**prplOS Patch Automation Suite - Professional Guide**

**Table of Contents**

1. [Overview](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#overview)
2. [Features](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#features)
3. [System Requirements](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#system-requirements)
4. [Installation](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#installation)
5. [Configuration](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#configuration)
6. [Commands Reference](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#commands-reference)
7. [Environment Variables](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#environment-variables)
8. [Usage Examples](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#usage-examples)
9. [Error Handling](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#error-handling)
10. [HTML Dashboard](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#html-dashboard)
11. [Best Practices](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#best-practices)
12. [Troubleshooting](https://claude.ai/chat/16d4d779-6015-4afe-8ff9-d75e92f10413#troubleshooting)

**Overview**

The prplOS Patch Automation Suite is a comprehensive patch management system designed to automate the process of applying, validating, building, and reporting on patches for prplOS systems. It features intelligent error handling, HTML dashboard reporting, and flexible configuration options.

**Key Capabilities**

* **Automated patch validation and application**
* **Intelligent error detection and recovery**
* **HTML dashboard with real-time statistics**
* **Comprehensive logging and reporting**
* **Backup and rollback capabilities**
* **System resource monitoring**
* **Flexible configuration with defaults**

**Features**

**1. Smart Default Values**

* All commands work without arguments using sensible defaults
* User inputs override defaults when provided
* Environment variables for persistent configuration

**2. Advanced Error Handling**

* **Patch format validation**: Detects invalid patch files
* **Naming convention checks**: Warns about non-standard naming
* **Hunk failure detection**: Identifies and reports failed hunks
* **Fuzz factor support**: Attempts recovery with fuzz when needed
* **Offset detection**: Warns when patches apply with offset

**3. HTML Dashboard**

* **Real-time statistics**: Success rate, failure count, warnings
* **Interactive visualizations**: Progress bars, status badges
* **Detailed patch table**: Status, timestamps, and details
* **Professional dark theme**: Modern, responsive design
* **Exportable reports**: Save and share results

**4. Comprehensive Logging**

* **Multi-level logging**: ERROR, WARN, INFO, DEBUG
* **Timestamped entries**: Full audit trail
* **Color-coded output**: Easy visual parsing
* **Persistent log files**: Historical reference

**System Requirements**

**Minimum Requirements**

* **OS**: Linux-based system (prplOS compatible)
* **Shell**: Bash 4.0 or higher
* **Tools**: patch, tar, make, cmake (optional)
* **Memory**: 512MB RAM minimum
* **Disk**: 1GB free space for workspace

**Recommended Requirements**

* **CPU**: Multi-core processor for parallel builds
* **Memory**: 2GB+ RAM
* **Disk**: 5GB+ free space
* **Network**: Internet connection for updates

**Installation**

**1. Download the Script**

wget https://your-repo/prplos-patch-automation-suite.sh

# or

curl -O https://your-repo/prplos-patch-automation-suite.sh

**2. Make Executable**

chmod +x prplos-patch-automation-suite.sh

**3. (Optional) Install System-wide**

sudo cp prplos-patch-automation-suite.sh /usr/local/bin/

sudo chmod 755 /usr/local/bin/prplos-patch-automation-suite.sh

**4. Create Configuration File (Optional)**

cat > ~/.prplos-patch.conf << EOF

export PRPLOS\_WORKSPACE=$HOME/prplos-patches

export PRPLOS\_SOURCE=/opt/prplos/source

export PRPLOS\_PATCH\_LEVEL=1

export PRPLOS\_PARALLEL\_JOBS=4

EOF

# Source in your shell profile

echo "source ~/.prplos-patch.conf" >> ~/.bashrc

**Configuration**

**Directory Structure**

The suite creates the following directory structure:

$PRPLOS\_WORKSPACE/

├── patches/ # Patch files (.patch, .diff)

├── backups/ # Automatic backups before patching

├── logs/ # Detailed operation logs

├── results/ # HTML reports and summaries

└── build/ # Build output directory

**Patch Naming Convention**For best results, follow this naming convention:

NNNN-description.patch

Where:

* NNNN: 3-4 digit number for ordering
* description: Brief description (hyphens allowed)
* .patch or .diff: Standard extensions

Examples:

* 0001-fix-memory-leak.patch
* 0002-update-driver-api.patch
* 1001-security-update.diff

**Commands Reference**

**1. setup**

Initialize the patch automation environment.

./prplos-patch-automation-suite.sh setup

**Actions:**

* Creates workspace directories
* Validates environment
* Initializes logging

**2. apply**

Apply patches to the source tree.

# Apply all patches

./prplos-patch-automation-suite.sh apply

# Apply specific patch

./prplos-patch-automation-suite.sh apply /path/to/specific.patch

# Dry run mode

./prplos-patch-automation-suite.sh apply --dry-run

**Actions:**

* Validates patch format
* Checks for conflicts
* Creates backups
* Applies patches
* Generates report

**3. build**

Build the project after patching.

./prplos-patch-automation-suite.sh build

# With custom parallel jobs

./prplos-patch-automation-suite.sh build -j 8

**Actions:**

* Detects build system (make/cmake)
* Configures build
* Compiles with parallel jobs
* Reports build status

**4. clean**

Clean the build environment.

./prplos-patch-automation-suite.sh clean

**Actions:**

* Removes build artifacts
* Cleans reject files (.rej, .orig)
* Prepares for fresh build

**5. monitor**

Monitor system resources.

./prplos-patch-automation-suite.sh monitor

**Actions:**

* Reports CPU usage
* Shows memory utilization
* Displays disk usage
* Logs resource metrics

**6. report**

Generate HTML report.

./prplos-patch-automation-suite.sh report

**Actions:**

* Compiles statistics
* Generates HTML dashboard
* Creates summary file
* Updates results directory

**7. full**

Run complete automation workflow.

./prplos-patch-automation-suite.sh full

**Actions:**

* Runs setup
* Applies all patches
* Builds project
* Generates reports

**Environment Variables**

**Core Directories**

| **Variable** | **Default** | **Description** |
| --- | --- | --- |
| PRPLOS\_WORKSPACE | $HOME/prplos-patches | Main workspace directory |
| PRPLOS\_SOURCE | /opt/prplos/source | Source code directory |
| PRPLOS\_PATCH\_DIR | $WORKSPACE/patches | Patch files location |
| PRPLOS\_BACKUP\_DIR | $WORKSPACE/backups | Backup storage |
| PRPLOS\_LOG\_DIR | $WORKSPACE/logs | Log files location |
| PRPLOS\_RESULTS\_DIR | $WORKSPACE/results | Reports output |
| PRPLOS\_BUILD\_DIR | $WORKSPACE/build | Build output directory |

**Configuration Options**

| **Variable** | **Default** | **Description** |
| --- | --- | --- |
| PRPLOS\_PATCH\_LEVEL | 1 | Patch strip level (-p option) |
| PRPLOS\_DRY\_RUN | false | Enable dry run mode |
| PRPLOS\_PARALLEL\_JOBS | $(nproc) | Parallel build jobs |

**Setting Environment Variables**

# Temporary (current session)

export PRPLOS\_WORKSPACE=/custom/path

# Permanent (add to ~/.bashrc)

echo 'export PRPLOS\_WORKSPACE=/custom/path' >> ~/.bashrc

source ~/.bashrc

# Using env file

source /path/to/prplos.env

**Usage Examples**

**Example 1: Basic Patch Application**

# Place patches in default directory

cp \*.patch ~/prplos-patches/patches/

# Run full automation

./prplos-patch-automation-suite.sh full

**Example 2: Custom Workspace**

# Setup with custom directories

./prplos-patch-automation-suite.sh setup \

-w /opt/custom-workspace \

-s /usr/src/prplos

# Apply patches

./prplos-patch-automation-suite.sh apply

**Example 3: Dry Run Testing**

# Test patches without applying

./prplos-patch-automation-suite.sh apply --dry-run -v

# Review results

cat ~/prplos-patches/results/patch\_report\_\*.html

**Example 4: Production Deployment**

# Set production environment

export PRPLOS\_WORKSPACE=/var/prplos/production

export PRPLOS\_SOURCE=/opt/prplos/stable

export PRPLOS\_DRY\_RUN=false

# Run with specific patches

./prplos-patch-automation-suite.sh apply \

-p /var/prplos/approved-patches \

-L 0 \

-j 16

**Example 5: Automated CI/CD Integration**

#!/bin/bash

# ci-patch-apply.sh

# Load CI environment

source /etc/prplos-ci.env

# Run patch automation

/usr/local/bin/prplos-patch-automation-suite.sh full

# Check exit code

if [ $? -eq 0 ]; then

echo "Patches applied successfully"

# Copy results to artifacts

cp $PRPLOS\_RESULTS\_DIR/\* $CI\_ARTIFACTS\_DIR/

else

echo "Patch application failed"

exit 1

fi

**Error Handling**

**Patch Format Errors**

**Detection:**

* Invalid patch header
* Corrupted patch content
* Missing files

**Resolution:**

* Validates patch format before application
* Reports specific format issues
* Skips invalid patches

**Hunk Failures**

**Detection:**

* Failed hunks during application
* Context mismatches
* Line number conflicts

**Resolution:**

* Attempts fuzz factor 3
* Reports specific hunk failures
* Creates detailed error logs

**Naming Convention Warnings**

**Detection:**

* Non-standard patch names
* Missing sequence numbers
* Invalid extensions

**Resolution:**

* Warns about naming issues
* Continues with application
* Logs warnings in report

**Build Failures**

**Detection:**

* Configure errors
* Compilation failures
* Missing dependencies

**Resolution:**

* Captures build output
* Reports specific errors
* Preserves partial builds

**HTML Dashboard**

**Features**

1. **Real-time Metrics**
   * Total patches processed
   * Success/failure counts
   * Warning indicators
   * Success rate visualization
2. **Detailed Patch Table**
   * Patch filename
   * Application status
   * Timestamp
   * Error details
3. **Visual Design**
   * Dark theme for reduced eye strain
   * Responsive layout for all devices
   * Color-coded status indicators
   * Animated progress bars

**Accessing Reports**

Reports are saved in the results directory:

# View latest report

firefox ~/prplos-patches/results/patch\_report\_\*.html

# List all reports

ls -la ~/prplos-patches/results/

**Best Practices**

**1. Patch Organization**

* **Number patches sequentially**: 0001, 0002, etc.
* **Use descriptive names**: Include issue/feature reference
* **Group related patches**: Use number ranges (1xxx, 2xxx)
* **Document patches**: Include comments in patch files

**2. Testing Strategy**

* **Always dry run first**: Test before production
* **Review warnings**: Address before deployment
* **Check build status**: Ensure compilation success
* **Verify functionality**: Test patched system

**3. Backup Policy**

* **Automatic backups**: Created before each patch
* **Retention period**: Define based on needs
* **External backups**: For critical systems
* **Test restoration**: Verify backup integrity

**4. Monitoring**

* **Resource usage**: Monitor during patching
* **Log analysis**: Review for patterns
* **Report archival**: Keep historical data
* **Performance metrics**: Track application time

**Troubleshooting**

**Common Issues**

**1. Permission Denied**

# Error: Permission denied accessing source directory

# Solution: Run with appropriate permissions

sudo ./prplos-patch-automation-suite.sh apply

**2. Patch Not Found**

# Error: No patches found in directory

# Solution: Verify patch location

ls -la $PRPLOS\_PATCH\_DIR

**3. Build Failures**

# Error: Make command not found

# Solution: Install build tools

sudo apt-get install build-essential

**4. Hunk Failures**

# Error: Hunk #1 FAILED at 123

# Solution: Review patch context

patch --dry-run -p1 < problematic.patch

**Debug Mode**

Enable verbose output for detailed debugging:

./prplos-patch-automation-suite.sh apply -v

**Log Analysis**

# View latest log

tail -f ~/prplos-patches/logs/patch\_automation\_\*.log

# Search for errors

grep ERROR ~/prplos-patches/logs/\*.log

# Count warnings

grep -c WARN ~/prplos-patches/logs/\*.log

**Advanced Usage**

**Custom Patch Validation**

Add custom validation by modifying the validate\_patch\_format function:

# Example: Check for specific headers

if ! grep -q "^Signed-off-by:" "$patch\_file"; then

log WARN "Patch missing Signed-off-by header"

fi

**Integration with Version Control**

# Git integration example

git format-patch -o $PRPLOS\_PATCH\_DIR HEAD~5

./prplos-patch-automation-suite.sh apply

**Automated Scheduling**

# Crontab entry for nightly patch application

0 2 \* \* \* /usr/local/bin/prplos-patch-automation-suite.sh full >> /var/log/prplos-patch.log 2>&1

**Security Considerations**

1. **Validate patch sources**: Only apply trusted patches
2. **Review patch content**: Check for malicious code
3. **Limit permissions**: Run with minimal required privileges
4. **Audit logs**: Monitor for unauthorized changes
5. **Backup encryption**: Protect sensitive backups

**Support and Maintenance**

**Log Rotation**

# Add to /etc/logrotate.d/prplos-patch

/home/\*/prplos-patches/logs/\*.log {

weekly

rotate 4

compress

missingok

notifempty

}

**Cleanup Script**

# cleanup-old-reports.sh

find $PRPLOS\_RESULTS\_DIR -name "\*.html" -mtime +30 -delete

find $PRPLOS\_BACKUP\_DIR -name "\*.tar.gz" -mtime +60 -delete

**Conclusion**

The prplOS Patch Automation Suite provides a robust, enterprise-ready solution for patch management. With its comprehensive error handling, beautiful HTML reporting, and flexible configuration options, it streamlines the patch application process while maintaining safety and auditability.

For additional support or feature requests, please consult the prplOS documentation or community forums.