 2:    General Status (0x00 = Success)
Byte 3:    Extended Status Size (in words)
Byte 4...: Response Data
```

EPATH encoding:

Segment type	Format	Example for Class 1
Class ID (≤ 0xFF)	`0x20 [Class]`	`0x20 0x01`
Class ID (> 0xFF)	`0x21 0x00 [Lo] [Hi]`	`0x21 0x00 0x00 0x01`
Instance ID (≤ 0xFF)	`0x24 [Inst]`	`0x24 0x01`
Attribute ID (≤ 0xFF)	`0x30 [Attr]`	`0x30 0x01`
Symbolic	`0x91 [StrLen] [Bytes…] [Pad]`	`0x91 0x0B "MyMotorSpd" 0x00`

14.3 CIP General Status Codes

Code	Meaning
0x00	Success
0x08	Service not supported
0x09	Invalid attribute value
0x0C	Object state conflict
0x0E	Attribute not settable

0x14	Attribute not supported
0x16	Too much data
0x17	Object does not exist
0x25	Path segment error
0x26	Path destination unknown

14.4 Byte Order

EtherNet/IP / CIP uses **little endian** throughout for all numeric values (except in some vendor-specific implementations). Decoding in `CipBuilder.decodeValue()` uses `ByteOrder.LITTLE_ENDIAN`.

14.5 Logix Read Tag (0x4C)

For Allen-Bradley ControlLogix / CompactLogix:

```
Request:
  Service:    0x4C
  Path:       Symbolic Segment (tag name)
  Data:       Element Count (2 bytes, little endian)

Response (on success):
  Byte 0-3:   CIP response header (Service | 0x80, Reserved, Status,
ExtStatusSize)
  Byte 4-5:   CIP Data Type Code (e.g. 0x00CA = REAL)
  Byte 6-7:   Element Count
  Byte 8-...: Value data (little endian, type from bytes 4-5)
```

15. Database

Room database (version 1) with three tables:

log_sessions

Recording sessions per device

```
id          - autoincrement (PK)
deviceId    - ID of the recorded device
deviceName  - Name at time of recording
startTime   - Unix timestamp (ms)
endTime     - Unix timestamp (ms), null if active
entryCount  - Number of stored entries
```

log_entries

Individual measurements within a session

```
id          - autoincrement (PK)
sessionId   - Reference to session
timestamp   - Unix timestamp (ms)
tagId       - UUID of the tag
tagName     - Name at time of recording
value       - Decoded display value (with unit)
```

alarm_events

Alarm events and their acknowledgement status

```
id           – autoincrement (PK)
timestamp    – Unix timestamp (ms)
deviceId     – UUID of the device
deviceName   – Name at time of alarm
tagId        – UUID of the tag
tagName      – Name at time of alarm
value        – Triggering value
threshold    – Threshold at time of alarm
type         – WARN / ALARM
acknowledged – Boolean (false = unacknowledged)
```

16. Permissions

Permission	Purpose
`INTERNET`	TCP connections to EtherNet/IP devices (port 44818) and optional MQTT connection to the configured broker
`ACCESS_NETWORK_STATE`	Check network status
`POST_NOTIFICATIONS`	Alarm notifications (Android 13+)
`NFC`	Write and read NFC tags with device profiles (optional, only on devices with NFC)

> ***Note:** `INTERNET` is used for EtherNet/IP communication (port 44818) and — when enabled — for the connection to the MQTT broker. No data is transmitted to servers operated by the app developer. `NFC` is declared as `required=false`: the app can be used fully on devices without NFC. `POST_NOTIFICATIONS` is requested automatically on the first alarm (Android 13+).*

17. Wear OS

Setup: Pair a Wear OS smartwatch with the Android device → EtherNet/IP++ runs automatically on the watch as soon as data is sent.

The Wear OS companion app receives measurement values after each polling cycle:

Display elements on the watch:

- Timestamp of last update (top)
- Scrollable list of all tag values (ScalingLazyColumn)
- Per entry: **Tag name** (small) · **Value** (large)
- "No data" placeholder when no values have been received yet

Data transfer: Wearable Data API, path `/eip/values` — triggered automatically after each successful polling operation.

Documentation updated 2026-05-21 · EtherNet/IP++ v1.0