Documentation: Proxmox Device Mapper Analysis and Cleanup Script (Version 27)

Overview

The **Proxmox Device Mapper Analysis and Cleanup Script v27** is a comprehensive Bash-based tool designed to assist system administrators in identifying and optionally removing stale device-mapper (DM) entries on a Proxmox Virtual Environment (PVE) host. **NEW in v27:** The script now includes a **Config Validation Module** that cross-references VM configurations against device mapper entries to detect configuration mismatches, duplicates, and orphaned entries that could prevent VM startup.

The script performs real-time analysis, generates detailed HTML reports with system health metrics and config validation results, and optionally delivers these reports via Mailjet email API. It also includes an enhanced interactive cleanup mode for safe manual removal of stale entries and configuration mismatches.

Key Features

Core Analysis Features

- Analyzes all DM entries related to VM disks
- Identifies and reports stale entries (DM devices referencing non-running VMs)
- **NEW: Config Validation Module -** Compares VM configurations vs actual device mapper entries
- **NEW: Enhanced Disk Type Support -** Handles EFI disks, TPM disks, unused disks, and standard VM disks
- Provides comprehensive summary of valid, stale, and mismatched entries

Config Validation Features (New in v27)

- **Orphaned Entry Detection**: Device mapper entries that don't match any VM configuration
- **Duplicate Entry Detection**: Multiple device mapper entries for the same logical disk
- **Missing Entry Detection**: VM configuration expects disk but no device mapper entry exists
- **Cross-Reference Validation**: Ensures running VMs have proper DM entries matching their configs

Reporting and Monitoring

- Gathers extensive host health and performance metrics
- Grades system health (A+ to D) based on performance, stale entries, and config issues
- Generates professional HTML email reports with dynamic visual indicators

- **NEW: Config validation results** prominently displayed in reports
- Sends reports using the Mailjet email API

Interactive Cleanup

- Enhanced interactive cleanup mode for safe removal
- **NEW: Config-aware cleanup** with detailed explanations
- User confirmation for each issue type with safety information

Functions Breakdown

1. VM and DM Entry Discovery

- Lists all running VMs on the host using `qm list`
- **NEW: Lists ALL VMs** (running and stopped) for config validation
- Retrieves all current DM entries matching `vm--<VMID>--disk`

> **Important:** Only VMs that are actively running on the node are considered valid for stale entry analysis. However, ALL VMs are used for config validation to detect mismatches.

2. DM Entry Classification (Enhanced)

- Each DM entry is parsed to extract its associated VM ID
- Entry is marked as:
  - **Valid**: VM is currently running on this host AND matches VM config
  - **Stale**: VM is not running (entry is an orphaned leftover)
  - **NEW: Orphaned**: DM entry exists but doesn't match any VM configuration
  - **NEW: Duplicate**: Multiple DM entries exist for the same logical disk

3. Config Validation Module (New in v27)

VM Configuration Parsing

- Reads all VM config files from `/etc/pve/qemu-server/*.conf`
- **Enhanced parsing** supports all disk types:
  - **virtio, ide, scsi, sata**: Standard VM disks
  - **efidisk**: EFI System Disks
  - **tpmstate**: TPM State disks
  - **unused**: Unused/detached disks

- Extracts expected storage pool and disk paths for each VM

Cross-Reference Analysis

- Compares what **should** exist (from VM configs) vs what **does** exist (from dmsetup)
- Identifies several types of mismatches:
  - **Orphaned entries**: DM entries that don't match any VM config
  - **Duplicate entries**: Multiple DM entries for the same logical disk
  - **Missing entries**: VM config expects a disk but no DM entry exists
  - **Wrong entries**: DM entry exists but points to wrong storage/path

Debug and Validation Output

- Shows parsing progress and results
- Debug output for troubleshooting config issues
- Sample entries displayed for verification

4. System Metrics Collection

- Uptime, load averages, CPU usage and model, memory and swap usage
- Proxmox version, kernel version
- VM/container counts
- ZFS, LVM, network traffic, and disk usage statistics
- Boot time and top CPU-consuming processes

5. Enhanced Performance Grading

- Performance score calculated from:
  - CPU and RAM usage thresholds
  - Count of stale entries
  - **NEW: Count of config validation issues**
- Letter grade (A+ to D) with associated color code
- **Config issues now impact overall health grade**

6. Enhanced HTML Email Report Generation

- Uses a template-style output with embedded system and DM entry data
- **NEW: Config Validation Results section** with detailed breakdown

- **NEW: Issue-specific styling** (orphaned, duplicate, missing entries)
- **NEW: Color-coded metrics** showing severity of each issue type
- Styles and layout optimized for readability and mobile responsiveness
- Dynamic colors reflect health status and resource utilization

7. Email Delivery via Mailjet

- HTML content embedded in a JSON payload
- **Enhanced email subjects** include config issue counts
- Email includes subject with hostname, health grade, and total issues
- Text fallback provided for email clients without HTML support

8. Enhanced Interactive Cleanup Mode

- **Two-phase cleanup process**:
  1. **Stale DM entries** (traditional cleanup)
  2. **NEW: Orphaned config entries** (config validation cleanup)
- Prompts user for each issue with detailed explanations:
  - **(y)** remove entry
  - **(n)** skip
  - **(a)** auto-remove remaining without prompts (stale entries only)
  - **(q)** quit
- **Enhanced safety information** for each entry type before prompting
- **Config-aware explanations** showing why each entry is problematic

Installation on a Proxmox Node

1. Copy the script to the node

```bash
scp Proxmox_DM_Cleanup_v27.sh root@<node-ip>:/root/
```

2. Set execution permissions

```bash
chmod +x /root/Proxmox_DM_Cleanup_v27.sh
```

3. Run the script

```bash
./Proxmox_DM_Cleanup_v27.sh
```

Scheduling with Cron

When you run `crontab -e` as root for the first time, you may be prompted to choose an editor. We recommend selecting option `1` for `/bin/nano` as the easiest option.

> Example Prompt:

```
Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
Choose 1-3 [1]:
```

Once inside the crontab file (opened with nano), scroll past the existing commented explanation lines (which begin with `#`) to the bottom of the file. Then, on a new line, **add the following line** to run the script every night at 10 PM:

```bash
0 22 * * * /root/Proxmox_DM_Cleanup_v27.sh > /var/log/proxmox_dmcheck.log 2>&1
```

📌 This command means:

- `0` minute
- `22` hour (10 PM)
- `* * *` = every day, every month, every weekday

Save with `Ctrl+O`, press `Enter`, and exit with `Ctrl+X`.

If done correctly, this sets a scheduled job visible by running:

```bash
crontab -l
```

❌ To Remove the Cron Job Later

Edit the crontab again:

```bash
crontab -e
```

Then delete the line containing the script command, save, and exit.

Suggested Commands (Run as Root)

These are the recommended commands for installing, executing, and removing the script:

```bash
nano /root/Proxmox_DM_Cleanup_v27.sh     # Edit or paste the script
chmod +x /root/Proxmox_DM_Cleanup_v27.sh # Make it executable
./Proxmox_DM_Cleanup_v27.sh              # Run the script manually
rm /root/Proxmox_DM_Cleanup_v27.sh       # Remove the script completely
```

To completely remove the automation job and script from the system:

❌ Remove the Cron Job

```bash
crontab -e
```

Then delete the line containing:

```
/root/Proxmox_DM_Cleanup_v27.sh
```

Save and exit with Ctrl+O and Ctrl+X.

🧹 Remove the Script File

```bash
rm /root/Proxmox_DM_Cleanup_v27.sh
```

Educational Note: Understanding Device Mapper Issues

Traditional Stale DM Entries

Stale device-mapper entries are remnants left on a node when a VM is:

- Migrated to another node
- Shut down without full cleanup
- Manually removed from configuration but still referenced in storage

NEW: Config Validation Issues (v27)

Orphaned DM Entries

- Device mapper entries that exist but don't correspond to any VM configuration
- **Common causes:**
  - VM was deleted but DM entries weren't cleaned up
  - Disk was removed from VM config but DM entry remains
  - Restore process corruption left mismatched entries
- **Impact:** Can prevent VMs from starting due to name conflicts

Duplicate DM Entries

- Multiple device mapper entries pointing to the same logical disk
- **Common causes:**
  - Failed migration cleanup
  - Storage operation interruption
  - Manual storage manipulation
- **Impact:** Can cause "device busy" errors and storage conflicts

Missing DM Entries

- VM configuration expects a disk but no device mapper entry exists
- **Common causes:**
  - Normal for stopped VMs (entries cleaned up properly)
  - Storage connectivity issues
  - Incomplete VM migration
- **Impact:** Usually informational, entries recreated on VM start

⚠️ Impact of Unmaintained Issues

Traditional Stale Entries:

- Can cause **"Device or resource busy"** errors

- May prevent new or migrated VMs from starting due to name conflicts
- Consume system resources and clutter device mappings

NEW: Config Validation Issues:

- **Orphaned entries** can prevent VM startup with naming conflicts
- **Duplicate entries** cause storage access conflicts
- **Missing entries** may indicate storage connectivity problems

These entries **do not affect the actual storage data**, but keeping them around introduces unnecessary risk and operational problems.

Proxmox Behavior: What Entries Should Exist?

Traditional Understanding

Only device-mapper entries **for VMs currently running on the host** should exist. Proxmox creates these entries when starting a VM.

NEW: Config Validation Understanding (v27)

Enhanced validation ensures:

1. **Running VMs** have DM entries that match their configuration exactly
2. **No orphaned entries** exist that don't correspond to any VM config
3. **No duplicate entries** exist for the same logical disk
4. **Missing entries** are tracked for awareness (normal for stopped VMs)

How They Appear:

- When Proxmox starts a VM, it creates DM mappings for that VM's disks based on its configuration
- **Config mismatches** can occur during:
  - Failed restore operations
  - Storage migration interruptions
  - Manual configuration changes
  - Disk attachment/detachment operations

What Happens If You Remove These Entries?

Traditional Stale Entries: Removing a DM entry for a stopped VM is **safe** and will **not prevent** the VM from starting in the future.

NEW: Config Validation Entries:

- **Orphaned entries:** Safe to remove, prevents conflicts
- **Duplicate entries:** May require investigation to determine which to keep
- **Missing entries:** Cannot be "removed" - they're missing by definition

When the VM is started again, Proxmox will automatically recreate the appropriate DM entries based on its configuration and storage backend.

Do These Entries Clear After a Reboot?

No. A Proxmox node reboot does **not** automatically remove stale device-mapper entries. These entries are managed by the LVM subsystem (`lvm2`) and persist unless explicitly removed. Config validation issues also persist across reboots.

Why Proxmox Lacks Native Handling

Proxmox currently lacks built-in tooling to:

- Periodically detect and clean stale DM entries
- **Validate DM entries against VM configurations**
- **Detect and resolve config mismatches**
- Alert administrators of orphaned mappings

This script fills that critical operational gap, providing both analysis and remediation with safety checks and comprehensive config validation.

Safety Measures

Traditional Safety (v26)

- **No changes** are made during analysis unless interactive cleanup is triggered
- **Stale entries** are safe to remove as they do **not** affect actual VM disk data
- Each action is confirmed by the user or auto-confirmed only if explicitly selected

Enhanced Safety (v27)

- **Config validation** provides detailed explanations for each issue type
- **Two-phase cleanup** separates traditional stale entries from config issues
- **Enhanced user prompts** with safety information for each action
- **Debug output** allows verification of parsing accuracy

- **Orphaned entry removal** is safe and prevents future conflicts
- All sensitive operations are logged to terminal output for auditing

Dependencies

- `qm`, `dmsetup`, `pvecm`, `zpool`, `vgs`, `pct`, `curl`, `mailjet API credentials`
- Optional tools: `mpstat`, `bc`, `dmidecode`, `top`, `ps`, `uptime`, `free`, `lscpu`
- **NEW: Access to VM config files** in `/etc/pve/qemu-server/`

Usage

1. Execute Script on a Proxmox Node:

```bash
./Proxmox_DM_Cleanup_v27.sh
```

2. Review Analysis Output:

- Valid and stale DM entries
- **NEW: Config validation results** (orphaned, duplicate, missing entries)
- System metrics and performance grade

3. Receive Email Report (auto-sent)

- **Enhanced report** includes config validation section
- **Color-coded issues** for easy identification

4. (Optional) Run Interactive Cleanup

- **Phase 1:** Traditional stale entries - Choose `y`, `n`, `a`, or `q`
- **Phase 2:** Orphaned config entries - Choose `y`, `n`, or `q`

Configuration

Located near the top of the script:

```bash
```

```
MAILJET_API_KEY="<your-mailjet-api-key>"
MAILJET_API_SECRET="<your-mailjet-api-secret>"
FROM_EMAIL="automation@yourdomain.com"
FROM_NAME="ProxMox DMSetup Health Check"
TO_EMAIL="recipient@yourdomain.com"
```

Security Notes

- Mailjet API credentials are embedded in the script and should be rotated periodically
- Ensure only privileged users can access and execute the script
- **NEW: Script reads VM configuration files** - ensure appropriate permissions

File Cleanup

- Temporary files used for processing are removed on exit
- **NEW: Additional temp files** for config validation are properly cleaned up

Testing the Script

To safely generate various types of issues for testing purposes:

✅ Method 1: Live Migrate a VM (Stale Entries)

1. Start a VM on **Node A**.
2. Live-migrate the VM to **Node B**: `qm migrate <VMID> <Node-B>`
3. Check Node A: The DM entry for the VM may still be present and will be detected as stale.

✅ Method 2: Manually Create a Fake Entry (Orphaned Entries)

Create a dummy DM mapping that simulates an orphaned VM disk:

```
bash

dmsetup create test--vm--999--disk--0 --table '0 204800 linear /dev/sda 0'
```

- This entry will appear in `dmsetup ls` but has no associated VM config.
- The script will treat it as orphaned.

✅ Method 3: Config Modification (Missing Entries)

1. Stop a VM
2. Note that its DM entries are cleaned up

3. The script will detect the VM config expects disks but no DM entries exist (normal for stopped VMs)

> ⚠️ Avoid forcibly deleting actual VM storage or modifying production VMs.

Script Safety and Behavior

This script is designed with safety as the highest priority. Below is a summary of why it is considered safe for use in production environments:

✅ Safe by Design

- **Read-Only by Default:** Running the script performs analysis and reporting only. No changes are made to the system unless the user explicitly opts into interactive cleanup.
- **Enhanced User Prompts:** During cleanup, each issue is presented with detailed context and requires user input before removal.
- **What Gets Removed:** Only stale `dmsetup` table entries and orphaned config entries --- i.e., device-mapper mappings that don't serve operational purposes.
- **Actual Disk Data Is Safe:** The underlying storage (LVM, ZFS, Ceph, etc.) is untouched. These DM entries are recreated by Proxmox when needed.
- **Graceful Handling:** The script skips malformed entries, logs errors, and does not attempt unsafe operations.
- **Config Validation Safety:** Only identifies mismatches - does not modify VM configurations.

⚠️ Why Different Issue Types Matter

Stale Entries (Traditional):

- VMs not running but still have DM entries
- Safe to remove, entries recreated on VM start

Orphaned Entries (New):

- DM entries that don't match any VM configuration
- **These are the entries that cause VM startup failures**
- Safe to remove, prevents conflicts

Duplicate Entries (New):

- Multiple DM entries for same disk
- May require investigation before removal

Missing Entries (New):

- VM config expects disk but no DM entry
- Usually normal for stopped VMs
- Cannot be "cleaned up" - informational only

Version 27 Improvements Summary

🆕 New Features

- **Config Validation Module** - Comprehensive VM config vs DM entry validation
- **Enhanced Disk Type Support** - EFI, TPM, unused disks
- **Orphaned Entry Detection** - Finds entries that don't match any VM config
- **Duplicate Entry Detection** - Identifies multiple entries for same disk
- **Missing Entry Tracking** - VM config expects disk but no DM entry
- **Two-Phase Interactive Cleanup** - Separate handling for different issue types
- **Enhanced Email Reports** - Config validation results prominently displayed

🔧 Improvements

- **Better Performance Grading** - Now includes config issues in health score
- **Enhanced Safety Information** - Detailed explanations for each cleanup action
- **Debug Output** - Better troubleshooting and verification capabilities
- **Improved Error Handling** - More robust parsing and validation

🐛 Bug Fixes

- **Bash Syntax Errors** - Fixed `local` variable declarations outside functions
- **Config Parsing Issues** - Now correctly handles all Proxmox disk types
- **Compatibility Improvements** - Better support for different bash versions

Summary

This script acts as both a diagnostic tool and an optional remediation utility, offering detailed insights into the Proxmox DM state while ensuring safe and controlled cleanup operations. **Version 27's Config Validation Module** provides the missing link between VM configurations and actual device mapper state, catching the exact types of configuration corruption that can prevent VM startup.

The script is highly suitable for administrators maintaining clustered or standalone PVE nodes with complex storage mappings, providing comprehensive health monitoring and preventive maintenance

capabilities.

End of Documentation