v0.6.0 Group Addressing - Implementation Changelog

Date: February 15, 2026 **Status:** COMPLETE AND VERIFIED **Result:** `ALL:READ` with 2 nodes now returns both responses simultaneously via a single BLE Mesh group broadcast to `0xC000`.

Final File Changes Summary

File	Lines Changed	What Changed
`ESP/ESP-Provisioner/main/main.c`	~35 new	Group constant, `vnd_srv_subscribed` flag, `subscribe_vendor_model_to_group()`, extend `bind_next_model()`, handle `MODEL_SUB_ADD` in `config_client_cb()`
`ESP/ESP_GATT_BLE_Gateway/main/main.c`	~60 new/modified	Group constant, replace ALL loop with group send, skip busy-wait for group, `RECV_PUBLISH_MSG_EVT` handler, group timeout handling
`ESP/ESP-Mesh-Node-sensor-test/main/main.c`	~4 new	Fix `recv_dst` override for group-addressed reply context
`gateway-pi5/gateway.py`	~15 modified	Simplify `send_to_node()` ALL path, simplify `_poll_all_nodes()`

Total: ~115 lines across 4 files (more than the planned ~65 due to bugs discovered during implementation).

Batch 1: Provisioner (Tasks 1-2) - PASSED FIRST TRY

Changes Made

1. **Line 33** - Added `#define MESH_GROUP_ADDR 0xC000`
2. **Line 67** - Added `bool vnd_srv_subscribed;` to `mesh_node_info_t`
3. **Lines 311-334** - New `subscribe_vendor_model_to_group()` function using `MODEL_SUB_ADD` opcode
4. **Lines 356-359** - New step 5 in `bind_next_model()`: subscribe vendor server to group after all binds done
5. **Lines 666-667** - Error handler catches `MODEL_SUB_ADD` failures (skips to next step)
6. **Lines 739-745** - Success handler for `MODEL_SUB_ADD`: sets flag, chains to "FULLY CONFIGURED"

Verification

Provisioner console showed the expected new lines in the bind chain:

```
Subscribing Vnd Server on 0x0005 to group 0xc000
Group subscription added on 0x0005
========== NODE 0x0005 FULLY CONFIGURED ==========
```

GATT Gateway (vendor CLIENT, no vendor SERVER) correctly skipped group subscription and went straight to FULLY CONFIGURED.

Batch 2: GATT Gateway (Tasks 3-4) - Required Multiple Iterations

Changes Made

1. **Line 70** - Added `#define MESH_GROUP_ADDR 0xC000`
2. **Lines 354-401** - `send_vendor_command()` modified:
   - Detects group address via `bool is_group = (target_addr == MESH_GROUP_ADDR)`
   - Skips busy-wait for group sends
   - Skips setting `vnd_send_busy`/`vnd_send_target_addr` for group
   - Uses `need_ack=true, timeout=5000` for all sends (required for TX context matching)
3. **Lines 607-609** - `is_all` path: replaced sequential `for` loop + discovery probe with single `send_vendor_command(MESH_GROUP_ADDR, ...)`
4. **Lines 465-490** - Timeout handler: added early-exit for group send timeouts (silently ignores expected SDK timeout)
5. **Lines 490-511** - NEW `ESP_BLE_MESH_CLIENT_MODEL_RECV_PUBLISH_MSG_EVT` handler (critical fix - see Bug #3)

Build Error Fix

Added `case ... { }` compound statement for timeout handler but forgot closing brace. The `case ESP_BLE_MESH_CLIENT_MODEL_SEND_TIMEOUT_EVT: {` opened a block that was never closed, causing all subsequent functions to be parsed as nested functions. Fixed by adding `}` before `default:`.

Batch 3: Pi 5 Gateway (Tasks 5-6) - PASSED FIRST TRY

Changes Made

1. **`send_to_node()` (lines 942-947)** - Old: loop over each known node, send `{nid}:{command}`, wait per-node. New: send one `ALL:{command}` string
2. **`_poll_all_nodes()` (lines 412-422)** - Old: loop per-node with individual READ + wait. New: send one `ALL:READ`, then `_wait_for_responses(timeout=3.0)`

Batch 3.5: Mesh Node Fix (Unplanned) - Critical Bug

Change Made

`ESP/ESP-Mesh-Node-sensor-test/main/main.c` lines 619-624 - Override `ctx.recv_dst` with node's own unicast address before replying to group-addressed messages.

```
// When message arrived via group address (0xC000), recv_dst is the
// group addr. The server send uses recv_dst as the reply source,
// but we can't send FROM a group address -- override with our unicast.
if (ctx.recv_dst != node_state.addr) {
    ctx.recv_dst = node_state.addr;
}
```

This was NOT in the original plan ("mesh node needs zero code changes") but was required. See Bug #2 below.

Bugs Found and Fixed During Implementation

Bug #1: `need_ack=false` Drops Responses (GATT Gateway)

Symptom: `ALL:DUTY:100` worked (both nodes lit up) but `ALL:READ` returned nothing. No `Vendor STATUS from 0x000X` on gateway log.

Root Cause: With `need_ack=false, timeout=0`, the ESP-IDF SDK's client model doesn't create a TX context. When the mesh node's `VND_OP_STATUS` reply arrives, the SDK can't match it to anything. Because `VND_OP_STATUS` is registered in the client model's `vnd_op_pair[]` as the response to `VND_OP_SEND`, the SDK routes it through the client-model TX matching path. With no TX context, the message is silently dropped.

Why DUTY worked: `DUTY` is fire-and-forget (no response expected from the Pi 5 perspective). The node processes it but any response is irrelevant.

Fix: Changed to `need_ack=true, timeout=5000` for all sends. This creates a TX context that can receive the reply.

Lesson: In ESP-IDF BLE Mesh, if a vendor opcode is registered in the client model's `op_pair`, ALL incoming messages with that opcode go through client-side TX context matching first. `need_ack=false` doesn't bypass this - it just means there's no TX context to match against, so the message is dropped.

Bug #2: Node Can't Reply FROM Group Address (Mesh Node)

Symptom: Nodes received the group READ (confirmed on node serial: `Vendor SEND from 0x0005: read`, `Response -> 0x0005`, `Vendor send COMP OK`) but gateway never received the STATUS.

Root Cause: When a message arrives at the node addressed to group `0xC000`, the incoming context has `recv_dst = 0xC000`. The node copies this context verbatim:

```
esp_ble_mesh_msg_ctx_t ctx = *param->model_operation.ctx;
```

Then calls `esp_ble_mesh_server_model_send_msg(&vnd_models[0], &ctx, VND_OP_STATUS, ...)`. The SDK uses `recv_dst` as the **source address** of the reply. A node can't send FROM a group address `0xC000` - the reply is silently dropped by the mesh stack.

Fix: Override `ctx.recv_dst` with the node's own unicast address (`node_state.addr`) before sending the reply.

Lesson: The original plan stated "mesh node needs zero code changes" and "ESP-IDF delivers group messages automatically." This is true for RECEIVING. But REPLYING requires the node to fix up the context, because the SDK's server send function uses `recv_dst` as the reply source address. This is only an issue for group-addressed messages where `recv_dst` is a group address rather than the node's unicast.

Bug #3: `CLIENT_MODEL_RECV_PUBLISH_MSG_EVT` Not Handled (GATT Gateway)

Symptom: After fixing Bugs #1 and #2, node logs confirmed replies were being sent and leaving the radio (`Vendor send COMP OK`). Gateway still showed no `Vendor STATUS from 0x000X`.

Root Cause: With `need_ack=true`, the SDK creates a TX context for dest `0xC000`. When the first reply arrives with matching opcode `VND_OP_STATUS`, the SDK matches it and delivers it via `ESP_BLE_MESH_CLIENT_MODEL_RECV_PUBLISH_MSG_EVT` - NOT via `ESP_BLE_MESH_MODEL_OPERATION_EVT`. Our `custom_model_cb` only had a handler for `MODEL_OPERATION_EVT`. The publish event fell through to `default: break;` silently.

Fix: Added full handler for `ESP_BLE_MESH_CLIENT_MODEL_RECV_PUBLISH_MSG_EVT` with identical logic to the `MODEL_OPERATION_EVT` handler (extract src, format NODE:DATA:, GATT notify to Pi 5).

Lesson: ESP-IDF BLE Mesh client model delivers matched responses via DIFFERENT events depending on how they're matched:

- `MODEL_OPERATION_EVT` - unmatched/unsolicited messages (opcode registered in model ops)
- `CLIENT_MODEL_RECV_PUBLISH_MSG_EVT` - messages matched to a TX context via `op_pair`

Both events must be handled to receive responses in all scenarios.

Bug #4: Compound Statement Missing Closing Brace (Build Error)

Symptom: Build failed with "invalid storage class for function" errors on every function after the timeout handler.

Root Cause: `case ESP_BLE_MESH_CLIENT_MODEL_SEND_TIMEOUT_EVT: {` opened a compound statement block (needed for the `uint16_t timeout_target` variable declaration), but the closing `}` was placed after `break;` instead of before it. This left the block open, and all subsequent functions were parsed as nested inside it.

Fix: Moved `}` to before `default:`.

Debugging Practices Used

1. Incremental Batch Verification

Implemented in 3 batches (Provisioner, GATT Gateway, Pi 5 gateway), verifying each works before proceeding. This isolated which layer had issues.

2. Cross-Device Log Correlation

When gateway showed no response, checked mesh node logs to determine whether the node received the message and attempted to reply. This narrowed Bug #2 to the node's reply path.

3. ESP-IDF Event Tracing

Added `ESP_LOGD` catch-all in the `default:` case of `custom_model_cb` to discover that events were arriving but being silently dropped. This led to discovering Bug #3.

4. Fire-and-Forget vs Response Commands

Used `ALL:DUTY:100` (fire-and-forget) to verify group delivery worked, then `ALL:READ` (requires response) to isolate the response path. The fact that DUTY worked but READ didn't confirmed the issue was in response handling, not message delivery.

5. Address Tracking

Carefully tracked which unicast addresses the provisioner assigned (varies each time due to provisioning order). Caught that `node 1` was targeting the gateway's own address (0x0006) instead of a mesh node.

Key Architecture Insights Discovered

1. ESP-IDF Client Model Opcode Routing

If an opcode is registered in a client model's `op_pair`, the SDK ALWAYS routes incoming messages with that opcode through client-side TX context matching first. This happens regardless of `need_ack` setting. The opcode only falls through to `MODEL_OPERATION_EVT` if there's no TX context AND the opcode is also registered in the model's `vnd_op[]` array.

2. Group Address Reply Context

BLE Mesh server model send uses `ctx.recv_dst` as the source address of replies. For unicast messages, `recv_dst` is the node's own address (correct). For group messages, `recv_dst` is the group address (incorrect for replies). Nodes MUST override `recv_dst` with their unicast address before replying.

3. Flash Erase Coordination

ALL mesh devices must be erased together. Erasing one device and re-provisioning it while others retain old keys creates a split network. Group subscriptions are only added during provisioning, so nodes provisioned before the provisioner code change won't have them.

4. Provisioning Order Non-Determinism

The provisioner assigns unicast addresses in the order devices are discovered, which varies. The GATT Gateway might get 0x0005 or 0x0006 depending on which device advertises first. Pi 5 gateway node ID mapping (`NODE_BASE_ADDR + id`) must account for this.

Performance Results

Before (v0.5.0)

- `ALL:READ` with 2 nodes: ~5s (2 x 2.5s stagger)
- PM poll cycle: ~5s per cycle

After (v0.6.0)

- `ALL:READ` with 2 nodes: ~0.5s (single group broadcast, both respond within ~500ms)
- Both responses arrive at the same timestamp on Pi 5:

```
[15:50:46] NODE1 >> D:100%,V:11.721V,I:503.75mA,P:5904.6mW
[15:50:46] NODE2 >> D:100%,V:11.641V,I:402.50mA,P:4685.6mW
```

- `ALL:DUTY:100` applies to both nodes simultaneously

Scaling

- With N nodes, `ALL:READ` is still O(1) for the send. Response collection time depends on mesh propagation but should remain well under 3s for typical mesh topologies.

Verified Working Scenarios

1. `node 0` -> `read` (unicast READ to single node) - WORKS
2. `node 2` -> `read` (unicast READ to second node) - WORKS
3. `node all` -> `read` (group READ to all nodes) - WORKS
4. `node all` -> `100` (group DUTY to all nodes) - WORKS
5. `node all` -> `0` (group DUTY:0 stop) - WORKS
6. Both responses returned simultaneously at same timestamp - CONFIRMED
7. GATT Gateway log shows `Vendor SEND to 0xc000` for group commands - CONFIRMED
8. GATT Gateway log shows unicast addresses for single-node commands - CONFIRMED
9. Provisioner correctly subscribes vendor server nodes to 0xC000 - CONFIRMED
10. Provisioner correctly skips subscription for vendor client (GATT Gateway) - CONFIRMED