CoAP Client Documentation

Overview

This document provides comprehensive documentation for interacting with the CoAP-based IoT device management system. The system consists of a main server and worker nodes, all communicating via CoAP (Constrained Application Protocol).

Table of Contents

System Architecture

```
┌─────────────────┐      CoAP       ┌─────────────────┐
│   Main Server   │◄──────────────►│  Worker Node 1  │
│ 192.168.100.79  │                 │ 192.168.100.80  │
│     :5683       │                 │     :5683       │
└─────────────────┘                 └─────────────────┘
         │
         │ CoAP
         ▼
┌─────────────────┐
│  Worker Node 2  │
│ 192.168.100.81  │
│     :5683       │
└─────────────────┘
```

Main Server Endpoints

Base URL: `coap://192.168.100.79:5683`

1. Health Management

GET /health

Get overall system health status.

Request:

```
coap-client -m get coap://192.168.100.79:5683/health
```

Response:

```json
{
  "status": "healthy",
  "timestamp": "2025-09-17T10:00:00",
  "nodes": [
    {
      "node_id": "node-123",
      "status": "online",
      "last_health_check": "2025-09-17T09:58:00"
    }
  ]
}
```

PUT /health

Report health check from a worker node.

Request:

```bash
echo '{
  "node_id": "node-123",
  "timestamp": "2025-09-17T10:00:00",
  "overall_healthy": true,
  "cpu_percent": 15.5,
  "memory_percent": 45.2,
  "disk_percent": 30.1,
  "temperature": 42.5,
  "services_status": {
    "systemd": true,
    "network": true,
    "ssh": true,
    "docker": false
  },
  "error_messages": ["Service docker is not running"]
}' | coap-client -m put coap://192.168.100.79:5683/health
```

Response:

```
Health status updated
```

2. Node Management

GET /nodes

List all registered nodes.

Request:

```
coap-client -m get coap://192.168.100.79:5683/nodes
```

Response:

```
{
  "nodes": [
    {
      "node_id": "node-123",
      "hostname": "worker-01",
      "ip_address": "192.168.100.80",
      "status": "online",
      "last_seen": "2025-09-17T10:00:00",
      "services": ["docker", "ssh"],
      "drivers": ["gpio", "i2c"],
      "system_info": {
        "os": "linux",
        "arch": "arm64"
      }
    }
  ]
}
```

POST /nodes

Register a new node.

Request:

```
echo '{
  "node_id": "node-456",
  "hostname": "worker-02",
  "ip_address": "192.168.100.81",
  "status": "online",
  "last_seen": "2025-09-17T10:00:00",
  "services": ["docker", "ssh", "mqtt"],
  "drivers": ["gpio", "i2c", "spi"],
  "system_info": {
    "os": "linux",
    "arch": "arm64",
    "kernel": "5.4.0"
  }
}' | coap-client -m post coap://192.168.100.79:5683/nodes
```

Response:

```
{
  "message": "Node registered successfully",
  "node_id": "node-456"
}
```

3. Update Management

GET /updates

List all update jobs.

Request:

```
coap-client -m get coap://192.168.100.79:5683/updates
```

Response:

```
{
  "updates": [
    {
      "job_id": "abc123-def456-ghi789",
      "status": "pending",
      "created_at": "2025-09-17T10:00:00",
      "target_nodes": ["all"],
      "package_name": "python-packages-update",
      "package_version": "2025.09.16",
      "update_type": "package"
    }
  ]
}
```

POST /updates (Create Update)

Create a new update job.

Request:

```
echo '{
  "name": "python-packages-update",
  "version": "2025.09.16",
  "package_type": "pip",
  "target_nodes": ["all"],
  "packages": [
    "aiocoap==0.4.7",
    "aiohttp==3.9.1",
    "pydantic==2.5.0"
  ]
}' | coap-client -m post coap://192.168.100.79:5683/updates
```

Response:

```
{
  "job_id": "abc123-def456-ghi789",
  "status": "created",
  "message": "Update request created successfully"
}
```

POST /updates (Trigger Update)

Trigger an update installation.

Request:

```
echo '{
  "action": "install",
  "job_id": "abc123-def456-ghi789"
}' | coap-client -m post coap://192.168.100.79:5683/updates
```

Response:

```
{
  "success": true,
  "message": "Update job abc123-def456-ghi789 triggered successfully",
  "job_id": "abc123-def456-ghi789"
}
```

POST /updates (Check Status)

Check update job status.

Request:

```
echo '{
  "action": "status",
  "job_id": "abc123-def456-ghi789"
}' | coap-client -m post coap://192.168.100.79:5683/updates
```

Response:

```
{
  "job_id": "abc123-def456-ghi789",
  "status": "in_progress",
  "node_statuses": {
    "node-123": "pending",
    "node-456": "in_progress"
  },
  "error_message": null
}
```

4. System Management

GET /system

Get system information.

Request:

```
coap-client -m get coap://192.168.100.79:5683/system
```

Response:

```
{
  "status": "running",
  "timestamp": "2025-09-17T10:00:00",
  "uptime": "unknown",
  "version": "1.0.0",
  "endpoints": {
    "health": "/health",
    "nodes": "/nodes",
    "updates": "/updates",
    "system": "/system"
  },
  "available_actions": [
    "restart",
    "shutdown",
    "status"
  ]
}
```

POST /system (System Actions)

Execute system actions.

Restart System:

```
echo '{"action": "restart"}' | coap-client -m post
coap://192.168.100.79:5683/system
```

Shutdown System:

```
echo '{"action": "shutdown"}' | coap-client -m post
coap://192.168.100.79:5683/system
```

Get System Status:

```
echo '{"action": "status"}' | coap-client -m post
coap://192.168.100.79:5683/system
```

Response:

```
{
  "status": "running",
  "timestamp": "2025-09-17T10:00:00",
  "services": {
    "main_server": "running",
    "database": "connected"
  },
  "action": "status"
}
```

5. Test Endpoint

GET /test

Simple test endpoint for connectivity verification.

Request:

```
coap-client -m get coap://192.168.100.79:5683/test
```

Response:

```
Test resource working
```

Worker Node Endpoints

Base URL: `coap://{NODE_IP}:5683` (e.g., `coap://192.168.100.80:5683`)

1. Health Endpoint

GET /health

Get node health status.

Request:

```
coap-client -m get coap://192.168.100.80:5683/health
```

2. System Endpoint

GET /system

Get node system information.

Request:

```
coap-client -m get coap://192.168.100.80:5683/system
```

3. Update Endpoints

POST /updates/available

Receive update notifications from main server (used internally).

Client Tools

1. coap-client

The primary tool for interacting with CoAP endpoints.

Installation:

```
# Ubuntu/Debian
sudo apt-get install libcoap2-dev coap-client

# CentOS/RHEL
sudo yum install libcoap-devel coap-client

# macOS
brew install libcoap
```

Basic Usage:

```
# GET request
coap-client -m get coap://192.168.100.79:5683/health

# POST request with JSON payload
echo '{"action":"status"}' | coap-client -m post
coap://192.168.100.79:5683/system

# PUT request with JSON payload
echo '{"node_id":"test"}' | coap-client -m put
coap://192.168.100.79:5683/health
```

Examples

Complete Update Workflow

```
#!/bin/bash
# Complete update workflow example

echo "=== CoAP Update Workflow ==="

# 1. Check system health
echo "1. Checking system health..."
coap-client -m get coap://192.168.100.79:5683/health
echo

# 2. List current nodes
echo "2. Listing nodes..."
coap-client -m get coap://192.168.100.79:5683/nodes
echo

# 3. Create update
echo "3. Creating update..."
RESPONSE=$(echo '{
  "name": "python-packages-update",
  "version": "2025.09.16",
  "package_type": "pip",
  "target_nodes": ["all"],
  "packages": ["aiocoap==0.4.7", "aiohttp==3.9.1"]
}' | coap-client -m post coap://192.168.100.79:5683/updates)

echo "Response: $RESPONSE"
echo

# 4. Extract job_id (manual step)
echo "4. Please copy the job_id from the response above"
echo "5. Then run:"
echo "   echo '{\"action\":\"install\",\"job_id\":\"YOUR_JOB_ID\"}' | coap-
client -m post coap://192.168.100.79:5683/updates"
echo "   echo '{\"action\":\"status\",\"job_id\":\"YOUR_JOB_ID\"}' | coap-
client -m post coap://192.168.100.79:5683/updates"
```

Node Registration Workflow

```bash
#!/bin/bash
# Node registration workflow

echo "=== Node Registration Workflow ==="

# 1. Register new node
echo "1. Registering new node..."
echo '{
  "node_id": "worker-03",
  "hostname": "worker-03",
  "ip_address": "192.168.100.82",
  "status": "online",
  "last_seen": "2025-09-17T10:00:00",
  "services": ["docker", "ssh"],
  "drivers": ["gpio", "i2c"],
  "system_info": {
    "os": "linux",
    "arch": "arm64"
  }
}' | coap-client -m post coap://192.168.100.79:5683/nodes
echo

# 2. Verify registration
echo "2. Verifying registration..."
coap-client -m get coap://192.168.100.79:5683/nodes
echo

# 3. Report health from new node
echo "3. Reporting health from new node..."
echo '{
  "node_id": "worker-03",
  "overall_healthy": true,
  "cpu_percent": 10.5,
  "memory_percent": 35.2,
  "services_status": {"systemd": true, "network": true}
}' | coap-client -m put coap://192.168.100.79:5683/health
```

Error Handling

Common Response Codes

- `2.05 Content` - Success with content
- `2.04 Changed` - Success, resource modified
- `2.01 Created` - Success, resource created
- `4.00 Bad Request` - Invalid request
- `4.04 Not Found` - Resource not found
- `4.05 Method Not Allowed` - Invalid method
- `5.00 Internal Server Error` - Server error

Error Handling Best Practices

1. **Always check response codes** - CoAP uses numeric response codes
2. **Parse JSON responses** - Most responses are JSON formatted
3. **Handle timeouts** - CoAP requests can timeout on slow networks
4. **Validate input** - Ensure required fields are present before sending

Troubleshooting

Common Issues

1. **Connection Refused**
    - Check if CoAP server is running
    - Verify IP address and port
    - Check firewall settings
2. **Timeout Errors**
    - Increase timeout values
    - Check network connectivity
    - Verify server responsiveness
3. **JSON Parse Errors**
    - Validate JSON format
    - Check for special characters
    - Ensure proper encoding
4. **Resource Not Found**
    - Verify endpoint paths
    - Check if resource exists
    - Validate request format