Troubleshooting

Diagnostic recipes for when a deployment stalls or a VM gets stuck: how to collect WHMCS module logs, cron output, resolving web server and checkout timeouts ("Invalid Invoice ID"), Proxmox node logs and firewall configuration, plus the exact information to attach when opening a support ticket.

- Log Collection
- Troubleshooting Checkout Timeouts & "Invalid Invoice ID"

Log Collection

Proxmox KVM module **WHMCS**

Order now | Download | Community

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.

Troubleshooting Checkout Timeouts & "Invalid Invoice ID"

Proxmox KVM module **WHMCS**

Order now | Download | Community

Symptoms

During customer checkout (especially when ordering multiple virtual machines or when account credit is applied) or when an admin clicks **Accept Order**:

1. **HTTP 504 Gateway Timeout:** The browser waits for 30–60 seconds, after which a white screen or a `504 Gateway Timeout` is displayed.
2. **"Invalid Invoice ID":** The user is redirected to `https://your-domain/cart.php?a=complete`, which renders a blank page with the plain text error message:

   ```
   Invalid Invoice ID
   ```

3. **Order stuck in Pending with Paid Invoice:** In the WHMCS Admin area (`orders.php?action=view&id=...`), the order status is `Pending`, the Invoice is marked `Paid` / `Complete`, but all services remain in `Pending` status and were never provisioned.

Root Cause Analysis

What Happens During Checkout

When a client completes an order, WHMCS executes a chain of heavy operations sequentially inside the same HTTP POST request (`cart.php?a=checkout`):

1. **Order & Invoice Creation:** Creates the order record in `tblorders`, services in `tblhosting`, and invoice in `tblinvoices`.
2. **Payment Application & Hooks:** If the invoice is paid immediately (e.g. via account credit or instant payment gateway), WHMCS applies payment, triggers `InvoicePaid` hooks, and executes fraud/sanctions screening.
3. **PDF Generation & SMTP Delivery:** WHMCS generates a PDF invoice via TCPDF and connects synchronously to the mail server via SMTP to dispatch the "Customer Invoice" email (this typically takes **5–10 seconds**).
4. **Payment Confirmation Email:** WHMCS sends another email confirming payment (another **5–10 seconds**).
5. **Module Provisioning:** WHMCS loops through each order item and calls `CreateAccount`.
6. **Session Finalization:** WHMCS stores `$_SESSION['orderdetails']` containing `InvoiceID` and redirects the browser to `cart.php?a=complete`.

Why the Error Occurs

By default, many web servers (Nginx reverse proxies, cPanel, Plesk, HestiaCP) have a strict **30-second proxy timeout** (`proxy_read_timeout 30s` or `fastcgi_read_timeout 30s`).

If the combined duration of PDF generation, SMTP delivery, and payment processing reaches **30 seconds**, the web server forcefully drops the HTTP connection and returns an HTTP 504 Gateway Timeout.

Because the connection was aborted mid-flight:

- The PHP process is terminated by the server before reaching session finalization and module provisioning.
- When the client's browser reloads or lands on `cart.php?a=complete`, `$_SESSION['orderdetails']['InvoiceID']` is empty or null.
- WHMCS queries the database with an empty invoice ID, fails to locate the record, and halts with **`Invalid Invoice ID`**.
- The services were never invoked by the module and remain in `Pending`.

Resolution

1. Enable Smart Hybrid Bulk Provisioning

in Module Settings

In the WHMCS Admin area:

1. Navigate to **Addons > PUQ Proxmox KVM > Settings > General**.
2. Locate **Bulk Provisioning Queue Threshold** (`bulk_provision_threshold`).
3. Set the threshold to `2` (the default).

How this helps: When an order contains more than 2 VMs (i.e. 3 or more), the module bypasses heavy synchronous Proxmox API calls during checkout. Instead, it queues the VMs for background provisioning via cron in approximately **0.04 seconds**, marks the services `Active`, and defers the Welcome Email until the VM is actually running. This keeps the total provisioning time during checkout under 1 second.

---

2. Increase Web Server & PHP Timeouts (Recommended: 120s – 300s)

To accommodate PDF generation and external SMTP latency during multi-item checkouts, increase script and proxy timeouts to **at least 120 seconds** (recommended: **300 seconds**):

PHP Configuration (`php.ini` & `www.conf`)

In your active PHP configuration (e.g. `/etc/php/8.x/fpm/php.ini`):

```
max_execution_time = 120    ; Recommended: 120 to 300
max_input_time = 120        ; Recommended: 120 to 300
```

In your PHP-FPM pool configuration (e.g. `/etc/php/8.x/fpm/pool.d/www.conf`):

```
request_terminate_timeout = 300s
```

Nginx Configuration (Reverse Proxy with Apache)

If Nginx proxies requests to Apache:

```
# Add to your Nginx site configuration or /etc/nginx/conf.d/timeout.conf:
proxy_connect_timeout 300s;
proxy_send_timeout    300s;
proxy_read_timeout    300s;
```

Nginx Configuration (Direct FastCGI / PHP-FPM)

If Nginx connects directly to PHP-FPM:

```
# Add to your location ~ \.php$ block or /etc/nginx/conf.d/timeout.conf:
fastcgi_connect_timeout 300s;
fastcgi_send_timeout    300s;
fastcgi_read_timeout    300s;
```

Apache Web Server

In `/etc/apache2/apache2.conf` or `/etc/httpd/conf/httpd.conf`:

```
Timeout 300
```

After modifying the configuration, reload the web server and PHP-FPM:

```
systemctl reload nginx
systemctl reload php8.x-fpm
systemctl reload apache2
```

How to Fix an Existing Stuck Order

If an order was already interrupted and remains in `Pending` status:

1. Open the WHMCS Admin area and navigate to **Orders > List All Orders**.
2. Click on the affected Order ID (e.g. Order #1308).
3. Verify that the invoice is marked `Paid` / `Complete`.
4. At the bottom of the page, ensure **Run Module Create** and **Send Welcome Email** are checked.
5. Click **Accept Order**.

With **Smart Hybrid Provisioning** active, all services will be queued in less than a second, and the background cron (`Process Virtual Machines`) will deploy the virtual machines one by one.