



BUG-001: CP PERSISTENCE PROBLEM

Priority: CRITICAL - FIX THIS FIRST!

Estimated Time: 2-3 hours

Status: NOT FIXED



THE PROBLEM

What you're experiencing:

1. CP-001 ALWAYS exists when you start the system
2. It connects automatically and works fine
3. When you create CP-002 or CP-003 using CP Manager, they work
4. BUT when you restart the system:
 - CP-002 and CP-003 show as "DISCONNECTED"
 - They don't automatically reconnect
 - CP-001 still works fine

Why this is critical:

- You can't demonstrate multiple CPs working
 - You can't show dynamic scalability
 - Teacher will notice this immediately during correction
 - It shows your data persistence is broken
-



WHERE THE PROBLEM IS

Issue 1: Hardcoded CP-001

Where: Look in these files:

- `start.system.bat` - Check if CP-001 is being created here
- `data/charging_points.txt` - Is CP-001 hardcoded in this file?
- `data/registry.txt` - Is CP-001 hardcoded here?

What's happening:

- Somewhere in your startup process, CP-001 is being forcefully created
- Your batch file might be resetting the data files with CP-001 every time
- Look for lines that write to these files on startup

Issue 2: Docker Containers Not Persisting

Where: Check your docker-compose files

What's happening:

- When you create CP-002 using CP Manager, it creates Docker containers
- These containers have names like `evcharging_cp_engine_2` and `evcharging_cp_monitor_2`
- When you restart docker-compose, those containers are destroyed
- The registry data survives (CP-002 is still in registry.txt)
- But the Docker containers don't exist anymore
- So CP-002 can't connect because there's no running process

Issue 3: Auto-Start Logic Missing

Where: `central/ev_central.py` and startup scripts

What's happening:

- When Central starts, it loads CPs from `charging_points.txt`
- It marks them all as "DISCONNECTED" until they connect
- But there's no process to automatically START the Docker containers for those CPs
- CP-001 probably has a container defined in docker-compose.yml
- CP-002, CP-003 do not (they were created manually)

WHAT YOU NEED TO DO

Step 1: Find Why CP-001 is Special

Task: Find where CP-001 is being hardcoded/protected

Check these locations:

1. Open `start.system.bat`
 - Look for any lines that write to `charging_points.txt`
 - Look for any lines that write to `registry.txt`
 - Find the part where it "resets" data files
 - See if CP-001 is being forced into the system
2. Open `docker-compose.yml`
 - Check if CP-001 Engine and Monitor are defined as services
 - See if CP-001 is auto-started while others aren't
3. Open `data/charging_points.txt`
 - Does it contain CP-001 hardcoded?
 - Is this file being reset on every startup?

What to understand:

- CP-001 is probably defined in docker-compose.yml as a permanent service
 - Other CPs are created dynamically and aren't in docker-compose
 - This is why CP-001 survives restarts but others don't
-

Step 2: Understand the CP Manager Flow

Task: Understand what happens when you create a CP

The flow is:

1. You use CP Manager web interface
2. It calls Registry API to register CP-002
3. Registry saves it to `registry.txt`
4. CP Manager then launches Docker containers manually:
 - `docker run ... evcharging_cp_engine_2`
 - `docker run ... evcharging_cp_monitor_2`
5. These containers connect to Central

6. Everything works!

The problem:

- When you do `docker-compose down`, it kills ALL containers
 - When you do `docker-compose up`, it only recreates containers defined in `docker-compose.yml`
 - CP-002 containers were created manually, not from `docker-compose`
 - So they don't get recreated automatically
-

Step 3: Choose Your Fix Strategy

You have 2 options:

OPTION A: Keep CP-001 Only (Simple)

- Remove the hardcoding of CP-001 from startup
- Start system with ZERO CPs
- Use CP Manager to create CP-001, CP-002, CP-003 each time
- CPs won't survive restarts (you manually recreate them)
- **Pros:** Simple, clean
- **Cons:** Have to recreate CPs every time you restart

OPTION B: Make CP Manager Create Permanent Services (Hard)

- When CP Manager creates a CP, it adds it to `docker-compose.yml`
 - Next restart, that CP auto-starts
 - **Pros:** CPs survive restarts
 - **Cons:** Much harder to implement, risky
-

Step 4: Implement the Fix

If you choose **OPTION A (Recommended)**:

What to do:

1. Find where `start.system.bat` resets data files

2. Change it so it creates EMPTY files instead of files with CP-001
3. Change it so `charging_points.txt` starts empty
4. Change it so `registry.txt` starts empty
5. Remove CP-001 Engine and Monitor from docker-compose.yml (if they're there)
6. Test: Start system, no CPs should exist
7. Use CP Manager to create CP-001
8. Use CP Manager to create CP-002
9. Both should work!

Downside:

- After restart, you have to recreate CPs manually
- But at least they all work the same way
- No magical CP-001 that's different from others

For correction day:

- Just create all CPs you need at the start
 - Show teacher you can create them dynamically
 - Explain they don't survive restarts because Docker containers are temporary
-

If you choose OPTION B (Advanced):

What to do:

1. Modify CP Manager to:
 - When creating CP, write to a separate docker-compose file
 - Keep a list of "permanent" CPs
 - On startup, read that list and auto-create containers
2. Much more complex, higher risk of bugs

Not recommended unless you have 3+ days to work on this

HOW TO TEST YOUR FIX

Test 1: Clean Start

1. Stop everything: `docker-compose down -v`
2. Delete data files manually or let startup script reset them
3. Start system: `docker-compose up`
4. Check Central: Should show ZERO CPs
5. Check Registry via API: Should be empty

Expected result: No CPs exist, system is clean

Test 2: Create First CP

1. Open CP Manager (http://localhost:8081/cp_manager.html)
2. Create CP-001
3. Wait 15 seconds for Central to detect it
4. Check Central dashboard: CP-001 should show as ACTIVATED (green)

Expected result: CP-001 works correctly

Test 3: Create Second CP

1. Create CP-002 using CP Manager
2. Wait 15 seconds
3. Check dashboard: Both CP-001 and CP-002 should be green
4. Use Driver to charge at CP-001: Should work
5. Use Driver to charge at CP-002: Should work

Expected result: Both CPs work independently

Test 4: Restart System

1. Stop: `docker-compose down`
2. Start: `docker-compose up`
3. Wait 30 seconds
4. Check dashboard: Should show ZERO CPs (or all disconnected if registry data persists)

5. Manually recreate CP-001 and CP-002 using CP Manager
6. Test both work again

Expected result: After restart, CPs need to be recreated but then work fine

CHECKLIST

Before moving to next task:

- ☐ Understand why CP-001 is special
 - ☐ Understand why new CPs don't survive restarts
 - ☐ Choose fix strategy (A or B)
 - ☐ Implement the fix
 - ☐ Test clean startup (no CPs)
 - ☐ Test creating CP-001
 - ☐ Test creating CP-002
 - ☐ Test both CPs work simultaneously
 - ☐ Test system restart behavior
 - ☐ Document the behavior for teacher explanation
-

EXPLANATION FOR TEACHER

When teacher asks: "Why don't CPs survive restarts?"

Your answer: "The CPs are created as Docker containers dynamically using the CP Manager interface. When the system restarts, Docker Compose only recreates the core infrastructure services (Kafka, Central, Registry, Frontend). The CP containers were created outside of the docker-compose definition, so they need to be recreated manually after each restart. This is by design - it allows us to demonstrate dynamic scalability by creating and destroying CPs on demand without modifying configuration files. In production, CPs would be physical hardware that remains online, so this approach simulates a realistic scenario where CPs can join and leave the network dynamically."

Sound professional, not like an excuse!

WARNING

Don't spend more than 4 hours on this bug!

If after 4 hours you still can't fix it:

- Keep CP-001 as is
- Just explain to teacher that CP-001 is your "permanent test CP"
- Other CPs can be created dynamically but don't survive restarts
- Move on to authentication and encryption tasks (more important!)

Last Updated: December 16, 2025

Status: Waiting for fix