

Troubleshooting

- [Apple MDM Troubleshooting](#)
- [Backup Procedures for FileWave Hosted Servers](#)
- [FileWave Error Codes](#)
- [FileWave Log File Locations](#)
- [How to Restart FileWave Components](#)
- [Resolving Network Issues with FileWave Server or Boosters on macOS when using Carbon Black EDR Extension](#)
- [What is Compatibility Mode?](#)
- [FileWave Automation Scripts](#)

Apple MDM Troubleshooting

This Knowledge base article will help you troubleshoot mdm with FileWave.

Before going deep into troubleshooting, make sure that you have got these steps correct:

1. Your FileWave server should have a fully qualified DNS name (this dns name is the one entered in the Admin Preferences->Mobile)
2. If for some reason you changed the Server DNS Name in Admin Preferences->Mobile, did you re-generate the certificate? If you did, then you have to trust the new certificate from the enrollment page (<https://dns:20443/ios>)
3. If the APN cert upload fails from Admin Preferences, make sure you followed the exact steps from step 1, as this can be caused of password-protected certificate
4. If all of the above are set and still have problems, you need to create an admin user account for debugging django:
 - a. go to the FileWave server and type this command: "sudo fwcontrol mdm addadminuser" and follow the instructions
5. Another important log file is "/usr/local/filewave/log/filewave_django.log"
6. Make sure that your FileWave Admin displays "iOS/MDM Service OK" in the left lower corner in order to be able to manage your devices.

The following are some of the problems encountered before:

Enrolment Error (FileWave MDM Configuration is invalid):

The profile "Filewave MDM Configuration" is invalid. The MDM payload "Mobile Device Management" contains an invalid topic

This is usually solved by re-generating the APN certificates because you have not generated them correctly.

CONNECTION PROBLEMS**.**

There are cases where ios devices fail to enroll and you get an error similar to this from sentry:

```
error
(61, 'Connection refused')
Request Method: PUT
Request URL: https://sscfilwave.co.sbm:20443/ios/mdm_checkin
Exception Type: error
Exception Value:
(61, 'Connection refused')
Exception Location: /usr/local/filewave/python/lib/python2.7/socket.py in meth, line 222
```

This error is associated with a port "2195" being closed, you can verify by :

```
telnet gateway.push.apple.com 2195
Trying 17.172.239.89...
telnet: connect to address 17.172.239.89: Connection refused
```

the issue will be solved if the IT Admin opens port 2195 for FileWave.

5223 : IOS to apn server port:

port 5223 should be open for IOS clients to reach out to the APN server and receive push notifications.

For a list of all ports used, check this man-

Backup Procedures for FileWave Hosted Servers

What

This article details the backup procedures and policies for FileWave Hosted Servers. Understanding how and what data is backed up is essential for effectively managing and safeguarding your organization's devices and information.

When/Why

Backups are automatically performed daily for all FileWave Hosted Servers. This routine is crucial for disaster recovery, maintaining data integrity, and ensuring minimal downtime in unexpected data loss situations. The retention period for these backups is 30 days, ensuring a sufficient window for recovery if needed.

How

Backups are executed daily and stored securely in highly available AWS S3 buckets. The following paths are included in the backups, ensuring comprehensive coverage of both configuration and operational data:

- /usr/local/filewave/fwserver/DB
- /private/var/log
- /usr/local/filewave/fwserver/Data Folder
- /usr/local/filewave/instrumentation_data
- /usr/local/filewave/apache/conf
- /usr/local/filewave/apache/logs
- /usr/local/filewave/apache/passwd
- /usr/local/filewave/django/filewave
- /usr/local/filewave/conf
- /usr/local/filewave/certs
- /usr/local/filewave/fwcd
- /usr/local/filewave/ipa
- /usr/local/filewave/log
- /usr/local/filewave/media
- /usr/local/filewave/tmp
- /var/log
- /usr/local/filewave/scripts
- /usr/local/etc
- /tmp
- /usr/local/filewave/nats
- /install/docker-entrypoint.sh
- /usr/local/filewave/tls
- /etc/filewave_init
- /etc/filewave_setup

It is important to note that while FileWave ensures the security and availability of backups, direct access to these backups by customers is not provided. Going to a backup would generally be based on scenarios such as database or file corruption or the loss of an AWS datacenter due to a disaster.

Related Links

- [FileWave Server Backup and Restore](#) for On-Premise

Digging Deeper

Backups are a critical aspect of data management and recovery strategies. They ensure that in the event of data loss, corruption, or disaster, operations can be restored with minimal impact. FileWave's Hosted Server backups are designed to provide a robust and secure safety net for your organization's device management infrastructure.

FileWave Error Codes

Server

Error	Context	Explanation	Solution
-8	During Database Verification	There are some orphaned objects in the database	The first thing to do is to run a DB compact. You can run it from Xserver monitor which is in /Applications/FileWave. This should solve the issue. If the compact is not fixing it, then we must be missing a certain type of cleanup in that operation. Generally, this doesn't pose a problem. If you'd like, you can stop the server and zip up the /fwxserver/Data Folder/*datand*idx files and post them to our ftp site.
-1	While doing a Model Update you see a blank window	error creating Fileset File: XXXXXX, folderID: YYYYYYY not found, database damaged, call FileWave Tech Support 889 0xb0513000 FATAL Error: -1 when updating filesets during model update	This issue is fixed in FileWave version 4.1.1. If you are hitting this issue please upgrade Filewave Admin to 4.1.1.
14	Error 14 on file, found process: fwserver/XXXXX exiting due to database error: 14 (Only Applicable to Server 3.7.4)	This is a soft database failure caused by a hard restart of the FileWave Server, it doesn't actually reflect a serious issue, but will cause the server to stop.	Upgrade to 3.7.5 to 4.0.X
<code>
fwxserver_conflicts_with_fwmdm-server-10.1.1-1.0.x86_64
</code>	upgrading to FileWave 11.x from a previous version	This is normally caused by upgrading from a system that originally installed filewave with two packages: fw-mdm-server fwxserver	As of version 11, FileWave installs both servers with just the single fwxserver installer. To fix this simply remove the mdm component before updating the server. This will not remove any of your MDM data <code>
sudo yum remove -y fw-mdm-server
</code>

Client / Admin

Error	Context	Explanation	Solution
-150		the file size downloaded to disk does not match the file size stored in the database of the FileWave Server	Delete this file from the Fileset and add a fresh copy from the Admin's hard disk
-125	Client downloading fileset	Booster does not have the file to serve to the client yet and so the client will try again later	Please wait
-13	fwgui is not running	On the client fwgui process is not running	Restart the filewave client from terminal : macOS / Linux <code>
sudo fwcontrol client restart
</code>
-3	During Admin File Upload	On slow networks an upload may timeout.	
-1	Not in inventory	That comes from a client attempting to activate a fileset before it has downloaded it. After the model update, it adds the activation action back into the queue	Please wait for sometime as the client is still downloading the fileset and once it has finished downloading it will activate
2	reading file from disk	This error is due to a wrong offset request	Upgrade to 3.7.5 or 4.0.4 will solve this issue
15		This could happen if there is no or very less disk space left on the booster that the client is downloading the filesets from	Please check if the booster has enough free disk space. If the disk space is enough and still you are seeing this error contact support help.filewave.com
32	while trying to send file data XXXXXD (Where XXXXX is the file ID)	Error 32 means broken pipe in the network. Generally this error should resolve by itself if everything in the network is fine. Troubleshooting : check to ping from the booster/server to client and vice versa and check if the problem doesn't exist in the network	1. If you see this error for long time try to remove the association of this fileset with the client and then associate again. This should solve the problem. 2. Update to latest Filewave 3.7.X or 4.0.X
Failed CRC Validation		A CRC check is a form of a checksum which is used to make sure data in	Please send the client log file from the client exhibiting this issue to

		files is the same on the client as on the server. The error "failed CRC validation" means that files on the client for whatever reason are being altered compared to what is on the server.	support help.filewave.com
Kiosk Errors		See: VPP Kiosk Errors	

Booster

Error	Context	Explanation	Solution
Failed CRC Validation		A CRC check is a form of a checksum which is used to make sure data in files is the same on the client as on the server. The error "failed CRC validation" means that files on the client for whatever reason are being altered compared to what is on the server	Please send the booster log file from the booster exhibiting this issue to help.filewave.com

FileWave Log File Locations

The following lists the locations of log files, as well as some additional files used by FileWave across the FileWave family of products

FileWave Admin

FileWave Admin Logs

Details	File	Location
FileWave Admin Log Logs all FileWave Admin Connection Activity	FileWaveAdmin.log, FileWaveAdmin.log.*	macOS <code>~/Library/Application Support/FileWave/FileWaveAdmin.log</code> Windows <code>C:\ProgramData\FileWave\FileWaveAdmin.log</code>
Client Logs Retrieved Client Logs	ClientLog_\$IP_\$Port_\$date.log	macOS <code>~/Library/Application Support/FileWave/Client Logs/</code> Windows <code>C:\ProgramData\FileWave\Client Logs\</code>
Server Logs Retrieved Server Logs FileWave Admin > Server > Get Logfile	fwxserver_\$timestamp.log	macOS <code>~/Library/Application Support/FileWave/Server Logs/</code> Windows <code>C:\ProgramData\FileWave\Server Logs\</code>

FileWave Admin Files

Details	File	Location
FileWave Admin Settings Settings for the local FileWave Admin App	macOS <ul style="list-style-type: none">com.filewave.FileWaveAdmin.plistcom.filewave.admin.plist Windows <ul style="list-style-type: none">Registry	macOS <code>~/Library/Preferences/</code> Windows <code>HKCU\Software\FileWave\FileWave Admin</code>
Exported Views Views saved from FileWave Admin: * Views > Export Current View	Filesets Export (\$date).txt	macOS <code>~/Library/Application Support/FileWave/Exports</code> Windows <code>C:\ProgramData\FileWave\Exports</code>

FileWave Booster

Booster Logs

Details	File	Location
Booster Log Global Booster activity	fwbooster.log	macOS/Linux <code>/private/var/log/fwbooster.log</code> Windows <code>C:\ProgramData\FileWave\FWBooster\fwbooster.log</code>
NATS NATS Booster Logs	macOS/Linux <ul style="list-style-type: none">nats-booster.err.lognats-booster.out.log Windows <ul style="list-style-type: none">nats-booster.log	macOS/Linux <code>/private/var/log/</code> Windows <code>C:\ProgramData\FileWave\FWBooster\NATS\nats-booster.log</code>
Discovery Log Only exists when discovery configured and run	macOS/Linux <ul style="list-style-type: none">fwdiscovery.log	macOS/Linux <code>/private/var/log/fwdiscovery.log</code>

FileWave Client

FileWave Client Logs

Details	File	Location
Client Logs Global Client activity	fwcld.log	macOS <code>/var/log/fwclld.log</code> Windows <code>C:\ProgramData\FileWave\FWClient\fwcld.log</code>
Kiosk Logs Kiosk application activity	FWGUI.log	macOS <code>~/Library/Application Support/FileWave/FWGUI.log</code> Windows <code>C:\ProgramData\FileWave\FWGUI.log</code>
Fileset Script Logs Logs generated by Fileset scripts	macOS \$Fileset_ID/\$script_name_from_fileset.log Windows \$Fileset_ID/\$script_name_from_fileset.log	macOS <code>/var/log/fwclld/</code> Windows <code>C:\ProgramData\FileWave\log\fwclld\</code>
Custom Field Logs Logs generated by Custom Fields	custom_field_script.\$script_type.log e.g. <ul style="list-style-type: none">custom_field_script.ps1.logcustom_field_script.sh.log	macOS <code>/var/log/fwclld/1/</code> Windows <code>C:\ProgramData\FileWave\log\fwclld\1\</code>
Fileset Blocker Script Logs Logs generated by Blocker Scripts	blocker_script.\$script_type.log e.g. <ul style="list-style-type: none">blocker_script.py.logblocker_script.bat.log	macOS <code>/var/log/fwclld/1/</code> Windows <code>C:\ProgramData\FileWave\log\fwclld\1\</code>
Installer (PKG / MSI) Logs Logs generated from PKG/MSI Filesets	\$Fileset_ID.log	macOS <code>/usr/local/etc/FileWaveInstallerLogfiles/</code> Windows <code>C:\ProgramData\FileWave\FileWaveInstallerLogfiles\</code>

FileWave Client Files

Details	File	Location
FileWave Client Settings Settings for the FileWave Client	macOS <ul style="list-style-type: none">fwcld.plist Windows <ul style="list-style-type: none">Registry	macOS <code>/usr/local/etc/</code> Windows <code>HKLM\SOFTWARE\Wow6432Node\Filewave\WinClient</code>
FileWave Client Preferences Preference file containing unique client details	macOS <ul style="list-style-type: none">com.filewave.Client.plist Windows <ul style="list-style-type: none">client.ini	macOS <code>/Library/Preferences/</code> Windows <code>C:\ProgramData\FileWave\</code>
FileWave Client Certificate Unique certificate & key per client	<ul style="list-style-type: none">client.crtclient.key	macOS <code>/var/FileWave/</code> Windows <code>C:\ProgramData\FileWave\FWClient\</code>
Trust Store Store for self-signed certificates	*.crt	macOS <code>/private/var/FileWave/trust_store</code> Windows <code>C:\ProgramData\FileWave\FWClient\trust_store</code>



Cells highlighted in blue indicate files that are unique per client. These files should not be included when copying or migrating clients from one machine to another. To de-personalise a device, without removing the FileWave Client, some files would require editing, whilst others would need to be removed. If it was felt this was a requirement, consider contacting support to assist with this process.

FileWave Imaging Server (IVS)

IVS Logs

Details	File	Location
Django Imaging Server Logs Django logs for requests regarding Serial numbers, names etc. made by netbooted clients	filewave_imaging_server*.log	/imaging/logs/
Windows Image Upload Logs Captured Windows image uploads	fwadmin.log	/imaging/logs/fwadmin.log
Windows Image Upload Logs Captured Windows image uploads	fwadmin-dlog.log	/var/log/fwadmin-dlog.log
Messages Logs Netboot/PXE Queries & Responses, TFTP transfers, NFS Mounts	dnsmasq Log	CentoS /var/log/messages Debian /var/log/syslog
Apache Imaging Server Logs Apache logs for requests regarding Serial numbers, names etc. made by netbooted clients	netboot_*.log	/imaging/logs/
Client Imaging Logs Client logs - indicating progress of imaging operation of netbooted clients	\$Serial/\$Mac-\$Date	/imaging/logs/
FileWave Client Log IVS FileWave Client Log	fwcld.log	/var/log/fwcld.log

FileWave Server

FileWave Server Logs

Details	File	Location
Apache Logs Server Apache logs	<ul style="list-style-type: none">access_log, access_log.*error_log, error_log.*forensic_loghttpd.pid	/usr/local/filewave/apache/logs/
Apache Exporter Logs Server Apache Exporter Logs	<ul style="list-style-type: none">apache_exporter.out.logapache_exporter.err.log	/usr/local/filewave/log/
Alert Manager Logs Server Alert Manager logs	<ul style="list-style-type: none">alertmanager.out.logalertmanager.err.log	/usr/local/filewave/log/
FileWave Admin Audit Logs Audit logs from FileWave Admin	audit.log	/usr/local/filewave/log/audit.log
FileWave Admin Audit Logs Audit logs from FileWave Admin	fwaaudit-[date].txt	/private/var/log/FWAdmin Audit/
FileWave Dotenv file Environment variable like configs across services.	*.env	/usr/local/etc/filewave/.env
Django Logs Server Django logs	<ul style="list-style-type: none">filewave_django.log,filewave_django.log.*filewave_django_vpp.logfilewave_django_classroom.log	/usr/local/filewave/log/
LDAP Logs Logs from LDAP	fwldap.log, fwldap.log.*	/private/var/log/
Software Update Logs Software Update logs	fwsu.log	/private/var/log/fwsu.log
FWX Process Logs	<ul style="list-style-type: none">fwxadmin.log	/private/var/log/

Various fwX process logs	<ul style="list-style-type: none"> fwxother.log fwxserver.log 	
Model Update Service Log	model_update_service.log	/usr/local/filewave/log/
Migration Logs Server migration logs	fwxserver-migration-*	/var/log/fwxserver-migration-*
Grafana Logs	<ul style="list-style-type: none"> grafana.log grafana.out.log 	/usr/local/filewave/log/
Installer Logs Linux installer logs	install.log	/private/var/log/install.log
mtail Logs Server mtail logs	<ul style="list-style-type: none"> mtail.out.log mtail.err.log 	/usr/local/filewave/log/
NATS Logs NATS logs	<ul style="list-style-type: none"> nats-server.out.log nats-server.err.log nats-server-jwt.out.log nats-server-jwt.err.log 	/usr/local/filewave/log/
Web Admin Logs	<ul style="list-style-type: none"> node_exporter.out.log node_exporter.err.log task_executor.log 	/usr/local/filewave/log/
Postgres Exporter Logs	<ul style="list-style-type: none"> postgres_exporter.out.log postgres_exporter.err.log 	/usr/local/filewave/log/
Postgres Database Logs	postgresql-\$day.log	/usr/local/filewave/fwxserver/DB/pg_data/pg_log/*.log
Prometheus Logs	<ul style="list-style-type: none"> prometheus_pushprox.out.log prometheus_pushprox.err.log prometheus.out.log prometheus.err.log redis_exporter.out.log redis_exporter.err.log redis.out.log redis.err.log redis.log 	/usr/local/filewave/log/
FileWave Server Logs	request_errors.log	/usr/local/filewave/log/
SQL Logs	sql.log	/usr/local/filewave/log/
Update Controller Logs Removed in FileWave 14.10	<ul style="list-style-type: none"> update_controller_access.log update_controller.log 	/usr/local/filewave/log/
Client Monitor	client-monitor.log	/usr/local/filewave/log/
FileWave Log Messages	task_executor.log	/usr/local/filewave/log/
Scheduler Log Messages	huey.log	/usr/local/filewave/log/

Additional Logging

All of the above will default to standard log level. There are 3 levels of logging available:

- 10 – Standard Log Level
- 99 – Debug Log Level
- 101 – Trace Log Level

The level of logging may be set as per our guide:

[How to set FileWave Server components to debug mode](#)

How to Restart FileWave Components

There may be times where you will need to restart all components within the FileWave server, or just a single component (postgres or apache). From your macOS or Linux server you can type "fwcontrol", which should give examples of fwcontrol usage.

macOS or Linux Server

You need to prefix commands with sudo to run them with elevated privileges.

At a command prompt:

```
sudo fwcontrol server stop
sudo fwcontrol server start
```

You can also accomplish the same end goal by performing a single command:

```
sudo fwcontrol server restart
```

It is a matter of preference, but some admins will prefer to execute a stop, then a manual start so that they can see all processes are indeed stopped.

Subcomponents can be individually stopped as follows:

```
sudo fwcontrol apache start|stop|restart

sudo fwcontrol postgres start|stop|restart

sudo fwcontrol scheduler start|stop|restart


sudo fwcontrol client start|stop|restart

sudo fwcontrol booster start|stop|restart
```

Troubleshooting

If you find that the fwcontrol control command is not found, you re-create the alias by inputting this command and then try the fwcontrol commands again:

```
alias fwcontrol='/usr/local/bin/fwcontrol'
```

Resolving Network Issues with FileWave Server or Boosters on macOS when using Carbon Black EDR Extension

What

FileWave has observed network issues when the Carbon Black EDR (Endpoint Detection and Response) extension is installed on a FileWave server or booster running on macOS. The issues can manifest as Boosters stopping to answer or respond, leading to disruption in device management workflows.

When/Why

The issue occurs when there is a high volume of network traffic and the Carbon Black EDR extension is inserted into the network stack. The extension's presence in the network stack seems to cause performance issues, which can result in network connectivity and communication problems.

How

If you experience network issues with FileWave when the Carbon Black EDR extension is installed, you can resolve the problem by removing the extension from the FileWave server or booster. This solution has been proven to resolve the issue in multiple cases. On a macOS system, you can use the following command in Terminal.app to list all kernel extensions:

```
systemextensionsctl list
```

The output will appear like this:

```
--- com.apple.system_extension.endpoint_security
enabled   active    teamID    bundleID (version)  name      [state]
*         *        7AGZNQ2S2T  com.vmware.carbonblack.cloud.se-agent.extension (3.7.2fc81/3.7.2fc81)
com.vmware.carbonblack.cloud.se-agent.extension  [activated enabled]
```

You should check the output of this command to determine if the Carbon Black EDR extension is present on your system. If you have concerns about the performance of the Carbon Black EDR extension in high-volume network traffic environments, it may be worth contacting Carbon Black's support team to discuss the issue further.

Related Content

- [General Troubleshooting and Errors](#)
- [FileWave Log File Locations](#)
- [Booster Installation](#)

Digging Deeper

Kernel extensions (KEXTs) are software modules that can be inserted into the macOS kernel to extend its functionality. They can be used to add new features, support new hardware, or modify the behavior of existing drivers. KEXTs run in kernel mode, which means they have the highest level of privilege and can access system resources directly.

However, KEXTs can also introduce stability and performance issues. Since they run in kernel mode, they can crash the system or cause conflicts with other KEXTs. In addition, they can potentially introduce security vulnerabilities if they're not properly designed or implemented.

The Carbon Black EDR extension is an example of a kernel extension that inserts itself into the macOS network stack. By doing so, it's able to monitor network traffic and detect security threats. However, in high-volume network traffic environments, the extension can cause performance issues, which can lead to disruptions in FileWave's device management workflows.

To manage kernel extensions on macOS, Apple provides the `systemextensionsctl` command. This command allows you to list, enable, disable, and uninstall extensions. If you're experiencing issues with a KEXT, you can use this command to disable or uninstall it to see if that resolves the issue.

In general, it's important to use kernel extensions with caution and only install those from trusted sources. If you're unsure whether a particular KEXT is necessary or safe to use, you should consult with the vendor or seek advice from a subject matter expert.

What is Compatibility Mode?

FileWave 13.1 introduced new security options and a mode to allow older clients to connect.

Compatibility Mode was removed in FileWave 15.4.0 in favor of only using secure connections.

Problem

I don't know what compatibility mode is and what enable and disable do for me.

The screenshot shows the FileWave administration interface with the 'General' tab selected. The 'SSL Certificate Management' section displays the CA chain as 'DST Root CA X3 -> Let's Encrypt Authority X3', the common name, and the expiration date. Below this, the 'Server Settings' section has a red box highlighting the 'Compatibility mode for versions older than FileWave 13.1' checkbox, which is checked. A note below the checkbox states: 'This option should be enabled if any component is not up-to-date. FileWave 13.1 uses secured communication channels for compatible clients, boosters, and imaging servers running version 13.1 or newer.' The 'Local Settings' section includes options for auto logout, confirmation dialogs, and other system preferences.

Environment

FileWave 13.1 introduces a new method of certificate-based security for communication between components (client, booster, server and IVS). Only 13.1 and greater components are able to generate and properly use certificates to communicate with other components using the new method. Therefore, if your server is running 13.1 but you have components that are older than 13.1 they can not generate the needed certificates to have the highest level of security, and will not be able to communicate together.

Resolution

Compatibility Mode Enabled

The server allows older clients, boosters, and IVS to communicate with the server with or without valid certificates

Compatibility Mode Disabled

The server will not allow any client, booster, or IVS to communicate with the server unless it has a valid and unique certificate. Boosters and clients are also checking peer certificates and will only communicate if the peer certificate is valid.

Additional Information

When you disable compatibility mode (uncheck the box in preferences) you will receive a warning of clients, boosters, and imaging appliances (AKA IVS), that may potentially be disconnected by you enabling this mode. If you get this warning, it is recommended that you cancel, and resolve the issue before compatibility mode is disabled.



Warning: clients, boosters, and imaging servers must be on version 13.1 or greater to support full security mode.

**Do you really want to enable full security mode?
The following components will no longer be able to communicate with the FileWave Server:**

1 client(s)

1 booster(s)

0 imaging server(s)

Out-dated components will no longer connect to the FileWave Server. They must be updated and re-enrolled.

Note: This change will automatically logout other FileWave Admin sessions.

Cancel

OK

Related Content

- For approving devices see: [Enrolling Computer Clients](#)
- For approving boosters see: [Booster installation](#)

FileWave Automation Scripts

This article contains the scripts used by the Downloads page. The scripts are here for documentation purposes.

Scripts

FileWave Server Upgrade

[fwxserver_upgrade.sh](#) - This script is used by the Download page for FileWave Server upgrades on Debian. Some details;

```
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v 15.5.0 -r 1 -p -y
```

▼ fwxserver_upgrade.sh

```
#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/411 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/409 | sudo bash -s -- -v <version> -r <revision> [-b for beta | -p for prod] [-y for auto-yes]"
    echo "Options:"
    echo "  -v, --version      Specify the version of FileWave Server to install (e.g., 15.5.0)"
    echo "  -r, --revision     Specify the revision number (e.g., 1)"
    echo "  -b, --beta         Use beta server for downloads (default is production)"
    echo "  -p, --prod         Use production server for downloads"
    echo "  -y, --yes          Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {
    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_server_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
```

```

        die "Cannot write to log file $LOG_FILE. Please check permissions."
    fi
    exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
    echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave Server packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi
if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type
if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "curl" "zip" "gdebi"; do
    if ! command -v $dep &> /dev/null; then
        echo "$dep not found. Installing $dep..."
        apt-get install -y $dep || die "Failed to install $dep."
    fi
done

if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
    echo "net-tools not found. Installing net-tools..."
    apt-get install -y net-tools || die "Failed to install net-tools."

```

```

fi

# Check the current version of FileWave Server
INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' fwxserver 2>/dev/null || true)
# If no FileWave Server package is installed, set the version to 1.0.0 for comparison purposes
if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
    echo "No FileWave Server packages are installed. Proceeding with initial installation of version $VERSION..."
    INSTALLED_VERSION="1.0.0"
fi

install_packages() {
    echo "Downloading and installing FileWave Server package..."
    PACKAGE="fwxserver_${VERSION}_amd64.deb"
    DOWNLOAD_DIR="/tmp/filewave_install_$$"
    mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
    trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
    echo "Downloading $PACKAGE..."
    for i in {1..3}; do
        curl -fSL "$BASE_URL/$VERSION/$PACKAGE" -o "$DOWNLOAD_DIR/$PACKAGE" && break
        echo "Retrying download ($i/3)..."
        sleep 5
    done
    if [[ ! -f "$DOWNLOAD_DIR/$PACKAGE" ]]; then
        die "Failed to download $PACKAGE after multiple attempts."
    fi
    echo "Installing $PACKAGE..."
    gdebi -n "$DOWNLOAD_DIR/$PACKAGE" || die "Failed to install $PACKAGE"

    # Clean up the download directory
    if [[ -d "$DOWNLOAD_DIR" ]]; then
        rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
    fi
}

# This is the meat of the script.
# The actions to take based on version.
if dpkg --compare-versions "$INSTALLED_VERSION" lt "$VERSION"; then
    echo "Upgrading to FileWave Server version $VERSION..."
    install_packages
elif dpkg --compare-versions "$INSTALLED_VERSION" eq "$VERSION"; then
    echo "FileWave Server is already at version $VERSION, no further installation required."
else
    echo "FileWave Server is newer than $VERSION and downgrade is not possible with this script."
fi

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi

# 4. OS Upgrade
echo "Warning: This script will upgrade the entire OS. This is required for security."
if [[ "$auto_yes" == "true" ]]; then
    upgrade_confirm="yes"
else
    read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
fi
if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Skipping OS upgrade."
else
    echo "Upgrading the system..."
    apt-get update -y || die "Failed to update package list."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System upgrade failed."
fi

# 5. Reboot the system to complete the installation
echo "The system will reboot in 15 seconds to complete the installation."
if [[ "$auto_yes" == "true" ]]; then

```

```

        reboot_confirm="yes"
    else
        read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
    fi
    if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
        echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
        sleep 15
        reboot
    else
        echo "Skipping reboot. Please remember to reboot manually to apply all changes."
    fi

# The end

```

FileWave Booster Upgrade

[fwbooster_upgrade.sh](#) - This script is used by the Download page fro FileWave Booster upgrades on Debian. Some details;

```

# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

```

▼ fwbooster_upgrade.sh

```

#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v <version> -r <revision> [-b for
beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/412 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/409 | sudo bash -s -- -v <version> -r
<revision> [-b for beta | -p for prod] [-y for auto-yes]"
    echo "\nOptions:"
    echo "  -v, --version      Specify the version of FileWave Booster to install (e.g., 15.5.0)"
    echo "  -r, --revision     Specify the revision number (e.g., 1)"
    echo "  -b, --beta         Use beta server for downloads (default is production)"
    echo "  -p, --prod         Use production server for downloads"
    echo "  -y, --yes          Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {

```

```

    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_booster_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
    die "Cannot write to log file $LOG_FILE. Please check permissions."
fi
exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave Booster packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi
if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type
if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "curl" "zip" "gdebi"; do

```

```

    if ! command -v $dep &> /dev/null; then
        echo "$dep not found. Installing $dep..."
        apt-get install -y $dep || die "Failed to install $dep."
    fi
done

if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
    echo "net-tools not found. Installing net-tools..."
    apt-get install -y net-tools || die "Failed to install net-tools."
fi

# Check the current version of FileWave Booster
INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' fwbooster 2>/dev/null || true)
# If no FileWave Booster package is installed, set the version to 1.0.0 for comparison purposes
if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
    echo "No FileWave Booster packages are installed. Proceeding with initial installation of version $VERSION..."
    INSTALLED_VERSION="1.0.0"
fi

install_packages() {
    echo "Downloading and installing FileWave Booster package..."
    PACKAGE="fwbooster_${VERSION}_amd64.deb"
    DOWNLOAD_DIR="/tmp/filewave_install_$$"
    mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
    trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
    echo "Downloading $PACKAGE..."
    for i in {1..3}; do
        curl -fSL "$BASE_URL/$VERSION/$PACKAGE" -o "$DOWNLOAD_DIR/$PACKAGE" && break
        echo "Retrying download ($i/3)..."
        sleep 5
    done
    if [[ ! -f "$DOWNLOAD_DIR/$PACKAGE" ]]; then
        die "Failed to download $PACKAGE after multiple attempts."
    fi
    echo "Installing $PACKAGE..."
    gdebi -n "$DOWNLOAD_DIR/$PACKAGE" || die "Failed to install $PACKAGE"

    # Clean up the download directory
    if [[ -d "$DOWNLOAD_DIR" ]]; then
        rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
    fi
}

# This is the meat of the script.
# The actions to take based on version.
if dpkg --compare-versions "$INSTALLED_VERSION" lt "$VERSION"; then
    echo "Upgrading to FileWave Booster version $VERSION..."
    install_packages
elif dpkg --compare-versions "$INSTALLED_VERSION" eq "$VERSION"; then
    echo "FileWave Booster is already at version $VERSION, no further installation required."
else
    echo "FileWave Booster is newer than $VERSION and downgrade is not possible with this script."
fi

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi

# 4. OS Upgrade
echo "Warning: This script will upgrade the entire OS. This is required for security."
if [[ "$auto_yes" == "true" ]]; then
    upgrade_confirm="yes"
else
    read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
fi
if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Skipping OS upgrade."
else

```

```

        echo "Upgrading the system..."
        apt-get update -y || die "Failed to update package list."
        DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System
upgrade failed."
    fi

    # 5. Reboot the system to complete the installation
    echo "The system will reboot in 15 seconds to complete the installation."
    if [[ "$auto_yes" == "true" ]]; then
        reboot_confirm="yes"
    else
        read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
    fi
    if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
        echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
        sleep 15
        reboot
    else
        echo "Skipping reboot. Please remember to reboot manually to apply all changes."
    fi
fi

# The end

```

FileWave IVS Upgrade

[ivs_upgrade.sh](#) - This script is used by the Download page for IVS upgrades on Debian. Some details;

To run this script, use the following 1-liner:
curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v <version> -r <revision> [-b for beta] -y
Example for version 15.5.0 with revision 1 in production:
curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

▼ ivs_upgrade.sh

```

#!/bin/bash
# Documentation
# To run this script, use the following 1-liner:
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v <version> -r <revision> [-b for
beta] -y
# Example for version 15.5.0 with revision 1 in production:
# curl -fsSL https://kb.filewave.com/attachments/408 | sudo bash -s -- -v 15.5.0 -r 1 -p -y

# Ensure script is run with sudo/root privileges
if [ "$EUID" -ne 0 ]; then
    die "This script must be run with sudo or as root."
fi

# Ensure script is running on a Debian-based system
if ! grep -iq "debian" /etc/os-release; then
    die "This script must be run on a Debian-based system."
fi

# Set default values for version, revision, and server type
VERSION="15.5.0"
REVISION="1"
SERVER_TYPE="prod"
BASE_URL="https://fwdl.filewave.com"
auto_yes=false
setup_cron_job=false
install_packages=false

if [ "$#" -eq 0 ]; then
    echo "Usage: curl -fsSL https://kb.filewave.com/attachments/401 | sudo bash -s -- -v <version> -r
<revision> [-b for beta] [-p for prod] [-y for auto-yes]"

```



```

    echo "
Options:"
    echo "  -v, --version      Specify the version of FileWave IVS to install (e.g., 15.5.0)"
    echo "  -r, --revision    Specify the revision number (e.g., 1)"
    echo "  -b, --beta        Use beta server for downloads (default is production)"
    echo "  -p, --prod        Use production server for downloads"
    echo "  -y, --yes         Automatically answer 'yes' to all prompts"
    exit 1
fi

# Function to handle errors and display messages
die() {
    echo "[ERROR] $1" >&2
    exit 1
}

# Log all actions to syslog for audit purposes
LOG_FILE="/var/log/filewave_ivs_update.log"
touch "$LOG_FILE" || die "Failed to create log file $LOG_FILE. Check permissions."
chmod 644 "$LOG_FILE" || die "Failed to set permissions on log file $LOG_FILE."
if [[ ! -w "$LOG_FILE" ]]; then
    die "Cannot write to log file $LOG_FILE. Please check permissions."
fi

exec >>(stdbuf -oL tee -a "$LOG_FILE") 2>&1
echo "Logging to $LOG_FILE"

# Parse input arguments
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -v|--version)
            VERSION="$2"
            shift 2
            ;;
        -r|--revision)
            REVISION="$2"
            shift 2
            ;;
        -b|--beta)
            SERVER_TYPE="beta"
            shift
            ;;
        -p|--prod)
            SERVER_TYPE="prod"
            shift
            ;;
        -y|--yes)
            auto_yes=true
            shift
            ;;
        *)
            echo "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Confirm actions with the user
echo "This script will update your OS and install/upgrade the FileWave IVS packages."
echo "Version: $VERSION, Revision: $REVISION, Server Type: $SERVER_TYPE"
if [[ "$auto_yes" == "true" ]]; then
    confirm="yes"
else
    read -p "Do you wish to proceed? (yes/no): " confirm < /dev/tty
fi

if [[ ! "$confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Aborting installation as requested."
    exit 0
fi

# Set the appropriate URL based on server type

```

```

if [ "$SERVER_TYPE" == "beta" ]; then
    BASE_URL="https://fwbetas.filewave.com"
fi

# 1. Install Essential Tools
echo "Installing essential tools..."
apt-get clean || die "Failed to clean apt cache."
apt-get update -y || die "Failed to update package list."
apt-get --fix-broken install -y || die "Failed to fix broken installs from apt."
apt-get autoremove -y || die "Failed to autoremove from apt."

for dep in "python3" "curl" "zip" "gdebi"; do
    if ! command -v $dep &> /dev/null; then
        echo "$dep not found. Installing $dep..."
        apt-get install -y $dep || die "Failed to install $dep."
    fi
done

if ! dpkg-query -W -f='${Status}' net-tools 2>/dev/null | grep -q "install ok installed"; then
    echo "net-tools not found. Installing net-tools..."
    apt-get install -y net-tools || die "Failed to install net-tools."
fi

# Check current version
INSTALLED_VERSION=$(dpkg-query -W -f='${Version}' ivs-kernel 2>/dev/null || true)
if [[ -z "$INSTALLED_VERSION" || "$INSTALLED_VERSION" == "none" ]]; then
    die "No FileWave IVS packages are installed. Please use the FileWave IVS Appliance image to set up IVS initially."
fi

# Extract installed main version and revision
INSTALLED_MAIN_VERSION=$(echo "$INSTALLED_VERSION" | cut -d'-' -f1)
INSTALLED_REVISION=$(echo "$INSTALLED_VERSION" | cut -d'-' -f2)

if [[ -z "$INSTALLED_REVISION" ]]; then
    INSTALLED_REVISION=0
fi

# Prevent installing 15.5.0
if dpkg --compare-versions "$VERSION" eq "15.5.0"; then
    echo "Installing FileWave IVS 15.5.0 is not supported. Please install version 15.5.1 or newer."
    exit 1
fi

# Prevent downgrades or re-installing the same version
if dpkg --compare-versions "$INSTALLED_MAIN_VERSION" gt "$VERSION" || \
    ( dpkg --compare-versions "$INSTALLED_MAIN_VERSION" eq "$VERSION" && [[ "$INSTALLED_REVISION" -ge
"$REVISION" ] ] ); then
    echo "Installed version ($INSTALLED_VERSION) is newer or equal to the requested version ($VERSION-
$REVISION)."
    echo "Aborting to prevent downgrade or re-installation of the same version."
    exit 1
fi

install_packages() {
    echo "Downloading and installing FileWave IVS packages..."
    PACKAGE_ORDER=(
        "filewave-admin_${VERSION}_amd64.deb"
        "filewave-imaging-client_${VERSION}_amd64.deb"
        "ivs-kernel-${VERSION}-${REVISION}.x86_64.deb"
        "filewave-ivs_${VERSION}_amd64.deb"
    )
    DOWNLOAD_DIR="/tmp/filewave_install_$$"
    mkdir -p "$DOWNLOAD_DIR" || die "Failed to create download directory."
    trap 'rm -rf "$DOWNLOAD_DIR"' EXIT
    for package in "${PACKAGE_ORDER[@]"; do
        echo "Downloading $package..."
        for i in {1..3}; do
            curl -fSL "$BASE_URL/$VERSION/$package" -o "$DOWNLOAD_DIR/$package" && break
            echo "Retrying download ($i/3)..."
        done
    done
}

```

```

        sleep 5
    done
    if [[ ! -f "$DOWNLOAD_DIR/$package" ]]; then
        die "Failed to download $package after multiple attempts."
    fi
    echo "Installing $package..."
    gdebi -n "$DOWNLOAD_DIR/$package" || die "Failed to install $package"
done

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi
}

# If we reach this point, the requested version is newer than the installed version.
echo "Upgrading from $INSTALLED_VERSION to $VERSION-$REVISION..."
install_packages

# Clean up the download directory
if [[ -d "$DOWNLOAD_DIR" ]]; then
    rm -rf "$DOWNLOAD_DIR" || echo "Warning: Failed to remove download directory $DOWNLOAD_DIR"
fi

# 4. OS Upgrade
echo "Warning: This script will upgrade the entire OS. This is required for security."
if [[ "$auto_yes" == "true" ]]; then
    upgrade_confirm="yes"
else
    read -p "Do you wish to proceed with the OS upgrade? (yes/no): " upgrade_confirm < /dev/tty
fi
if [[ ! "$upgrade_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Skipping OS upgrade."
else
    echo "Upgrading the system..."
    apt-get update -y || die "Failed to update package list."
    DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -o Dpkg::Options::="--force-confold" || die "System
upgrade failed."
fi

# 5. Reboot the system to complete the installation
echo "The system will reboot in 15 seconds to complete the installation."
if [[ "$auto_yes" == "true" ]]; then
    reboot_confirm="yes"
else
    read -p "Do you want to reboot now? (yes/no): " reboot_confirm < /dev/tty
fi
if [[ "$reboot_confirm" =~ ^[Yy]([Ee][Ss])?$ ]]; then
    echo "Rebooting system in 15 seconds... Please remember that you can check the log file at $LOG_FILE after
the reboot to see the full details of the update."
    sleep 15
    reboot
else
    echo "Skipping reboot. Please remember to reboot manually to apply all changes."
fi

# The end

```