

Bacula Enterprise Mock Server

REST API Simulation for Development, Testing, and Training

Functionality, Configuration & Implementation Guide

Version 2.0.0 | February 2026

A complete simulation of the Bacula Enterprise REST API. Provides a fully functional API server — same endpoints, same response formats, same authentication — without requiring any actual Bacula infrastructure. Generates realistic backup data across 30+ client profiles with configurable error rates, TLS support, OAuth2 authentication, a CLI tool, and 86 automated API tests.

Server	Python 3.8+, systemd service, auto-restart
API	100% wire-compatible with Bacula Enterprise REST API
Profiles	30+ types: mail, DB, NFS, VMware, K8s, SAP HANA, Windows, IoT
Auth	HTTP Basic Auth and OAuth2 tokens (12s TTL)
Testing	86 automated tests (Go binary), CLI smoke-test, load testing
Security	TLS/HTTPS with auto-generated 4096-bit RSA certificates

Copyright © 2026 faaleoleo — License: BSD 2-Clause

“Bacula” is a registered trademark of Bacula Systems SA. This software is not affiliated with or endorsed by Bacula Systems SA.

Table of Contents

- 1. Introduction
- 2. Prerequisites
- 3. Installation
- 4. BWeb Integration Installer
- 5. Configuration Reference
- 6. Client Profiles
- 7. Authentication
- 8. TLS Configuration
- 9. CLI Tool Reference
- 10. REST API Endpoints
- 11. Usage Workflows
- 12. Error Analysis
- 13. Frontend Integration
- 14. Load Testing
- 15. External Test Client
- 16. Service Management
- 17. Upgrading and Uninstallation
- 18. Troubleshooting
- 19. File Layout
- Appendix A: Quick-Start Checklist
- Appendix B: CLI Command Reference
- Appendix C: API Endpoint Summary
- Appendix D: Byte Size Conversion Table

Version: 2.0.0 **Date:** February 2026

1. Introduction

1.1 Purpose

The Bacula Enterprise Mock Server is a complete simulation of the Bacula Enterprise REST API. It provides a fully functional API server that behaves identically to a real Bacula Enterprise Director — same endpoints, same response formats, same authentication — but without requiring any actual Bacula infrastructure.

The server generates realistic backup data across 30+ client profiles (mail servers, databases, NFS, VMware ESXi, Kubernetes, SAP HANA, Windows servers, and more), supports configurable error rates, and provides both a CLI tool and an external Go test client with 86 automated tests.

1.2 Use Cases

- **Dashboard development** — build monitoring dashboards against a realistic API without a production Director
- **CI/CD integration testing** — run 86 automated API tests against all endpoints in seconds
- **Administrator training** — learn the Bacula REST API without risk to production systems
- **QA testing** — simulate disaster scenarios with configurable error rates (0% to 100%)
- **Performance testing** — benchmark API clients with `ab`, `wrk`, or `hey`
- **BWeb development** — integrate with BWeb Management Suite via `lighttpd` reverse proxy
- **Demo environments** — showcase Bacula API capabilities to customers and partners

1.3 Key Capabilities

- **100% wire-compatible** with the Bacula Enterprise REST API
- **30+ client profiles** — mail, database, NFS, web, Windows, VMware, Kubernetes, SAP HANA, IoT
- **HTTP and HTTPS (TLS)** with auto-generated self-signed certificates
- **OAuth2 and HTTP Basic Auth** authentication
- **Configurable error rates** per initialization (0.0 to 1.0)
- **Configurable client profiles** with realistic file counts, byte sizes, job levels, and error messages
- **CLI tool** (`bacula-mock`) for server management, data queries, exports, and smoke tests
- **External Go test client** with 86 tests across 13 test groups
- **Systemd service** with auto-start, auto-restart, and `journald` logging
- **BVFS browsing** — file-level restore browsing endpoints
- **Data export** — CSV and JSON export of jobs and errors

2. Prerequisites

2.1 System Requirements

Component	Requirement	Notes
Operating System	Debian 11+, Ubuntu 20.04+, RHEL 8+, Fedora 35+, macOS 11+	systemd recommended
Python	3.8 or newer	For the mock server
Disk Space	~100 MB	Server, venv, data
Memory	512 MB minimum	1 GB recommended for 500+ clients
Network	Port 9101 available	Bacula Director standard port

2.2 Automatically Installed Dependencies

The installer handles all dependencies: `python3`, `python3-venv`, `curl`, `pip`, and Python packages (`pyyaml`, etc.).

2.3 BWeb Integration (Optional)

For `install_bweb_mock.sh`: Debian 11+ or Ubuntu 22.04+, root access, active Bacula Enterprise subscription, internet access.

3. Installation

3.1 Automated Installation

```
sudo ./install.sh           # As root, to /opt/bacula-mock
./install.sh                # As user, to ~/bacula-mock
./install.sh -d /srv/bacula-mock  # Custom directory
```

3.2 Non-Interactive Installation

```
sudo ./install.sh -y           # Accept all defaults
sudo ./install.sh -y --tls      # With TLS enabled
sudo ./install.sh -y -s         # Without systemd service
sudo ./install.sh -y -p 8080    # Custom port
```

3.3 Installer Flags

Flag	Description
-h, --help	Show help
-y, --yes	Non-interactive mode
-s, --skip-systemd	Skip systemd service
-p PORT, --port PORT	Custom port (default: 9101)
-d DIR, --dir DIR	Custom install directory
-t, --tls	Enable TLS with self-signed certificate
-u, --upgrade	Upgrade mode (preserves config)
-v, --verbose	Verbose output

3.4 Manual Installation

```
INSTALL_DIR="/opt/bacula-mock"
mkdir -p $INSTALL_DIR && cd $INSTALL_DIR
sudo apt install python3-venv
python3 -m venv venv
source venv/bin/activate
pip install pyyaml
sudo ln -sf $INSTALL_DIR/scripts/bacula-mock /usr/local/bin/bacula-mock
```

3.5 Post-Installation Verification

```
sudo systemctl start bacula-mock
curl http://localhost:9101/mock/health
bacula-mock init 100 0.25
bacula-mock smoke-test
```

4. BWeb Integration Installer

4.1 Purpose

The `install_bweb_mock.sh` script automates the complete setup of BWeb Management Suite connected to the Mock Server. It installs BWeb from the Bacula Enterprise repository, deploys the mock server, configures a lighttpd reverse proxy, and initializes test data.

4.2 Steps Performed

1. Prompts for Bacula Enterprise customer credentials
2. Adds the BWeb APT repository and installs packages
3. Clones and installs the Bacula Mock Server

- 4. Configures lighttpd as a reverse proxy (port 9180 to mock API on 9101)
- 5. Writes `bweb.conf` for mock server integration
- 6. Initializes mock data and verifies all endpoints

4.3 Usage

```
chmod +x install_bweb_mock.sh
sudo ./install_bweb_mock.sh
```

Requires: Debian 11+ or Ubuntu 22.04+, root access, active Bacula Enterprise subscription.

5. Configuration Reference

5.1 File Location and Structure

Main configuration: `$INSTALL_DIR/config.yaml` (default: `/opt/bacula-mock/config.yaml`). Every change requires a server restart followed by data re-initialization.

```
server:           # Network, port, TLS
authentication:   # API credentials
simulation:       # Default data generation settings
client_profiles:  # System types to simulate
job_defaults:     # Default values for generated jobs
logging:          # Log file and level
```

5.2 Server Section

Field	Type	Default	Description
host	string	0.0.0.0	Bind address (127.0.0.1 for local only)
port	integer	9101	TCP port (Bacula Director standard)
debug	boolean	false	Enable debug logging
tls.enabled	boolean	false	Serve API over HTTPS
tls.cert_file	string	certs/server.crt	TLS certificate path
tls.key_file	string	certs/server.key	TLS private key path

5.3 Authentication Section

Field	Type	Default	Description
-------	------	---------	-------------

username	string	admin	Username for Basic Auth and OAuth2
password	string	bacula2026	Password for both auth methods

5.4 Simulation Section

Field	Type	Default	Description
default_clients	integer	100	Default client count for initialization
error_rate	float	0.25	Job failure probability (0.0 to 1.0)
job_interval_min	integer	5	Reserved for client simulator
job_interval_max	integer	30	Reserved for client simulator
statistics_interval	integer	60	Reserved for client simulator

The `error_rate` can be overridden during initialization: `bacula-mock init 50 0.05`.

5.5 Job Defaults Section

Field	Type	Default	Description
poolid	integer	1	Default pool ID
pool	string	Default	Default pool name
storageid	integer	1	Default storage ID
storage	string	File	Default storage name
type	string	B	Job type (B=Backup, R=Restore, V=Verify)

5.6 Logging Section

Field	Type	Default	Description
level	string	INFO	Log level: DEBUG, INFO, WARNING, ERROR
file	string	logs/bacula_mock.log	Log file path

max_bytes	integer	10485760	Max file size before rotation (10 MB)
backup_count	integer	5	Number of rotated log files

6. Client Profiles

6.1 Profile Structure

Each profile under `client_profiles:` defines a system type. This is the most important configuration section.

Field	Required	Description
count	Yes	Number of clients of this type
name_prefix	Yes	Hostname prefix (e.g., db-server creates db-server-001)
fdversion	No	Bacula FD version (default: 13.0.2)
uname	No	OS identification string
typical_files_min/max	Yes	File count range per Full backup
typical_bytes_min/max	Yes	Data size range in bytes per Full backup
job_levels	Yes	Probability distribution: F, I, D (must sum to 1.0)
error_types	Yes	List of realistic error messages (3-7 recommended)

6.2 Job Level Factors

For non-Full backups, file counts and byte sizes are scaled:

Level	Factor	Example (10,000-50,000 files)
Full (F)	1.0	10,000-50,000 files
Differential (D)	0.3	3,000-15,000 files
Incremental (I)	0.1	1,000-5,000 files

Failed jobs are additionally multiplied by a random factor between 0.2 and 0.8.

6.3 Built-In Profiles

Profile	Count	Prefix	Typical Size	Description
mail_server	20	mail-server	5-20 GB	Email servers
database_server	15	db-server	20-100 GB	PostgreSQL, MySQL
nfs_server	10	nfs-server	50-300 GB	NFS file servers
linux_server	30	linux-server	2-20 GB	General Linux
windows_server	15	win-server	10-50 GB	Windows Server
web_server	10	web-server	3-15 GB	Apache, Nginx
sap_hana	5	sap-hana	500 GB-1 TB	SAP HANA databases
kubernetes_node	25	k8s-node	5-50 GB	Kubernetes nodes
vmware_esxi	6	esxi-host	100 GB-1 TB	VMware hypervisors

6.4 Adding Custom Profiles

Add a new entry under `client_profiles`: with a unique key name:

```
my_custom_system:
  count: 10
  name_prefix: "custom-sys"
  fdversion: "13.0.2"
  uname: "Linux custom 6.1.0-23-amd64 x86_64"
  typical_files_min: 5000
  typical_files_max: 25000
  typical_bytes_min: 1073741824
  typical_bytes_max: 10737418240
  job_levels:
    F: 0.15
    I: 0.65
    D: 0.20
  error_types:
    - "Custom error message one"
    - "Custom error message two"
    - "Custom error message three"
```

After adding profiles, restart the server and re-initialize data.

7. Authentication

7.1 HTTP Basic Auth

```
curl -u admin:bacula2026 http://localhost:9101/cat/Client
```

7.2 OAuth2 Tokens

```
TOKEN=$(curl -s -X POST http://localhost:9101/oauth/token \
-H "Content-Type: application/json" \
-d '{"username":"admin","password":"bacula2026"}' \
| python3 -c "import json,sys; print(json.load(sys.stdin)['access_token'])")

curl -H "Authorization: Bearer $TOKEN" http://localhost:9101/cat/Client
```

Tokens are valid for 12 seconds (Bacula Enterprise default). The `/mock/health` endpoint requires no authentication.

8. TLS Configuration

8.1 Enable During Installation

```
sudo ./install.sh -y --tls
```

Generates a 4096-bit RSA key and self-signed X.509 certificate (365 days) with SANs for hostname, localhost, and 127.0.0.1.

8.2 Enable After Installation

```
openssl req -x509 -newkey rsa:4096 \
-keyout certs/server.key -out certs/server.crt \
-days 365 -nodes -subj "/CN=$(hostname)" \
-addext "subjectAltName=DNS:$(hostname),DNS:localhost,IP:127.0.0.1"
chmod 600 certs/server.key
```

Set `tls.enabled: true` in `config.yaml`. Add `BACULA MOCK_TLS=true` to `.env`. Restart the service.

8.3 Using the API with TLS

```
curl -k https://localhost:9101/mock/health
curl -k -u admin:bacula2026 https://localhost:9101/cat/Client
```

The CLI handles TLS automatically when `BACULA MOCK_TLS=true` is set.

8.4 CA-Signed Certificates

Replace `certs/server.crt` and `certs/server.key` with your CA-signed files, `chmod 600` the key, and restart.

8.5 Disabling TLS

Set `tls.enabled: false` in `config.yaml` and restart the service.

9. CLI Tool Reference

The `bacula-mock` CLI reads settings from `.env` or environment variables. Add `--raw` to any command for unformatted JSON output.

9.1 Server Management

Command	Description
<code>bacula-mock start</code>	Start the server
<code>bacula-mock health</code>	Health check (no auth)
<code>bacula-mock init [N] [rate]</code>	Generate N clients with error rate
<code>bacula-mock reset</code>	Delete all data

9.2 Querying Data

Command	Description
<code>bacula-mock clients [limit]</code>	List clients
<code>bacula-mock client <name></code>	Client catalog data
<code>bacula-mock client-status <name></code>	Client FD status
<code>bacula-mock jobs [limit]</code>	List recent jobs
<code>bacula-mock jobs-failed [limit]</code>	List failed jobs only
<code>bacula-mock jobs-for <clientid> [limit]</code>	Jobs for a specific client
<code>bacula-mock jobtotals</code>	Aggregate statistics
<code>bacula-mock joblog <jobid></code>	Job log messages
<code>bacula-mock dir-status</code>	Director status

9.3 Job and Volume Commands

Command	Description
<code>bacula-mock run <job> --level F --client <n></code>	Start a backup job
<code>bacula-mock restore --jobid <id> --where /path</code>	Restore from backup
<code>bacula-mock estimate --client <n> --level F</code>	Estimate job size

<code>bacula-mock cancel --jobid <id></code>	Cancel a running job
<code>bacula-mock label --volume <n> --pool <pool></code>	Label a new volume
<code>bacula-mock purge --volume <n></code>	Purge volume records
<code>bacula-mock prune --client <n></code>	Prune expired jobs

9.4 Export and Testing

Command	Description
<code>bacula-mock export-csv</code>	Export all jobs as CSV
<code>bacula-mock export-errors</code>	Export failed jobs as JSON
<code>bacula-mock smoke-test</code>	Run all 24 endpoint tests
<code>bacula-mock token</code>	Request OAuth2 token
<code>bacula-mock config-clients</code>	Client resource configuration

10. REST API Endpoints

10.1 Catalog Endpoints

Endpoint	Method	Description
<code>/cat/Client</code>	GET	List clients (?limit=, ?offset=, ?clientid=)
<code>/cat/Job</code>	GET	List jobs (?limit=, ?offset=, ?jobstatus=)
<code>/cat/JobTotals</code>	GET	Aggregate job statistics
<code>/cat/JobLog</code>	GET	Job log messages (?jobid=)
<code>/cat/Pool</code>	GET	List storage pools
<code>/cat/Storage</code>	GET	List storage devices

10.2 Command Endpoints

Endpoint	Method	Description
<code>/cmd/run</code>	POST	Start a backup job

/cmd/cancel	POST	Cancel a running job
/cmd/restore	POST	Restore from backup
/cmd/estimate	POST	Estimate backup size
/cmd/label	POST	Label a new volume
/cmd/update	POST	Update pool settings
/cmd/purge	POST	Purge volume records
/cmd/prune	POST	Prune expired jobs
/cmd/list	GET	List jobs or clients

10.3 Status, Resource, and Mock Endpoints

Endpoint	Auth	Method	Description
/status/director	Yes	GET	Director daemon status
/status/client/	Yes	POST	Client FD status (?name=)
/res/director	Yes	GET	Director resource config (?resource=Client)
/res/filedaemon	Yes	GET	FD resource config
/res/storagedaemon	Yes	GET	SD resource config
/mock/health	No	GET	Health check
/mock/initialize	Yes	POST	Generate test data
/mock/reset	Yes	POST	Delete all data
/oauth/token	No	POST	Request OAuth2 token

11. Usage Workflows

11.1 Initialize and Verify

```
bacula-mock start
bacula-mock init 100 0.25
bacula-mock jobtotals
bacula-mock clients 10
bacula-mock jobs-failed 10
```

11.2 Client Detail Investigation

```

bacula-mock client mail-server-001
bacula-mock client-status mail-server-001
bacula-mock jobs-for 1 50
bacula-mock joblog 1

```

11.3 Test Scenarios

Scenario	Command
Clean data (no errors)	<code>bacula-mock reset && bacula-mock init 10 0.0</code>
Disaster simulation (90% errors)	<code>bacula-mock reset && bacula-mock init 100 0.90</code>
High load testing	<code>bacula-mock reset && bacula-mock init 500 0.25</code>
Minimal test set	<code>bacula-mock reset && bacula-mock init 5 0.50</code>

11.4 Data Export

```

bacula-mock export-csv > jobs.csv
bacula-mock export-errors > failed.json

```

11.5 Programmatic Processing

Use `--raw` to get JSON output for piping:

```

bacula-mock jobs-failed 50 --raw | python3 -c "
import json, sys
jobs = json.load(sys.stdin)['data']
print(f'Failed: {len(jobs)}')
for j in jobs:
    print(f'  Job {j["jobid"]:4d}  Errors: {j["joberrors"]}')
"

```

11.6 Continuous Monitoring Script

```

while true; do
    clear; date
    bacula-mock jobtotals --raw | python3 -c "
import json, sys
t = json.load(sys.stdin)['data'][0]
print(f'Jobs: {t["jobs"]}  Files: {t["files"]},  Errors: {t["joberrors"]}')
"
    bacula-mock jobs-failed 5
    sleep 30
done

```

11.7 Job and Volume Management

```
bacula-mock run backup-mail-001-full --level F --client mail-server-001
bacula-mock restore --jobid 42 --where /tmp/restore
bacula-mock estimate --client mail-server-001 --level F
bacula-mock cancel --jobid 99
bacula-mock label --volume Vol-0001 --pool Default
bacula-mock purge --volume Vol-0001
bacula-mock prune --client mail-server-001
```

12. Error Analysis

12.1 Error Report by Client

```
bacula-mock jobs-failed 100 --raw | python3 -c "
import json, sys
from collections import defaultdict
jobs = json.load(sys.stdin)['data']
by_client = defaultdict(list)
for j in jobs: by_client[j['clientid']].append(j)
for cid, cjobs in sorted(by_client.items()):
    total_err = sum(j['joberrors'] for j in cjobs)
    print(f'Client {cid}: {len(cjobs)} jobs, {total_err} errors')
"
```

12.2 Search Job Logs for Error Messages

```
bacula-mock jobs-failed 20 --raw \
| python3 -c "import json,sys; [print(j['jobid']) for j in
json.load(sys.stdin)['data']]" \
| while read JOBID; do
    bacula-mock joblog "$JOBID" --raw | python3 -c "
import json,sys
for l in json.load(sys.stdin)['data']:
    if 'ERROR' in l['logtext'] or 'Fatal' in l['logtext']:
        print(f'  Job $JOBID: {l["logtext"]}')
    done
```

12.3 Success Rate per Client

```
bacula-mock jobs 10000 --raw | python3 -c "
import json, sys
from collections import defaultdict
jobs = json.load(sys.stdin)['data']
stats = defaultdict(lambda: {'ok': 0, 'fail': 0})
for j in jobs:
    if j['jobstatus'] == 'T': stats[j['clientid']]['ok'] += 1
    else: stats[j['clientid']]['fail'] += 1
for cid, s in sorted(stats.items()):
    total = s['ok'] + s['fail']
```

```
rate = s['ok'] / total * 100 if total else 0
print(f' Client {cid:3d}: {rate:5.1f}%  ({s["ok"]}/{total})')
"
```

12.4 Find Largest Backups

```
bacula-mock jobs 10000 --raw | python3 -c "
import json, sys
jobs = sorted(json.load(sys.stdin)['data'], key=lambda j: j['jobbytes'],
reverse=True)
for j in jobs[:10]:
    print(f' Job {j["jobid"]:4d}  {j["jobbytes"]/1024**3:>8.2f}
GB  {j["jobfiles"]:>8,} files')
"
```

13. Frontend Integration

13.1 JavaScript / Fetch

```
const BASE = 'http://localhost:9101';
const AUTH = 'Basic ' + btoa('admin:bacula2026');

async function getClients(limit = 50) {
  const resp = await fetch(
    BASE + '/cat/Client?limit=' + limit,
    { headers: { 'Authorization': AUTH } }
  );
  return (await resp.json()).data;
}
```

13.2 Python / requests

```
import requests
BASE = 'http://localhost:9101'
AUTH = ('admin', 'bacula2026')
clients = requests.get(f'{BASE}/cat/Client', auth=AUTH).json()['data']
failed = requests.get(f'{BASE}/cat/Job?jobstatus=E', auth=AUTH).json()['data']
```

13.3 Swagger and ReDoc

Interactive API documentation: <http://localhost:9101/docs> (Swagger) and <http://localhost:9101/redoc>.

14. Load Testing

14.1 Apache Bench

```
ab -n 1000 -c 10 \  
-H "Authorization: Basic $(echo -n admin:bacula2026 | base64)" \  
"http://localhost:9101/cat/Job?limit=10"
```

14.2 wrk

```
wrk -t4 -c50 -d30s \  
-H "Authorization: Basic YWRtaW46YmFjdWxhMjAyNg==" \  
"http://localhost:9101/cat/Client?limit=10"
```

14.3 hey

```
hey -n 1000 -c 50 \  
-H "Authorization: Basic YWRtaW46YmFjdWxhMjAyNg==" \  
"http://localhost:9101/cat/Job?limit=5"
```

15. External Test Client

15.1 Overview

A compiled Go binary (`bacula_mock_test`) running 86 automated tests across 13 groups.

15.2 Usage

```
./bacula_mock_test -h localhost -p 9101  
./bacula_mock_test -h localhost -j > results.json  
./bacula_mock_test -h localhost -q
```

15.3 Test Groups

Group	Tests	Coverage
Connectivity and Health	2	/mock/health
Authentication	7	OAuth2, Bearer, Basic Auth, 401 rejection
Data Generation	2	Reset, initialize
Catalog Endpoints	18	All /cat/ endpoints, 28-field validation
Pagination	3	Limit/offset, no duplicates
Data Types	6	Integer, string, datetime validation

Command Endpoints	19	Run, cancel, restore, estimate, label, update, purge, prune
Status Endpoints	4	Director, client FD, storage
Resource Endpoints	4	Director, FD, SD resources
BVFS	7	File-level restore browsing
Configuration	8	Read/write 5 resource types
Error Handling	5	404, 400, edge cases
Performance	2	Latency, throughput

15.4 CI/CD Integration

```
./bacula_mock_test -h $SERVER_HOST -j > test-results.json
if [ $? -ne 0 ]; then
    echo "Tests failed!"
    exit 1
fi
```

16. Service Management

16.1 Systemd Commands

```
sudo systemctl start bacula-mock
sudo systemctl stop bacula-mock
sudo systemctl restart bacula-mock
sudo systemctl status bacula-mock
sudo systemctl enable bacula-mock
```

16.2 Log Files

```
sudo journalctl -u bacula-mock -f
tail -f /opt/bacula-mock/logs/bacula_mock.log
cat /opt/bacula-mock/installation.log
```

16.3 Firewall

```
sudo ufw allow 9101/tcp
sudo firewall-cmd --permanent --add-port=9101/tcp && sudo firewall-cmd --reload
```

16.4 Multiple Instances

```
sudo ./install.sh -d /opt/bacula-mock-1 -p 9101
sudo ./install.sh -d /opt/bacula-mock-2 -p 9102
```

17. Upgrading and Uninstallation

17.1 Upgrade

```
cd /path/to/new-release
./install.sh --upgrade
sudo systemctl restart bacula-mock
```

Preserves config.yaml, logs/, data/, and certs/. A timestamped backup is created in backups/.

17.2 Uninstallation

```
sudo systemctl stop bacula-mock
sudo systemctl disable bacula-mock
sudo rm /etc/systemd/system/bacula-mock.service
sudo systemctl daemon-reload
sudo rm -rf /opt/bacula-mock
sudo rm -f /usr/local/bin/bacula-mock
```

18. Troubleshooting

18.1 Server Won't Start

Check `sudo systemctl status bacula-mock` and `sudo journalctl -u bacula-mock -n 50`.
Common causes: port in use, config syntax error, missing TLS files.

18.2 Port Already in Use

Run `sudo lsof -i :9101` to find the conflicting process.

18.3 TLS Certificate Errors

Check `ls -la certs/`, verify with `openssl x509 -in certs/server.crt -noout -dates`, regenerate if needed.

18.4 CLI Tool Not Found

Verify: `ls -la /usr/local/bin/bacula-mock`. If missing: `sudo ln -sf /opt/bacula-mock/scripts/bacula-mock /usr/local/bin/bacula-mock`.

18.5 Virtual Environment Issues

```
cd /opt/bacula-mock && rm -rf venv
python3 -m venv venv && source venv/bin/activate
```

```
pip install -r requirements.txt
sudo systemctl restart bacula-mock
```

19. File Layout

```
/opt/bacula-mock/
+-- config.yaml           Main configuration
+-- config.yaml.dist      Default configuration reference
+-- .env                  Environment variables for CLI
+-- requirements.txt       Python dependencies
+-- src/                  Server source code
+-- scripts/
|   +-- bacula-mock        CLI tool
|   +-- start_server.sh    Manual start script
+-- certs/                TLS certificates (if enabled)
|   +-- server.crt
|   +-- server.key
+-- venv/                  Python virtual environment
+-- data/                  Generated test data
+-- logs/
|   +-- bacula_mock.log    Application log
+-- backups/              Upgrade backups
+-- installation.log       Installer log
+-- tests/
|   +-- bacula_mock_test   Go test binary
```

Appendix A: Quick-Start Checklist

Step	Command
Install	<code>sudo ./install.sh</code>
Install with TLS	<code>sudo ./install.sh --tls</code>
Start service	<code>sudo systemctl start bacula-mock</code>
Health check	<code>bacula-mock health</code>
Generate data	<code>bacula-mock init 100 0.25</code>
List clients	<code>bacula-mock clients</code>
List failed jobs	<code>bacula-mock jobs-failed</code>
Smoke test	<code>bacula-mock smoke-test</code>
View logs	<code>sudo journalctl -u bacula-mock -f</code>

Appendix B: CLI Command Reference

Command	Description
start	Start the mock server
health	Health check (no auth)
init [N] [rate]	Generate N clients with error rate
reset	Delete all generated data
clients [limit]	List clients
client <name>	Client catalog data
client-status <name>	Client FD live status
jobs [limit]	List recent jobs
jobs-failed [limit]	List failed jobs only
jobs-for <clientid> [limit]	Jobs for a specific client
jobtotals	Aggregate statistics
joblog <jobid>	Job log messages
dir-status	Director daemon status
config-clients	Client resource configuration
token	Request OAuth2 token
run <job>	Start a backup job
restore	Restore from backup
estimate	Estimate job size
cancel	Cancel a running job
label	Label a new volume
update-pool	Update pool settings
purge	Purge volume records
prune	Prune expired jobs
export-csv	Export jobs as CSV
export-errors	Export failed jobs as JSON
smoke-test	Test all API endpoints

Appendix C: API Endpoint Summary

Endpoint	Auth	Method	Description
/mock/health	No	GET	Health check
/mock/initialize	Yes	POST	Generate test data
/mock/reset	Yes	POST	Delete all data
/oauth/token	No	POST	Request OAuth2 token
/cat/Client	Yes	GET	List clients
/cat/Job	Yes	GET	List jobs
/cat/JobTotals	Yes	GET	Aggregate statistics
/cat/JobLog	Yes	GET	Job log messages
/cat/Pool	Yes	GET	List pools
/cat/Storage	Yes	GET	List storage
/cmd/run	Yes	POST	Start backup
/cmd/cancel	Yes	POST	Cancel job
/cmd/restore	Yes	POST	Restore backup
/cmd/estimate	Yes	POST	Estimate size
/cmd/label	Yes	POST	Label volume
/cmd/update	Yes	POST	Update pool
/cmd/purge	Yes	POST	Purge volume
/cmd/prune	Yes	POST	Prune jobs
/cmd/list	Yes	GET	List jobs/clients
/status/director	Yes	GET	Director status
/status/client/	Yes	POST	Client FD status
/res/director	Yes	GET	Director resources
/res/filedaemon	Yes	GET	FD resources
/res/storagedaemon	Yes	GET	SD resources

Appendix D: Byte Size Conversion Table

Size	Bytes	YAML Value
100 MB	104,857,600	104857600

500 MB	524,288,000	524288000
1 GB	1,073,741,824	1073741824
5 GB	5,368,709,120	5368709120
10 GB	10,737,418,240	10737418240
50 GB	53,687,091,200	53687091200
100 GB	107,374,182,400	107374182400
500 GB	536,870,912,000	536870912000
1 TB	1,099,511,627,776	1099511627776

Quick formula: Gigabytes x 1,073,741,824 = Bytes

"Bacula" is a registered trademark of Bacula Systems SA. This software is not affiliated with or endorsed by Bacula Systems SA.

"Bacula" is a registered trademark of Bacula Systems SA. This software is not affiliated with or endorsed by Bacula Systems SA.