UDC v1.0 Compliance Documentation

This document certifies that the Multi-Cloud Manager droplet is fully compliant with the Unified Droplet Communication (UDC) protocol version 1.0.

Compliance Status: ✅ CERTIFIED

Droplet: Multi-Cloud Manager (#4)
UDC Version: 1.0
Certification Date: 2025-11-13
Certified By: Hassan (Steward)

Table of Contents

---

1. Required Endpoints

Core Health & Status Endpoints

Endpoint	Method	Auth	Status	Notes
`/health`	GET	None	✅	Returns droplet health + dependencies
`/capabilities`	GET	None	✅	Returns features + UDC version
`/state`	GET	None	✅	Returns CPU, memory, uptime
`/version`	GET	None	✅	Returns build info + versions

Verification:

```bash
```

```
curl http://localhost:8010/health
curl http://localhost:8010/capabilities
curl http://localhost:8010/state
curl http://localhost:8010/version
```

Communication Endpoints

Endpoint	Method	Auth	Status	Notes
/message	POST	JWT	✅	Receives UDC messages
/send	POST	JWT	✅	Sends UDC messages
/dependencies	GET	JWT	✅	Lists connected droplets

Verification:

```bash
curl -H "Authorization: Bearer TOKEN" \
  -X POST http://localhost:8010/message \
  -H "Content-Type: application/json" \
  -d '{"trace_id":"test","source":5,"target":4,"message_type":"query","payload":{},"timestamp":"2025-11-13T08:00:00Z"}'
```

Monitoring & Observability

Endpoint	Method	Auth	Status	Notes
/metrics	GET	None	✅	Prometheus format metrics
/logs	GET	JWT	✅	Structured log entries
/events	GET	JWT	✅	Event history
/proof	GET	JWT	✅	Last verified action

Verification:

```bash
curl http://localhost:8010/metrics
curl -H "Authorization: Bearer TOKEN" http://localhost:8010/logs
curl -H "Authorization: Bearer TOKEN" http://localhost:8010/events
curl -H "Authorization: Bearer TOKEN" http://localhost:8010/proof
```

Control & Management

Endpoint	Method	Auth	Status	Notes
`/reload-config`	POST	JWT	✅	Hot reload configuration
`/shutdown`	POST	JWT	✅	Graceful shutdown
`/emergency-stop`	POST	JWT	✅	Immediate stop

Verification:

```bash
curl -H "Authorization: Bearer TOKEN" \
  -X POST http://localhost:8010/reload-config
```

---

2. Message Protocol

UDC Message Format ✅

Standard Message Structure:

```json
{
  "trace_id": "uuid-v4-format",
  "source": 5,
  "target": 4,
  "message_type": "command|query|event|response",
  "payload": {},
  "timestamp": "ISO-8601-format"
}
```

Message Types Supported

Type	Status	Description
`event`	✅	Notification of an event
`query`	✅	Request for information
`command`	✅	Action request
`response`	✅	Reply to a query

Trace ID Handling ✅

- All messages include `trace_id` (UUID v4)
- Trace IDs propagate through request chain
- Logged with every operation
- Available in `/logs` and `/events` endpoints

Example:

```python
# Incoming request
trace_id = request.headers.get("X-Trace-ID")
if not trace_id:
    trace_id = str(uuid.uuid4())

# Used in logging
log_event("action_completed", details, trace_id=trace_id)
```

---

3. Authentication

JWT Authentication ✅

Dual Algorithm Support:

1. **Outgoing (HS256):**
   - Used when calling Registry
   - Symmetric key signing
   - Token from Registry endpoint
2. **Incoming (RS256):**
   - Used when receiving calls from other droplets
   - Asymmetric key verification
   - Public keys from JWKS endpoint

Implementation:

```python
```

```
# Outgoing
token = await fetch_jwt_token()  # HS256 from Registry

# Incoming
payload = jwt.decode(
    token,
    public_key,  # From JWKS
    algorithms=["RS256"],
    audience=JWT_AUDIENCE,
    issuer=JWT_ISSUER
)
```

JWKS Verification ✅

- Fetches public keys from `/.well-known/jwks.json`
- Caches keys for 24 hours
- Extracts `kid` from token header
- Matches key and verifies signature

Fallback Authentication ✅

- If JWKS unavailable, uses API_TOKEN
- Graceful degradation
- Maintains service availability

Token Format:

```
Authorization: Bearer <token>
```

---

4. Logging & Monitoring

Structured Logging ✅

Format:

```
json
```

```
{
  "timestamp": "2025-11-13T08:00:00Z",
  "level": "INFO|WARNING|ERROR",
  "message": "Description",
  "droplet_id": 4,
  "trace_id": "uuid-here",
  "additional_fields": {}
}
```

Log Levels:

- `INFO` - Normal operations
- `WARNING` - Non-critical issues
- `ERROR` - Errors requiring attention

Retention:

- Last 1000 log entries in memory
- Queryable via `/logs` endpoint
- Filterable by level and limit

Event Tracking ✅

Event Structure:

```
json
{
  "event_id": "evt_uuid",
  "event_type": "action_name",
  "timestamp": "2025-11-13T08:00:00Z",
  "droplet_id": 4,
  "trace_id": "uuid",
  "details": {}
}
```

Event Types:

- System events (startup, shutdown, config reload)
- Integration events (registration, heartbeat, JWT fetch)

- Action events (cloud resource operations)
- Communication events (message sent/received)

Retention:

- Last 100 events in memory
- Queryable via `/events` endpoint
- Filterable by type

Metrics Export ✅

Prometheus Format:

```
# HELP droplet_requests_total Total requests
# TYPE droplet_requests_total counter
droplet_requests_total{droplet_id="4"} 150

# HELP droplet_uptime_seconds Uptime
# TYPE droplet_uptime_seconds gauge
droplet_uptime_seconds{droplet_id="4"} 3600
```

Standard Metrics:

- `droplet_requests_total` - Total request count
- `droplet_errors_total` - Total error count
- `droplet_uptime_seconds` - Uptime in seconds
- `droplet_cpu_percent` - CPU usage percentage
- `droplet_memory_bytes` - Memory usage in bytes

Proof of Action ✅

Last Action Tracking:

```
json
```

```
{
  "action": "create_do_droplet_web-server",
  "timestamp": "2025-11-13T08:00:00Z",
  "trace_id": "uuid",
  "droplet_id": 4,
  "signature": "proof_hash"
}
```

Retention:

- Last 50 actions stored
- Queryable via `/proof` endpoint
- Future: BrickChain integration

---

5. Error Handling

Standard Error Format ✅

```
json

{
  "detail": "Human-readable error message"
}
```

HTTP Status Codes ✅

Code	Usage	Example
200	Success	Resource retrieved
201	Created	Resource created
204	No Content	Successful deletion
400	Bad Request	Invalid input
401	Unauthorized	Missing/invalid token
403	Forbidden	Insufficient permissions
404	Not Found	Resource doesn't exist
422	Validation Error	Schema validation failed
500	Internal Error	Server error
503	Unavailable	Service not configured

Error Logging ✅

All errors are:

- Logged with ERROR level
- Include trace_id
- Increment error counter
- Available in `/logs` endpoint

---

6. Backward Compatibility

Version Support ✅

Current Version: 1.0
Backward Compatible With: 0.9

Compatibility Strategy:

- Maintains existing endpoint signatures
- New fields added, old fields kept
- Graceful handling of missing fields
- Version negotiation via `/capabilities`

Deprecation Policy

When deprecating features:

1. Announce in `/capabilities` response
2. Maintain for 2 version cycles
3. Add deprecation warnings in responses
4. Document migration path

---

7. Registry Integration

Registration ✅

Payload Structure:

```
json
```

```
{
  "id": "drop4.fullpotential.ai",
  "host": "drop4.fullpotential.ai",
  "droplet_id": "drop4.fullpotential.ai",
  "ip": "0.0.0.0",
  "status": "active",
  "metadata": {
    "numeric_id": 4,
    "name": "Multi-Cloud Manager",
    "steward": "Hassan",
    "version": "1.0.0",
    "udc_version": "1.0",
    "capabilities": [...]
  }
}
```


Process:

1. Fetch JWT token from Registry
2. Send registration payload
3. Receive confirmation
4. Begin heartbeat cycle

Heartbeat ✅

Frequency: Every 30 seconds (configurable)

Payload:

```
json
```

```
{
  "id": "drop4.fullpotential.ai",
  "host": "drop4.fullpotential.ai",
  "droplet_id": "drop4.fullpotential.ai",
  "status": "healthy",
  "load": 0.15,
  "metadata": {
    "cpu_percent": 12.5,
    "memory_mb": 5394.10,
    "requests_last_minute": 45,
    "errors_last_minute": 0,
    "uptime_seconds": 3600
  }
}
```


Health Status:

- `healthy` - All systems operational
- `degraded` - Partial functionality
- `unhealthy` - Critical issues

---

8. Dependencies Declaration

Required Dependencies ✅

json


```
{
  "required": [
    {
      "id": 18,
      "name": "Registry v2",
      "status": "connected",
      "url": "https://drop18.fullpotential.ai"
    }
  ]
}
```


Optional Dependencies ✅

json

```
{
  "optional": [
    {
      "id": 2,
      "name": "Dashboard",
      "status": "available"
    }
  ]
}
```


---

9. Configuration Management

Hot Reload ✅

Endpoint: `POST /reload-config`

Reloadable Settings:

- Cloud provider tokens
- Heartbeat interval
- Logging levels
- Feature flags

Process:

1. Receives reload request
2. Re-reads environment variables
3. Updates in-memory configuration
4. Logs reload event
5. No downtime required

Graceful Shutdown ✅

Endpoint: `POST /shutdown`

Process:

1. Receives shutdown request

2. Stops accepting new requests
3. Completes in-flight requests
4. Sends final heartbeat
5. Closes connections
6. Exits cleanly

Timeout: 30 seconds maximum

---

10. Testing & Verification

Automated Tests ✅

```bash
# Health endpoints
pytest tests/test_health.py

# Authentication
pytest tests/test_auth.py

# Message protocol
pytest tests/test_messages.py

# Error handling
pytest tests/test_errors.py
```

Manual Verification ✅

```bash
# Run verification script
python3 test_registry_v2.py

# Check all UDC endpoints
bash verify_udc_compliance.sh
```

---

11. Compliance Checklist

Core Requirements

- [x] Health endpoint returns status + dependencies
- [x] Capabilities endpoint declares UDC version
- [x] State endpoint provides system metrics
- [x] Version endpoint includes build info
- [x] Message endpoint accepts UDC messages
- [x] Send endpoint delivers UDC messages
- [x] Dependencies endpoint lists connections
- [x] Metrics endpoint exports Prometheus format
- [x] Logs endpoint provides structured logs
- [x] Events endpoint tracks actions
- [x] Proof endpoint records last action

Authentication

- [x] JWT Bearer token authentication
- [x] JWKS public key verification
- [x] Token validation (issuer, audience, expiry)
- [x] Graceful fallback for auth failures

Observability

- [x] Structured JSON logging
- [x] Trace ID propagation
- [x] Event tracking
- [x] Metrics export
- [x] Error logging

Communication

- [x] UDC message format compliance
- [x] Message type support (event, query, command, response)
- [x] Trace ID in all messages
- [x] Registry lookup for target droplets

Reliability

- [x] Graceful shutdown
- [x] Hot configuration reload
- [x] Error recovery

- [x] Heartbeat monitoring
- [x] Dependency health checks

---

12. Certification Statement

This droplet has been tested and verified to be fully compliant with UDC v1.0 specifications.

Certified Components:

- ✅ All required endpoints implemented
- ✅ UDC message protocol adhered to
- ✅ JWT authentication operational
- ✅ Structured logging configured
- ✅ Metrics export functional
- ✅ Registry integration complete
- ✅ Error handling standardized
- ✅ Backward compatibility maintained

Certification Valid: 2025-11-13 onwards

Recertification Required: Upon UDC version upgrade

Steward: Hassan
Date: 2025-11-13
Signature: `udc_cert_droplet4_20251113`

---

13. Non-Compliance Items

None. This droplet is fully compliant with UDC v1.0.

---

14. Future UDC Enhancements

Planned for future UDC versions:

- [ ] BrickChain signature verification
- [ ] Enhanced proof-of-action with blockchain
- [ ] Distributed tracing integration

- [ ] Advanced metrics aggregation
- [ ] Multi-version protocol negotiation
- [ ] WebSocket support for real-time messages

References

- UDC v1.0 Specification
- Registry v2 Integration Guide
- JWT Authentication Standards (RFC 7519)
- Prometheus Metrics Format
- OpenTelemetry Tracing Standards