Troubleshooting

Diagnostic recipes for when a deployment stalls or a VM gets stuck: how to collect WHMCS module logs, cron output, Proxmox node logs and firewall configuration, plus the exact information to attach when opening a support ticket.

Log Collection

Proxmox KVM module **WHMCS**

Order now | Download | FAQ

When troubleshooting issues with the PUQ Proxmox KVM module, collecting the right logs is essential for diagnosing problems. Follow the steps below to gather all relevant information.

Step 0: Temporarily Disable WHMCS Cron

Before collecting logs, temporarily disable the WHMCS system cron job to prevent it from interfering with your manual debugging. This ensures you have full control over when tasks execute.

Comment out or disable the WHMCS cron entry in your crontab:

```
# crontab -e
# Temporarily comment out the WHMCS cron line:
# */5 * * * * php -q /var/www/whmcs/crons/cron.php
```

> **Important:** Remember to re-enable the cron job after you have finished troubleshooting.

Step 1: Run Cron Manually

Run the module cron manually to observe its output in real time. This captures all deploy pipeline activity, task processing, and any errors.

v3.0 (addon standalone cron):

```
php modules/addons/puq_proxmox_kvm/cron.php --force 2>&1 | tee /tmp/puq-cron.log
```

Run this command from the WHMCS root directory. The `--force` flag ensures the cron runs immediately regardless of scheduling. The output is both displayed on screen and saved to `/tmp/puq-cron.log`.

Legacy (v2.x and earlier) — run WHMCS cron directly:

In v2.x the provisioning tasks were chained onto the regular WHMCS cron, so you could capture the same output with the plain `cron.php`:

```
/usr/bin/php -q /WHMCS_DIR/crons/cron.php | tee /root/whmcs-cron-debug.log
```

Replace `/WHMCS_DIR` with the real path to your WHMCS installation. This command still works in v3.0 if you are running the cron in **WHMCS Hook mode** instead of standalone mode (see the Cron chapter).

What a successful run looks like

```
========================================
VM_id: 2001
Service_id: 4785
User_id: 1
VMSetDedicatedIp: The local status should be creation
VMDeleteDNSRecords: success
VMSetDNSRecords: success
VMClone: The local status should be set_ip|clone
VMSetCpuRam: The local status should be clone
VMSetSystemDiskSize: The local status should be set_cpu_ram
VMSetSystemDiskBandwidth: The local status should be set_system_disk_size
VMSetCreatedAdditionalDisk: The local status should be set_system_disk_bandwidth
VMSetAdditionalDiskSize: The local status should be set_created_additional_disk
VMSetAdditionalDiskBandwidth: The local status should be set_additional_disk_size
VMSetNetwork: The local status should be set_additional_disk_bandwidth
VMSetFirewall: The local status should be set_network
VMSetCloudinit: success
VMStart: success
Remote_status: running
Local_status: ready
```

```
ServiceSendEmailVMReady: OK
```

Each step either reports `success` or a reason why it was skipped (`The local status should be ...` — means the state machine is not ready for this step on this pass; it will resume on the next cron tick).

What an error looks like

```
VMSetCloudinit: HTTP/1.1 500 volume 'local:snippets/user-dnsfix.yaml' does not exist
```

When you see a line like that, the VM is stuck at the step printed **before** the error — fix the underlying cause (missing snippet, API timeout, IPSet conflict, etc.) and the next cron run will resume from that exact step.

Step 2: WHMCS Module Debug Log

The WHMCS module log records all API calls made by the module, including requests and responses.

1. In the WHMCS admin area, go to **Utilities > Module Debug Log**
2. If module logging is not already enabled, click **Enable Debug Logging**
3. Reproduce the issue (e.g., trigger a provisioning action)
4. Return to the Module Debug Log page and review the recorded API calls

Look for entries related to `puqProxmoxKVM` — these will show the exact API requests sent to Proxmox and the responses received.

> **Note:** Disable debug logging after troubleshooting, as it records all module API traffic and can grow rapidly.

Step 3: WHMCS Activity Log

The WHMCS activity log captures general system events, including provisioning actions, errors, and status changes.

1. Go to **Utilities > Activity Log** in the WHMCS admin area
2. Use the search/filter to narrow down entries by date or keyword

3. Look for entries related to the affected service or containing error messages

Step 4: Proxmox Node Logs

On the Proxmox server itself, review the system logs for the relevant services:

```
journalctl -u pvedaemon -u pveproxy -u pve-firewall --since "2 hours ago"
```

This shows logs from:

- **pvedaemon** — the Proxmox API daemon that processes VM operations
- **pveproxy** — the Proxmox API proxy that handles HTTPS requests
- **pve-firewall** — the Proxmox firewall service

Adjust the `--since` parameter to match the timeframe of the issue.

Step 5: Filter for a Specific VM

If you know the VMID of the affected virtual machine, filter the Proxmox logs for relevant entries:

```
grep -E "<VMID>|ipset|firewall|cloudinit|error|fail" /var/log/pve/tasks/*
```

Replace `<VMID>` with the actual VM ID number (e.g., `100`).

You can also check the Proxmox task log for the specific VM:

```
grep -r "<VMID>" /var/log/pve/tasks/
```

Step 6: Firewall Configuration

If the issue is related to networking or firewall rules, check the VM-specific firewall configuration:

```
cat /etc/pve/firewall/<VMID>.fw
```

This file contains the firewall rules and IP sets applied to the specific VM. If the file does not exist, the VM has no specific firewall rules configured.

Also check the cluster-level and node-level firewall configuration:

```
cat /etc/pve/firewall/cluster.fw
```

What to Send to Support

When contacting PUQ support, include the following information:

1. **Cron output** — the `/tmp/puq-cron.log` file from Step 1
2. **WHMCS Module Debug Log** — export or screenshot the relevant entries from Step 2
3. **WHMCS Activity Log** — relevant entries from Step 3
4. **Proxmox logs** — output from Step 4 and Step 5
5. **Firewall configuration** — output from Step 6 (if applicable)
6. **VM status** — the current status of the affected VM in both WHMCS and the Proxmox web UI
7. **Module version** — the version of the PUQ Proxmox KVM module you are running
8. **WHMCS version** and **PHP version**
9. **Proxmox VE version**

> **Tip:** When collecting logs, try to reproduce the issue as closely to the log collection time as possible. This ensures the relevant entries are captured and easy to identify.