Operations Command Center - MVP Document & Development Guide

Project Overview

Module Name: Operations Command Center (ops_center)

Purpose: Unified dashboard for infrastructure monitoring and control

Integration with: Existing Life Planner application (Billas Planner 1.0 Beta)

Version: 1.0.0

Problem Statement

Currently managing 7+ different systems requiring separate logins:

- LibreNMS (monitoring)
- Hostmon (uptime tracking)
- Firewalla (security)
- OPNsense (IPS/firewall)
- MikroTik switches
- Zyxel switches
- APC PDUs (power management)

Pain Points:

- Multiple logins to check system status
- No unified view of infrastructure health
- Time wasted switching between interfaces
- Missing correlations between systems

Solution

Single dashboard that aggregates monitoring data and provides control capabilities across all systems.

Core Value: "See Everything. Control Everything. Miss Nothing."

Technical Context

Existing Application Structure

- **Framework:** Flask 3.0.0 with SQLAlchemy
- **Database:** SQLite
- **UI Style:** Dark theme with military/drill sergeant aesthetic
- **Existing Modules:** Daily planner, Equipment, Projects, Health, Financial, Network (inventory)
- **LibreNMS Integration:** Already exists in Network module (can be extended)

Important Notes from Existing Codebase

python

```python
# From config.py - LibreNMS already configured:
LIBRENMS_BASE_URL = "http://192.168.1.250"
LIBRENMS_API_TOKEN = "9c323b0b9ce872f007041870b5c2d248"
LIBRENMS_CACHE_TTL = 60

# Network module exists at /modules/network/ for inventory
# This NEW module is for operations, not documentation
```

MVP Features & User Interface

Dashboard Layout (Single Page)

```
┌─────────────────────────────────────────────────────────────┐
│              OPERATIONS COMMAND CENTER                       │
│      System Status: ✅  |  Last Refresh: 14:30:25           │
├─────────────────────────────────────────────────────────────┤
│                                                              │
│  [============= WAN 1 Bandwidth Graph =============]        │
│  Status: UP | 487↓/122↑ Mbps | Uptime: 99.2% (30d)         │
│                                                              │
│  [============= WAN 2 Bandwidth Graph =============]        │
│  Status: UP | 94↓/18↑ Mbps | Uptime: 99.8% (30d)           │
│                                                              │
├─────────────────────────────────────────────────────────────┤
│  ALERTS & INCIDENTS                                          │
│  ┌───────────┐ ┌───────────┐ ┌───────────┐ ┌───────────┐   │
│  │IPS ALERTS │ │FIREWALLA  │ │SYS LOGS   │ │HOSTMON    │   │
│  │   🔴 3    │ │   🟡 1    │ │   🟢 0    │ │ WAN1:     │   │
│  │SSH Brute  │ │Port Scan  │ │All Clear  │ │4min @3am  │   │
│  └───────────┘ └───────────┘ └───────────┘ └───────────┘   │
├─────────────────────────────────────────────────────────────┤
│  POWER & CONTROL                                             │
│  ┌──────────────────────┐ ┌──────────────────────┐         │
│  │ PDU-1 (8.2A/15A)     │ │ SWITCH MANAGEMENT    │         │
│  │ [1][2][❌][4][5]     │ │ MikroTik: 24 ports   │         │
│  │ [6][7][8][9][10]     │ │ Zyxel-1: 48 ports    │         │
│  │ Click to toggle      │ │ Zyxel-2: 24 ports    │         │
│  └──────────────────────┘ └──────────────────────┘         │
└─────────────────────────────────────────────────────────────┘
```

MVP Feature Set (Prioritized)

Phase 1 - Core Dashboard (Week 1-2)

- [ ] Unified dashboard with all sections
- [ ] Mock data for all components
- [ ] Auto-refresh every 30 seconds
- [ ] Dark theme matching Life Planner

Phase 2 - Real Data (Week 3-4)

- [ ] LibreNMS integration (extend existing)
- [ ] Hostmon uptime history
- [ ] OPNsense IPS alerts
- [ ] APC PDU status display
- [ ] Real metrics replacement

Phase 3 - Control Features (Week 5-6)

- [ ] PDU outlet on/off control
- [ ] Alert acknowledgment
- [ ] Basic switch port info
- [ ] Error handling

Technical Architecture

Module Structure (CRITICAL: Keep files under 300 lines!)

```
/modules/ops_center/
├── __init__.py              # Module initialization
├── models/
│   ├── __init__.py          # Empty
│   └── credentials.py       # Encrypted credential storage
├── routes/
│   ├── __init__.py          # Route registration
│   ├── dashboard.py         # Main dashboard (max 300 lines)
│   ├── api.py               # AJAX endpoints
│   ├── power.py             # PDU control
│   ├── alerts.py            # Alert management
│   └── network.py           # Switch control
├── services/
│   ├── __init__.py          # Empty
│   └── aggregator.py        # Data aggregation
├── integrations/
│   ├── __init__.py          # Empty
│   ├── base.py              # Base class for all integrations
│   ├── librenms.py          # LibreNMS API
│   ├── hostmon.py           # Hostmon integration
│   ├── opnsense.py          # OPNsense API
│   ├── apc_pdu.py           # APC SNMP/API
│   └── firewalla.py         # Firewalla API
├── templates/ops_center/
│   └── dashboard.html       # Main dashboard template
└── static/ops_center/
    ├── css/
    │   └── dashboard.css    # Module styles
    └── js/
        └── dashboard.js     # Dashboard JavaScript
```

Development Standards

1. Version Headers (EVERY Python file)

```python
"""
Operations Command Center - [Component Name]
Version: 1.0.0
Last Modified: 2024-01-XX
Author: Billas
Description: [Brief description]

CHANGELOG:
- 1.0.0 (2024-01-XX): Initial implementation
"""
```

2. Heavy Documentation

- Every class needs docstrings
- Every method needs docstrings with Args/Returns
- Inline comments for complex logic
- Example usage where helpful

3. File Size Limits

- Routes: Max 300 lines
- Services: Max 200 lines
- Integrations: Max 300 lines
- Templates: Max 250 lines (use includes)

4. Git Commits

```bash
[FEAT] Dashboard: Add WAN graphs
[FIG] PDU: Fix outlet toggle
[DOCS] README: Update setup
```

Implementation Roadmap

Sprint 1: Foundation (Days 1-3)

Goal: Get dashboard displaying with mock data

1. Create module structure
2. Set up routes and blueprint
3. Create dashboard template
4. Add mock data endpoints
5. Test dashboard loads at `/ops`

Deliverable: Working dashboard with fake data

Sprint 2: Integration (Days 4-7)

Goal: Connect real data sources

1. Create base integration class
2. Add LibreNMS integration (extend existing)
3. Add credential storage
4. Implement caching
5. Connect one real data source

Deliverable: Dashboard with at least one live data feed

Sprint 3: Control (Days 8-10)

Goal: Add control capabilities

1. Implement PDU control
2. Add alert management
3. Basic error handling
4. Testing

Deliverable: Ability to control infrastructure from dashboard

---

API Integrations Required

System	Priority	API Type	Purpose	Data Refresh
LibreNMS	HIGH	REST API	Graphs, alerts, logs	30 sec
Hostmon	HIGH	Database/API	Uptime history	60 sec
OPNsense	HIGH	REST API	IPS alerts, WAN stats	30 sec
APC PDU	HIGH	SNMP v3	Power control & metrics	30 sec
Firewalla	MEDIUM	REST API	Security alerts	60 sec
MikroTik	LOW	RouterOS API	Switch management	5 min
Zyxel	LOW	SSH/SNMP	Switch management	5 min

Key Design Decisions

1. **Separate from Network Module:** Network module is for inventory/documentation. This is for operations.
2. **Modular Design:** No file over 300 lines. Period.
3. **Mock First:** Build with fake data, then replace with real integrations.
4. **Caching Strategy:**
   - Critical data: 30 second cache
   - Status data: 60 second cache
   - Config data: 5 minute cache
5. **Security:** All credentials encrypted in database using Fernet
6. **UI Updates:** AJAX polling, not WebSockets (simpler)

Database Schema

ops_credentials

```sql
CREATE TABLE ops_credentials (
    id INTEGER PRIMARY KEY,
    integration_name VARCHAR(50),
    credential_name VARCHAR(100),
    encrypted_data TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    last_used DATETIME,
    created_at DATETIME,
    updated_at DATETIME
);
```

ops_cache

```sql
CREATE TABLE ops_cache (
    id INTEGER PRIMARY KEY,
    cache_key VARCHAR(255) UNIQUE,
    integration_name VARCHAR(50),
    data_type VARCHAR(50),
    cached_data JSON,
    expires_at DATETIME,
    hit_count INTEGER DEFAULT 0
);
```

Success Metrics

- **Zero** separate system logins needed daily
- **< 5 seconds** to assess infrastructure health
- **< 30 seconds** to respond to issues
- **90% reduction** in time gathering information

Session Recovery Instructions

For New Claude Session:

```
I need to build the Operations Command Center module for my Life Planner app.

Context:
- Existing Flask app with dark military theme
- Located at /modules/ops_center/
- Must integrate with existing LibreNMS config
- Files must stay under 300 lines
- Need heavy documentation

Please review this MVP document and help me build this module step by step, starting with the basic dashboard structure.

Current priority: [Insert what you're working on]
```

Quick Reference

- **URL:** `/ops` (after blueprint registration)
- **Main Template:** `ops_center/templates/ops_center/dashboard.html`
- **Config:** LibreNMS creds in `config.py`
- **Existing Example:** Check `/modules/network/` for patterns

Development Checklist

Initial Setup

- [ ] Create module directory structure
- [ ] Create **init**.py files
- [ ] Set up blueprint registration
- [ ] Update app.py to include module
- [ ] Create database tables
- [ ] Test dashboard loads

Dashboard Development

- [ ] Create HTML template
- [ ] Add CSS styling
- [ ] Implement auto-refresh
- [ ] Create mock API endpoints
- [ ] Add loading states
- [ ] Test all components display

Integration Development

- [ ] Build base integration class
- [ ] Add credential management
- [ ] Implement caching
- [ ] Create first integration (LibreNMS)
- [ ] Test live data flow
- [ ] Add error handling

Control Features

- [ ] PDU outlet control
- [ ] Alert acknowledgment
- [ ] Switch port info
- [ ] Action confirmation dialogs
- [ ] Audit logging
- [ ] Test all controls

Notes for Developer

1. **Start Simple:** Get dashboard working with mock data first
2. **One Integration at a Time:** Don't try to add all systems at once
3. **Test Often:** Each sprint should have a working deliverable
4. **Document Everything:** Future you will thank current you
5. **Keep it Modular:** Resist the urge to create large files

---

Questions Resolved from Discussion

- **Q:** Should this extend the Network module?
  - **A:** No, Network is for inventory. This is for operations.
- **Q:** How to handle multiple credentials?
  - **A:** Encrypted storage in database using Fernet.
- **Q:** Real-time updates?
  - **A:** AJAX polling every 30-60 seconds, not WebSockets.
- **Q:** Which integration first?
  - **A:** LibreNMS since it's already configured.

This document represents the complete MVP specification for the Operations Command Center module. Build incrementally, test frequently, and maintain the modular architecture throughout development.