# JUMBO BLOCK Rescue
# User Manual



# JUMBO BLOCK Rescue

## Resilience does not end at the structure.

Anyone who has so far associated JUMBO BLOCK mainly with rainwater retention, water storage and protecting vulnerable infrastructure from heavy rainfall and flooding knows only part of what lies behind JUMBO BLOCK.

Because critical infrastructure is not protected by concrete alone.

It only works if **structures, water management, technology, data, documentation, communication and IT** interact even under exceptional conditions.

This is precisely where **JUMBO BLOCK Rescue** comes in.

JUMBO BLOCK Rescue is a self-contained, bootable rescue, diagnostic, engineering and work system for situations in which the regular IT infrastructure is no longer available or only available to a limited extent.

It was not developed to be yet another operating system.

It was developed so that you **can keep working when the actual system no longer can.**

## KRITIS means: expecting disruptions before they occur

With the KRITIS Umbrella Act, the resilience of critical facilities is moving even more firmly into the spotlight.

For operators, resilience therefore means more than the question of how a facility works in normal operation.

The decisive questions begin where normal operation ends:

**What happens during heavy rainfall or flooding?**

**What happens with a damaged storage device?**

**What happens when a computer no longer starts?**

**What happens after a cyber incident?**

**Are project documents, technical documentation and engineering tools still available when central IT systems fail?**

**Can an organisation still read, check, back up and further process its technical information?**

And, in very practical terms:

**Is it still possible to work at all?**

JUMBO BLOCK Rescue is our answer to part of precisely these questions.

# One USB system. Four technical levels.

JUMBO BLOCK Rescue combines on one bootable system four worlds of tools that are normally considered separately.

## A – JB-Rescue

The first level is the actual technical rescue and diagnostic environment.

**35 specialised tools** are available for data recovery, storage device diagnostics, partitioning, file systems, malware and rootkit checks, network analysis, encryption, hardware diagnostics, remote access and technical document editing.

TestDisk, PhotoRec, GNU ddrescue, GParted, SMART diagnostics and NVMe diagnostics, ClamAV, Wireshark, Nmap, Cryptsetup, GnuPG and many other tools form a comprehensive technical tool chain.

One principle applies here:

**Understand first. Then back up. Then act.**

Original and damaged storage devices should remain as unchanged as possible. Unstable media are backed up first. Wherever possible, analyses and repairs are carried out on images or working copies.

For us, rescue does not mean hectic repairing.

Rescue means **a controlled technical approach**.

## B – MX tools

The second level adds carefully selected functions of the technical MX base to this tool chain.

Boot repair, UEFI management, chroot rescue, creation of bootable rescue media, system images and signature verification support the recovery and operation of the environment itself.

MX is the technical basis of JUMBO BLOCK Rescue – **not its purpose**.

JUMBO BLOCK Rescue is expressly intended to examine, back up and restore systems and data from other operating system worlds as well.

## C – Accessories

An emergency system must be able to do more than just find faults.

That is why practical work tools are also part of the environment: file manager, file search, archive manager, text editor, hash verification, network management, terminal and screenshot tools.

Because in a real disruption, information does not only have to be rescued.

It also has to be possible to:

**find, check, copy, document and pass it on.**

## D – JUMBO BLOCK Engineering Suite

And it is exactly at this point that JUMBO BLOCK Rescue goes one decisive step further.

The system additionally contains the **JUMBO BLOCK Engineering Suite**.

This means that the function of the rescue system does not end at:

**“The computer is running again.”**

It additionally provides a local working environment for essential technical, project-related and documentation processes.

Hydrology, plant design, budgeting, inspection, project management, tendering, scientific calculations, unit conversion, notes, tables, document creation, document viewing, text and code editing, PDF tools, conversion, archiving and file management are directly available within the rescue environment.

The applications work largely locally and are deliberately designed not to require central cloud services for their core functions.

This means:

**Rescue does not end with the recovery of data.**

It can be the point from which work continues.

# JUMBO BLOCK is far more than rainwater retention

The origin of JUMBO BLOCK lies in a very physical challenge:

**making water controllable where it becomes a danger to infrastructure.**

Retention and storage spaces are created beneath sealed surfaces. Rainwater can be retained, stored and managed in a controlled manner, while the surface can continue to be used for infrastructure.

In the context of critical infrastructure in particular, however, retention is not an isolated issue.

Heavy rainfall can:

- flood access roads,
- endanger technical installations,
- render operational areas unusable,

- impair energy and communication infrastructure,
- trigger business interruptions.

Anyone who considers such risks quickly realises:

**Physical and digital resilience meet at the same site.**

A flooded plant room is not purely a water problem.

A failed computer is not purely an IT problem.

A lost engineering data set is not purely a file problem.

And an existing emergency plan is of little help if it can no longer be accessed at the decisive moment.

That is why we do not think of JUMBO BLOCK as a single concrete product.

We think of it as an **infrastructure concept**.

# From retention to resilience

JUMBO BLOCK therefore combines different technical disciplines:

**Water management.**
**Structural engineering.**
**Infrastructure planning.**
**Engineering.**
**Inspection and documentation.**
**Data integrity.**
**IT resilience.**
**Recovery.**

At first glance, this may seem unusual.

For us, it is consistent.

Because the decisive question is not:

**“Which product do we have?”**

But rather:

**“What has to work when it really matters?”**

From this perspective, an underground retention space, a structural engineering model, a local project database, a SHA-256 hash, a data recovery tool and a bootable rescue system suddenly belong to the same idea:

**Maintaining the ability to act.**

# For everyone whose KRITIS thinking does not stop at paper

JUMBO BLOCK Rescue is aimed at people and organisations that bear responsibility for functioning infrastructure.

At operators.

At those with technical responsibility.

At engineers.

At IT departments.

At municipalities and utilities.

At industry, data centres and other operators of technically demanding sites.

And at everyone who, in the course of rising resilience requirements, will at some point have to ask themselves a very simple question:

**What do we actually have available when our usual infrastructure fails?**

JUMBO BLOCK Rescue is no substitute for complete emergency, information security or business continuity management.

Nor is it a promise that every damaged system or every lost data set can be restored.

It is something very practical:

**a comprehensive technical tool and working environment that is meant to be available when the normal environment may no longer be.**

# 70 tools and functions. One idea.

Baseline 20261001 of JUMBO BLOCK Rescue comprises the four blocks documented in this manual, with a total of **70 programs and functions**.

But the number alone is not what matters.

What matters is how these tools work together.

A storage device can be diagnosed.

A damaged medium can be backed up first.

Partitions and file systems can be examined.

Lost files can be searched for.

Network traffic can be analysed.

Hardware can be diagnosed.

Data can be encrypted and its integrity verified.

Rescued files can be searched, sorted and archived.

Technical documents can be read and edited.

Project information can be further processed locally.

And essential tools of the JUMBO BLOCK Engineering Suite are directly available.

That is the difference between a collection of utilities and a **rescue environment**.

# Perhaps that is exactly what KRITIS is.

Not the notion that nothing ever fails.

But the ability to be prepared **when something fails**.

That is why JUMBO BLOCK protects infrastructure not only where water becomes visible.

We address the question of how technical systems can become more resilient –
**underground, on the surface, in engineering and in the data.**

JUMBO BLOCK Rescue makes this attitude tangible.

Bootable.
Local.
Usable offline.
Technically traceable.
And made for the moment when improvisation alone is no longer enough.

**JUMBO BLOCK Rescue**

**Because resilience does not end at the structure.**

# Note on the table of contents

This manual deliberately has a **detailed table of contents**.

JUMBO BLOCK Rescue combines a large number of different tools for data recovery, diagnostics, networking, security, system recovery, general work tasks and the JUMBO BLOCK Engineering Suite. Accordingly, the situations in which this manual is needed can vary just as widely.

The table of contents is therefore not only intended to give an overview of the structure of the manual. It is also intended as a **quick technical signpost**.

Anyone using JUMBO BLOCK Rescue for the first time can use the four blocks to gain a structured overview of the system.

Anyone who already knows which tool is needed can jump directly to the relevant program or subject area.

And in an actual rescue situation, the decisive question should be answered as quickly as possible:

**“Where do I find the tool and the information I need right now?”**

That is why the individual programs and functions in the table of contents are not artificially grouped together or hidden. The detailed structure is intentional.

# Content

JUMBO BLOCK Rescue....2
Resilience does not end at the structure....2
KRITIS means: expecting disruptions before they occur....2
One USB system. Four technical levels....3
A – JB-Rescue....3
B – MX tools....3
C – Accessories....3
D – JUMBO BLOCK Engineering Suite....4
JUMBO BLOCK is far more than rainwater retention....4
From retention to resilience....5
For everyone whose KRITIS thinking does not stop at paper....6
70 tools and functions. One idea....6
Perhaps that is exactly what KRITIS is....7
Note on the table of contents....8
Block A....65
Tool groups....65
Assignment of areas, programs and versions....67
A.1 TestDisk....69
Purpose....69
Typical use cases....69
Starting and basic operation....69
Analysing a partition table....70

Recovering a partition....70
Restoring deleted files....70
Boot sectors and file system structures....70
Procedure for physically damaged storage devices....71
TestDisk and PhotoRec....71
Safety notes....71
Role in JUMBO BLOCK Rescue....72
A.2 PhotoRec....73
Purpose....73
Typical use cases....73
Working principle....73
Starting and selecting the storage device....74
Selecting the source data....74
Selecting the file system type....74
Entire partition or free space only....74
Selecting the file types to be recovered....75
Selecting the destination directory....75
Carrying out the recovery....75
File names and directory structure....76
Options for difficult cases....76
Working with storage device images....76
Fragmented files....77
Resuming an interrupted recovery....77
Safety notes....77
TestDisk or PhotoRec?....77
Role in JUMBO BLOCK Rescue....78
A.3 GNU ddrescue....79
Purpose....79
Distinction from dd and dd_rescue....79
Typical use cases....79
Basic principle....80
Source and target....80
The mapfile....80
Resuming a rescue process....81
Rescue phases....81
First rescue pass....81
Further rescue attempts....81
Direct storage device access....82
Writing to a block device....82
Image file or physical clone....82
Sizing the target medium correctly....83
Working independently of the file system....83
Avoid mounting storage devices where possible....83
No repair by ddrescue....83
Handling a failing storage device....84
Limitations....84
Safety notes....84
Role in JUMBO BLOCK Rescue....85
A.4 Foremost....86
Purpose....86
Typical use cases....86

Working principle....86
Supported built-in file types....87
Basic call....87
Selecting the output folder....88
Selecting specific file types....88
Verbose mode....89
Quick mode....89
Quiet mode....89
Audit-only mode....89
Option -a....90
Block size....90
Skipping areas....90
Indirect block detection....90
Time-stamped output folders....91
Custom file signatures....91
Structure of foremost.conf....91
Binary signatures....92
Audit file....92
Foremost and storage device images....92
Limitations of file carving....92
Original file names and directory structure....93
Foremost and PhotoRec....93
Safety notes....93
Role in JUMBO BLOCK Rescue....94
A.5 Extundelete....95
Purpose....95
Area of application....95
Basic call....95
Restoring a single file....96
Restoring a directory....96
Restoring all files that can be found....96
Restoring several selected files....97
Restoring on the basis of an inode number....97
Examining the superblock....97
Examining the journal....98
Displaying inode information....98
Displaying block content....98
Restricting by time....98
External journal....99
Backup superblock....99
Why acting quickly is important....99
Original or image....100
No guarantee of recovery....100
Distinction from TestDisk....100
Distinction from PhotoRec and Foremost....100
Distinction from GNU ddrescue....101
Limitations....101
Safety notes....101
Role in JUMBO BLOCK Rescue....101
A.6 GParted....103
Purpose....103

Significance in the rescue system....103
Program interface....103
Examining partitions....103
Supported operations....104
Creating a partition table....104
Creating a new partition....104
Deleting a partition....105
Formatting a partition....105
Resizing a partition....105
Moving a partition....105
Copying and pasting a partition....106
Checking a partition....106
Partition flags....106
Pending operations....106
Check before applying....106
Mounted partitions....107
Encrypted partitions....107
File system support....107
Storage device types....108
Special feature of GParted 1.6.0....108
GParted and TestDisk....108
GParted and GNU ddrescue....109
Typical rescue workflow....109
Limitations....109
Safety notes....109
Role in JUMBO BLOCK Rescue....110
A.7 Disks / GNOME Disks....111
Purpose....111
Storage device overview....111
Storage devices and block devices....112
Partitions and volumes....112
Mounting and unmounting....112
SMART data....112
SMART self-tests....113
Distinction from A.8 – SMART diagnostics....113
Performance test....113
Write benchmark....114
Storage device images....114
Creating a storage device image....114
Restoring a storage device image....114
Distinction from GNU ddrescue....115
Creating and managing partitions....115
Formatting....115
Formatting is not secure erasure....115
Changing the file system size....116
Checking and repairing a file system....116
Powering off a drive....116
GNOME Disks and GParted....116
Typical rescue workflow....117
Limitations....117
Safety notes....117

Role in JUMBO BLOCK Rescue....118
A.8 SMART diagnostics....119
Purpose....119
What is SMART?....119
Supported device classes....119
Identifying the device....120
Searching for devices automatically....120
Displaying device information....120
SMART health status....121
Comprehensive SMART information....121
Extended full output....122
SMART attributes with ATA devices....122
Normalised values and raw values....122
Error logs....123
Self-test log....123
Internal SMART self-tests....123
Short self-test....123
Long or extended self-test....124
Aborting a self-test....124
Captive mode....124
Enabling or disabling SMART....124
Specifying the device type....125
USB enclosures and adapters....125
RAID controllers....125
NVMe and SMART....126
Temperature....126
SMART is not a surface scan....126
SMART is not a data recovery program....126
SMART diagnostics and GNU ddrescue....127
SMART diagnostics and GNOME Disks....127
JSON output....127
Drive database....128
Limitations of SMART diagnostics....128
Safety notes....128
Typical rescue workflow....128
Role in JUMBO BLOCK Rescue....129
A.9 NVMe diagnostics....130
Purpose....130
NVMe....130
Controller and namespace....130
Listing detected NVMe devices....131
Identifying the controller....131
Identifying the namespace....132
SMART/health information....132
Global and namespace-specific SMART data....132
Typical health information....133
Critical Warning....133
Temperature....133
Available Spare....134
Percentage Used....134
Data Units Read and Data Units Written....134

Power Cycles and Power On Hours....134
Unsafe Shutdowns....134
Media and Data Integrity Errors....135
Error log....135
Number of error log entries....135
SMART log in JSON format....135
Outputting raw data....136
Device Self-Test....136
Short self-test....136
Extended self-test....137
Self-test status....137
Aborting a self-test....137
Waiting for the self-test....137
Self-test log....138
Distinction from SMART diagnostics....138
Diagnosis versus administration....138
Format....139
Namespace management....139
Write commands....139
Firmware functions....140
Controller reset....140
Safe initial diagnosis....140
Handling conspicuous NVMe drives....140
NVMe diagnostics and GNU ddrescue....141
Limitations....141
Safety notes....141
Version note....142
Role in JUMBO BLOCK Rescue....142
A.10 ClamAV Scan....143
Purpose....143
Signature database....143
Basic syntax....143
Scanning a single file....144
Scanning a directory....144
Recursive scan....144
Scanning the entire system....144
Scanning a mounted storage device....145
Detection reports....145
Scan summary....145
Writing the scan report to a file....145
Displaying infected files only....146
Quiet mode....146
Audible detection alert....146
Alternative signature database....146
Official signatures only....147
PUA detection....147
Archives and containers....147
Maximum file size....147
Maximum scan size per container....148
Using a file list....148
File system boundaries....148

Moving infected files....149
Copying infected files....149
Deleting files automatically....149
Heuristic alerts....150
Encrypted archives and documents....150
Temporary files....150
clamscan and clamdscan....150
Not a complete endpoint security system....151
Scanning a Windows storage device....151
Advantage of an externally booted rescue system....151
A detection is not proof....152
Access rights....152
Scanning and damaged storage devices....152
ClamAV and data recovery....152
Typical rescue workflow....153
Safety notes....153
Role in JUMBO BLOCK Rescue....153
A.11 ClamAV signatures....155
Purpose....155
Significance of the signature databases....155
FreshClam....156
Update process....156
Incremental update....156
Damaged database....156
Database types....157
CVD....157
CDIFF....157
Digital signatures....157
Manual update....157
Interactive operation....158
Daemon mode....158
DNS-based version check....158
Proxy support....158
Configuration file....159
Checking the configuration....159
Database directory....159
Database ownership and permissions....159
Update log....160
Network connection....160
Offline operation....160
Offline does not mean ineffective....160
Updating before a scan....161
Do not force an internet connection....161
Controlled offline update....161
FreshClam and local mirrors....162
Signatures and engine compatibility....162
Update errors....162
Repeated update attempts....162
Signatures are no guarantee....163
Signature version and scan report....163
Separating update and scan....163

Typical rescue workflow with an internet connection....163
Typical rescue workflow without an internet connection....164
Safety notes....164
Role in JUMBO BLOCK Rescue....164
A.12 Chkrootkit....166
Purpose....166
What is a rootkit?....166
Components of Chkrootkit....166
Basic call....167
Output....167
Displaying available tests....167
Running a single test....168
Quiet mode....168
Expert mode....168
Debug mode....169
Do not combine options....169
Examining an external system....169
Advantage of external examination....170
Using alternative trustworthy programs....170
Significance for JUMBO BLOCK Rescue....170
Skipping NFS directories....171
Excluding file system types....171
Excluding known false positives....171
Sniffer test and false positives....171
Sniffer exceptions....172
Promiscuous mode....172
Checking system programs....172
LKM checks....173
Login traces....173
chkwtmp....173
chklastlog....174
Known rootkit characteristics....174
Chkrootkit is not a virus scanner....174
No automatic clean-up....174
False positives....175
No finding is not proof either....175
Trust problem of a compromised system....175
Typical rescue workflow....175
Procedure in the event of a warning....176
Evidence preservation....176
Chkrootkit on a damaged storage device....176
Chkrootkit and ClamAV....177
Limitations....177
Safety notes....177
Role in JUMBO BLOCK Rescue....177
A.13 Rkhunter....179
Purpose....179
Basic check principle....179
Basic system check....180
Interactive process....180
Warnings....180

No finding is no guarantee of security....181
Displaying supported checks....181
Enabling individual checks....181
Disabling checks....181
Configuration....182
Checking the configuration....182
File properties....182
Hash methods....183
Properties database....183
--propupd....183
Do not use --propupd blindly....184
Selective --propupd....184
Package manager support....184
Updating Rkhunter data....185
Difference between --update and a software update....185
Version check....185
Offline operation....185
No obligation to connect to a network....186
Logging....186
Previous log....186
Displaying warnings only....187
Summary....187
Colours....187
Quiet mode....187
Network checks....188
System startup files....188
Hidden files and directories....188
Rootkit-specific checks....188
Shared memory check....189
Additional auxiliary programs....189
Chkrootkit and Rkhunter....189
Rkhunter and ClamAV....189
Special feature of a rescue system....190
Consequence for JUMBO BLOCK Rescue....190
Handling warnings correctly....190
Never use --propupd to eliminate warnings....190
Physically damaged storage devices....191
No automatic clean-up....191
Limitations....191
Safety notes....191
Typical rescue workflow....192
Role in JUMBO BLOCK Rescue....192
A.14 GUFW Firewall....193
Purpose....193
Area of use in the rescue system....193
GUFW and UFW....194
Enabling and disabling the firewall....194
Checking the status....194
Basic principle of a host-based firewall....195
Incoming and outgoing traffic....195
Default policies....195

Typical security approach....196
Rules....196
Allow....196
Deny....196
Reject....197
Limit....197
Ports....197
Protocols....197
IP addresses....198
Subnets....198
Rule order....198
Deleting rules....198
Profiles....199
Logging....199
GUFW and logging....199
Firewall and network diagnostics....199
GUFW and SSH....200
GUFW and Nmap....200
GUFW and Wireshark....200
GUFW and ClamAV....200
Reset....201
Disabling the firewall is not a diagnosis....201
A firewall does not protect against all attacks....201
Outgoing traffic....202
Unknown networks....202
Rescue system as a maintenance computer....202
Document firewall rules before changes....203
Typical rescue workflow....203
Safety notes....203
Limitations....204
Technical classification in JUMBO BLOCK Rescue....204
A.15 Binwalk....206
Purpose....206
Typical areas of use....206
Basic call....207
Offsets....207
Signature analysis....207
A signature match is not complete proof....208
Output of invalid matches....208
Including specific signatures....208
Excluding signatures....209
Searching for a byte sequence....209
Opcode scan....209
Custom magic file....210
Automatic extraction....210
External extraction programs....210
Extraction directory....211
Recursive extraction....211
Recursion depth....211
Limiting the number of extracted files....212
Limiting the size of extracted files....212

Carving without extraction programs....212
Recursive analysis with care....213
Entropy analysis....213
Significance of high entropy....213
Fast entropy analysis....214
Saving the entropy graph....214
Limiting the scan range....214
Base address....215
Changing the byte order....215
Logging results....215
CSV output....216
Quiet mode....216
Verbose mode....216
Typical first procedure....216
Example of a firmware examination....217
Binwalk and Hexedit....217
Binwalk and Foremost....217
Binwalk and PhotoRec....218
Binwalk is not a malware scanner....218
Binwalk is not a disassembler....218
Binwalk is not a data recovery program for damaged hardware....218
Do not change original files unnecessarily....218
Untrustworthy extracted content....218
External auxiliary programs....219
False positives....219
Limitations....219
Safety notes....219
Technical classification in JUMBO BLOCK Rescue....220
A.16 Hexedit....221
Purpose....221
Area of use in the rescue system....221
Basic call....222
Screen layout....222
Hexadecimal values....222
Offset....223
Navigation....223
Jumping to the beginning of the file....223
Jumping to the end of the file....223
Jumping to a specific offset....223
Decimal and hexadecimal positions....224
Search....224
Searching for strings....224
Searching for hexadecimal values....225
Continuing a search....225
Direct editing....225
Hexadecimal editing....225
Character-based editing....226
Switching between the hex and character areas....226
Changes are initially editable....226
Saving changes....226
Quitting....226

Read-only mode....227
Read-only as the default for examinations....227
Devices and storage devices....227
Direct write access to block devices....228
Image instead of original....228
Physically damaged storage devices....228
Binwalk and Hexedit....228
Example of an analysis workflow....229
Hexedit and file signatures....229
Damaged file headers....230
No automatic interpretation....230
No automatic structure check....230
Overwriting instead of structural editing....230
Character encoding....230
A search is not a file analysis....231
No malware detection....231
Not a disassembler....231
No file carving....231
No automatic data recovery....231
Typical rescue scenarios....232
Safe workflow....232
Forensic considerations....232
Limitations....233
Safety notes....233
Technical classification in JUMBO BLOCK Rescue....233
A.17 Nmap....235
Purpose....235
Area of use in the rescue system....235
Legal and operational prerequisite....236
Basic syntax....236
Default scan....236
Host discovery....237
Host discovery only....237
“Ping scan” does not mean ICMP only....237
Local Ethernet....237
Skipping host discovery....238
Consequence of -Pn....238
Listing targets without scanning....238
Ports....238
1.000 ports by default....239
Selecting specific ports....239
All ports....239
Fast scan....239
Top ports....240
Distinguishing TCP and UDP....240
The six port states....240
open....240
closed....241
filtered....241
unfiltered....241
open|filtered....241

closed|filtered....242
Results depend on perspective....242
TCP SYN scan....242
SYN scan and privileges....242
TCP connect scan....242
Difference between -sS and -sT....243
UDP scan....243
UDP results....243
UDP scans can be slow....243
Combining TCP and UDP....244
ACK scan....244
Other scan methods....244
Service detection....245
Version and service detection....245
Possible -sV information....245
nmap-service-probes....246
Version detection intensity....246
Light version scan....246
All version probes....246
Operating system detection....247
OS fingerprint....247
OS detection is an estimate....247
More aggressive OS guessing....248
Aggressive detection....248
Use -A deliberately....248
Nmap Scripting Engine....248
Default scripts....249
NSE is not inherently passive or harmless....249
NSE categories....249
safe does not mean risk-free....250
Intrusive scripts....250
vuln....250
exploit, brute and dos....251
Single NSE script....251
Help for an NSE script....251
Script arguments....251
Output....252
Normal output....252
XML output....252
Grepable output....252
Several formats at once....252
Verbose output....253
Open or potentially open ports only....253
Network interfaces and routes....253
Timing....253
Do not confuse timing templates with scan quality....254
Scope affects run time....254
Influence of firewalls....254
GUFW and Nmap....255
Nmap and ARP Scan....255
Nmap and Wireshark....255

Nmap and TCPDump....255
Nmap and MTR....255
Nmap and Dig....255
Typical rescue workflow....256
Example: unknown device in the local network....256
Example: service not reachable....257
Example: UDP service....257
Documenting scan results....257
Limitations....257
Safety notes....258
Technical classification in JUMBO BLOCK Rescue....258
A.18 Wireshark....260
Purpose....260
Area of use in the rescue system....260
Wireshark is not a port scanner....261
Live capture and offline analysis....261
Capture interfaces....261
Several interfaces....262
Capture privileges....262
Dumpcap....262
Promiscuous mode....262
Limitations in switched networks....263
Wi-Fi and monitor mode....263
Starting a capture....263
The three central view areas....264
Packet List....264
Packet Details....264
Packet Bytes....265
Protocol dissectors....265
A port number is not the same as a protocol....265
Decode As....265
Capture filters and display filters....265
Capture filter....266
Capture filter syntax....266
Use capture filters sparingly....266
Display filter....267
Example tcp....267
Example dns....267
Example IP address....267
Example TCP port....267
Source and destination direction....267
Logical combinations....268
Filters do not change the data....268
Finding packets....268
Marking packets....268
Time display....268
Time stamps....269
TCP connection setup....269
TCP connection teardown....269
TCP retransmissions....269
Checksums....270

DNS analysis....270
DHCP analysis....270
ARP analysis....271
ICMP analysis....271
HTTP....271
TLS....271
Encryption remains encryption....272
Follow Stream....272
Follow TCP Stream....272
Follow Stream does not change the capture....272
Statistics....272
Protocol Hierarchy....273
Endpoints....273
Conversations....273
I/O Graphs....273
Expert Information....274
Expert Information is not a final verdict....274
Capture files....274
PCAPNG....274
PCAP....274
Capture files contain sensitive data....275
Do not pass on capture files uncontrolled....275
Capture File Properties....275
Capture comments....276
Ring buffer....276
Stopping automatically....276
Packet loss during capture....276
Snap length....277
Name resolution....277
Exporting packets....277
Exporting packet bytes....277
Export Objects....277
Capinfos....278
Editcap....278
Mergecap....278
Wireshark and TCPDump....278
Wireshark and Nmap....278
Wireshark and GUFW....279
Wireshark and Dig....279
Wireshark and MTR....279
Typical rescue workflow....279
Example: DNS problem....280
Example: TCP service not reachable....281
Example: DHCP problem....281
Limitations....281
Safety notes....282
Technical classification in JUMBO BLOCK Rescue....282
A.19 TCPDump....284
Purpose....284
Area of use in the rescue system....284
Basic call....284

Displaying network interfaces....285
Selecting the interface....285
any....285
Capture permissions....285
Standard output....286
Limiting the number of packets....286
Disabling name resolution....286
Displaying ports numerically as well....287
Verbose output....287
Time stamps....287
Absolute time....288
Filter expression....288
BPF filters....288
Filtering a host....288
Source host....288
Destination host....289
Filtering a network....289
Filtering a port....289
Filtering TCP....289
Filtering UDP....289
TCP port....290
UDP port....290
DNS traffic....290
ARP....290
ICMP....291
IPv6....291
Logical combinations....291
AND....291
OR....291
NOT....291
Brackets....292
Capture filters reduce the amount of data....292
Snaplen....292
Truncated packets....293
Writing a capture to a file....293
-w saves packets, not terminal text....293
Reading a capture file....293
Applying filters to saved captures....293
Offline analysis....294
Handover to Wireshark....294
Terminal and GUI complement each other....294
ASCII output....295
Hex and ASCII output....295
Hex output....295
Link-layer header....295
Packet content can be sensitive....295
Capture direction....296
Incoming traffic....296
Outgoing traffic....296
Promiscuous mode....297
Promiscuous mode does not automatically see the entire LAN....297

Capture buffer....297
Packet loss....297
An incomplete capture can be misleading....298
Rotation by file size....298
Rotation by time....298
Limiting the number of files....298
Long-term captures....298
Immediate output....299
Capture via pipe....299
No packet injection....299
TCPDump is mostly passive....299
TCPDump and Nmap....299
TCPDump and Wireshark....300
TCPDump and ARP Scan....300
TCPDump and MTR....300
TCPDump and Dig....300
TCPDump and GUFW....301
Typical rescue workflow....301
Example: DNS does not work....302
Example: TCP connection....302
Example: ARP problem....302
Example: capture for Wireshark....303
Example: limited number of packets only....303
Capture files as sensitive data....303
Data protection and authorisation....303
Encrypted traffic....304
Limitations....304
Safety notes....304
Technical classification in JUMBO BLOCK Rescue....305
A.20 ARP Scan....306
Purpose....306
Area of use in the rescue system....306
ARP....307
Basic principle....307
ARP only works locally....307
Consequence for rescue use....307
IPv4....307
Basic syntax....308
Single target....308
Several targets....308
CIDR network....308
Address range....308
Network and netmask....309
Determining the local network automatically....309
Meaning of --localnet....309
Network and broadcast address....309
Selecting the interface....309
Automatic interface selection....310
Interface without an IPv4 address....310
Permissions....310
Standard output....310

MAC address....311
MAC addresses can be changed....311
Vendor detection....311
IEEE assignments....311
Additional assignments....312
Interpreting the manufacturer information correctly....312
Virtualisation....312
No vendor information....312
Updating the OUI file....312
Minimalist output....313
Host lines only....313
User-defined output format....313
Resolving host names....313
Numerical target addresses only....314
Targets from a file....314
--file and --localnet....314
Retries....315
Why retries?....315
Fewer retries....315
Timeout....315
Backoff....316
Bandwidth....316
Why limit bandwidth?....316
Higher bandwidth....316
Packet interval....316
Duplicates....317
Duplicate responses can be relevant....317
ARP and firewalls....317
A device that does not respond to ping can be visible....317
An ARP finding does not prove a service....318
ARP Scan and Nmap....318
ARP Scan and Wireshark....318
ARP Scan and TCPDump....318
ARP Scan and MTR....319
Typical rescue workflow....319
Example: IP address of a device unknown....319
Example: device does not respond to ping....320
Example: unknown device....320
Example: wrong interface....320
Limitations....320
Safety notes....321
Technical classification in JUMBO BLOCK Rescue....321
A.21 MTR....323
Purpose....323
Area of use in the rescue system....323
Basic principle....324
Step-by-step TTL....324
Difference from traceroute....324
Difference from ping....325
Basic call....325
Interactive display....325

Loss%....326
Snt....326
Last....326
Avg....326
Best....326
Wrst....326
StDev....326
Latency....326
Jitter and fluctuations....327
The most important interpretation error....327
Why?....327
Decisive interpretation principle....327
Continuing loss....328
The target is particularly important....328
Hops that do not respond....328
Firewalls and filters....328
Name resolution....329
Disabling DNS....329
Host name and IP address together....329
Forcing IPv4....329
Forcing IPv6....330
Why test IPv4 and IPv6 separately?....330
Report mode....330
Number of measurement cycles....331
Saving the report to a file....331
Why several measurement cycles?....331
Wide report....331
Example for rescue documentation....332
Interval....332
Measurement load....332
Packet size....332
Why packet size can be relevant....333
Selecting the interface....333
Specifying the source address....333
Customising report fields....333
JSON....334
XML....334
CSV....335
ICMP....335
UDP....335
TCP....335
Destination port....335
Why different probe protocols are important....335
Example....336
MPLS....336
Autonomous system information....336
Interactive control....336
Resetting statistics....337
The route can change....337
Load balancing....337
Asymmetric routing....337

ICMP prioritisation....338
Example: apparent loss at an intermediate hop....338
Example: loss continues....338
Example: latency jump....338
Geographical distance....339
Example: strong fluctuations....339
Typical rescue workflow....339
MTR and Nmap....340
MTR and Wireshark....340
MTR and TCPDump....340
MTR and ARP Scan....341
MTR and Dig....341
Limitations....341
Safety notes....341
Technical classification in JUMBO BLOCK Rescue....342
A.22 Dig....343
Purpose....343
Use in the rescue system....343
DNS is a diagnostic layer of its own....344
Basic syntax....344
Default resolver....345
No usable resolver configured....345
Querying a specific DNS server....345
Why @server matters....345
Distinguishing direct server queries from the system resolver....345
Structure of the standard output....346
Header....346
Status....346
NOERROR....347
NXDOMAIN....347
No response from the server....347
QUESTION SECTION....348
ANSWER SECTION....348
TTL....348
AUTHORITY SECTION....348
ADDITIONAL SECTION....349
Query Time....349
SERVER....349
MSG SIZE....349
A record....349
AAAA record....350
Examining A and AAAA separately....350
CNAME....350
MX....350
NS....350
SOA....351
Serial number....351
TXT....351
CAA....352
SRV....352
PTR and reverse DNS....352

IPv4 reverse DNS....352
IPv6 reverse DNS....353
Reverse DNS is not forward DNS....353
Short output....353
When +short is useful....353
Answer Section only....353
Forcing IPv4 transport....354
Forcing IPv6 transport....354
Important distinction....354
UDP and TCP....355
Why test TCP?....355
DNS does not use UDP only....355
DNS over TLS....356
Non-standard port....356
Timeout....356
Retries....356
Recursion....357
Recursive resolver....357
Authoritative name server....357
Cache differences....357
+trace....357
What +trace shows....358
+trace is not a standard resolver query....358
@server together with +trace....358
DNSSEC data....358
+dnssec does not automatically mean full validation....359
+trace and DNSSEC....359
DNS flags....359
aa....359
ra....359
ad....359
Multiple queries....360
Batch file....360
User configuration .digrc....360
Reproducible diagnostics....360
Exit codes....360
Implications for scripts....361
Zone transfer....361
AXFR is not a general discovery method....361
Typical rescue procedure....361
Example: browser reports “Server not found”....362
Example: Dig receives no response....362
Example: A works, AAAA does not....363
Example: resolver returns stale data....363
Example: delegation problem....363
Example: UDP/TCP difference....363
Dig and Wireshark....364
Dig and TCPDump....364
Dig and MTR....364
Dig and Nmap....364
Dig and ARP Scan....365

Limitations....365
Safety notes....365
Technical role within JUMBO BLOCK Rescue....366
A.23 Whois....368
Zweck....368
Einsatzgebiet im Rescue-System....368
WHOIS ist kein DNS....369
WHOIS ist keine Geo-IP-Datenbank....369
WHOIS ist keine Personensuche....369
Grundsyntax....370
Automatische Serverauswahl....370
Domain abfragen....370
IP-Adresse abfragen....371
IPv6-Adresse....371
AS-Nummer....371
Autonomous System....371
MTR und Whois....371
Bestimmten WHOIS-Server auswählen....372
Warum einen Server ausdrücklich auswählen?....372
Port festlegen....372
WHOIS verwendet TCP....372
IANA zuerst abfragen....373
Datenschutz bei -I....373
Referral....373
Registry und Registrar....373
Thin Registry....374
Referral-Rekursion deaktivieren....374
Warum Referral-Kontrolle wichtig ist....374
Ausführliche Programmausgabe....374
Rechtliche Hinweise ausblenden....374
Registry-Hinweise können wichtig sein....375
RIPE-artige WHOIS-Dienste....375
Wichtige Konsequenz....375
Kurze IP-/Abuse-Ausgabe....375
Objektfilterung deaktivieren....375
Exakte Netzpräfixsuche....376
Weniger spezifische Objekte....376
Spezifischere Objekte....376
Warum Netz-Hierarchien relevant sind....376
Quellen auswählen....376
Serverinformationen....377
Objekttypen....377
Objektvorlagen....377
Inverse Suche....377
Kontakt-Rekursion deaktivieren....378
Referral auf RIPE-artigen Servern....378
Konfigurationsdatei....378
Aufbau von whois.conf....378
Bedeutung für Rescue-Diagnosen....379
Umgebungsvariable WHOIS_SERVER....379
Umgebungsvariable WHOIS_OPTIONS....379

Internationalisierte Domains....379
ACE / Punycode....379
Zeichenkodierung....380
WHOIS-Daten sind nicht einheitlich....380
Automatische Auswertung ist daher schwierig....380
Domain-WHOIS und Datenschutz....380
Registrant ist nicht zwingend Betreiber....381
IP-WHOIS und Netzzuordnung....381
IP-Zuordnung beweist keine Benutzeridentität....381
ASN-Zuordnung....382
Registrierung und tatsächliches Routing....382
Whois und Dig....382
Whois und MTR....382
Whois und Nmap....383
Whois und Wireshark/TCPDump....383
Whois und ARP Scan....383
Typischer Rescue-Ablauf: unbekannte öffentliche IP-Adresse....383
Typischer Rescue-Ablauf: unbekannte Domain....384
Typischer Rescue-Ablauf: MTR-Hop einordnen....384
Typischer Rescue-Ablauf: AS-Nummer....385
Private IPv4-Adressen....385
Loopback und reservierte Adressen....385
Offline-Betrieb....385
WHOIS-Server kann nicht erreichbar sein....385
Rate Limits....386
Datenschutz und Nutzungsbedingungen....386
Daten können veraltet sein....386
WHOIS und RDAP....386
Grenzen....387
Sicherheitshinweise....387
Technische Einordnung in JUMBO BLOCK Rescue....387
A.24 Chromium....389
Zweck....389
Einsatzgebiet im Rescue-System....390
Browser und Netzwerkdiagnose....390
Typische Diagnosekette....391
Direkte IP-Adresse....391
Bedeutung für die Diagnose....392
Hostname kann technisch erforderlich sein....392
HTTP....392
HTTPS....393
Zertifikate....393
Zertifikatsfehler....393
Lokale Appliances und selbstsignierte Zertifikate....393
Vor dem Umgehen einer Zertifikatswarnung....394
Systemzeit....394
Sicherheitsanzeige....394
Weiterleitungen....394
HTTP-Statuscodes....395
Beispiele....395
Developer Tools....396

DevTools öffnen....396
Network Panel....396
Was das Network Panel sichtbar macht....396
Diagnosebeispiel....397
Request Headers....397
Response Headers....398
Payload....398
Response....398
Initiator....399
Timing....399
Preserve Log....399
Cache deaktivieren....399
Hard Reload....399
Filter....400
Fetch/XHR....400
WebSocket....400
HAR....401
HAR-Dateien können hochsensibel sein....401
DevTools ist kein vollständiger Paketmitschnitt....401
Wichtige Trennung....402
Inkognito-Modus....402
Sinn im Rescue-System....402
Inkognito bedeutet nicht Anonymität....402
Downloads im Inkognito-Modus....403
Lesezeichen im Inkognito-Modus....403
Inkognito und forensische Anforderungen....403
Downloads....403
Download bedeutet nicht Vertrauenswürdigkeit....403
Downloads verifizieren....404
Download aus unbekannter Quelle....404
ClamAV als Ergänzung....404
GnuPG als Ergänzung....404
Sandbox....405
Sandbox ist keine absolute Sicherheitsgrenze....405
chromium-sandbox....405
Site Isolation....405
Defense in Depth....405
Kein Grund, Sicherheitsmechanismen abzuschalten....406
Lokale Geräte mit alten Weboberflächen....406
Sicherheitsgrenzen nicht für Kompatibilität opfern....406
Captive Portals....406
Proxy....406
Cookies....407
Cookies können sicherheitsrelevant sein....407
Local Storage und andere Site-Daten....407
Cache....407
JavaScript....408
Console....408
Vorsicht mit Console-Befehlen....408
Passwort- und Zugangsdaten....408
Rescue-System als temporäre Arbeitsumgebung....409

Typischer Rescue-Ablauf: lokale Routeroberfläche nicht erreichbar....409
Typischer Rescue-Ablauf: Hostname funktioniert nicht....410
Typischer Rescue-Ablauf: Webseite lädt teilweise....410
Typischer Rescue-Ablauf: HTTPS-Warnung bei lokaler Appliance....410
Typischer Rescue-Ablauf: Download einer Reparaturdatei....411
Chromium und Dig....411
Chromium und MTR....411
Chromium und Nmap....412
Chromium und TCPDump....412
Chromium und Wireshark....412
Chromium und Whois....412
Grenzen....413
Sicherheitshinweise....413
Technische Einordnung in JUMBO BLOCK Rescue....414
A.25 Cryptsetup....417
Zweck....417
Einsatzgebiet im Rescue-System....417
Cryptsetup und dm-crypt....418
Wichtige Ebenentrennung....418
LUKS....419
LUKS1 und LUKS2....419
LUKS erkennen....419
Ausführliche Prüfung....419
Bedeutung von isLuks....420
Keine LUKS-Erkennung....420
Header untersuchen....420
luksDump....420
luksDump entschlüsselt nicht die Nutzdaten....421
UUID....421
UUID nicht mit Dateisystem-UUID verwechseln....421
Keyslots....421
Passphrase ist nicht direkt der Datenverschlüsselungsschlüssel....421
Konsequenz....422
LUKS-Volume öffnen....422
Beispielprinzip....422
Mapping-Name....422
Passphrase....423
Falsche Passphrase....423
Read-only öffnen....423
Warum Read-only wichtig ist....423
Read-only allein genügt nicht immer für einen vollständigen Schutz....423
Geöffnetes Mapping prüfen....424
Status eines nicht aktiven Mappings....424
Nach dem Öffnen....424
Dateisystem untersuchen....424
Verschlüsselung und Dateisystemschaden unterscheiden....424
LUKS erfolgreich öffnen bedeutet nicht „Daten gesund“....425
Mapping schließen....425
Vor dem Schließen....425
Warum die Reihenfolge wichtig ist....426
Rescue-Grundablauf....426

Beschädigter Datenträger....426
Verschlüsseltes Image....427
Vorteil....427
Verschlüsselung verhindert kein Imaging....427
Header als kritische Struktur....427
LUKS1....427
LUKS2....427
Header-Backup....428
Bedeutung des Header-Backups....428
Header-Backup enthält Schlüsselzugangsdaten....428
Konsequenz....428
Header-Backup verändert die Löschstrategie....428
Header wiederherstellen....429
Höchste Vorsicht bei luksHeaderRestore....429
Rescue-Regel....429
Header testen, ohne ihn sofort zurückzuschreiben....429
Detached Header....429
Bedeutung für isLuks....430
Keyfile....430
Keyfile nicht mit Header-Backup verwechseln....430
Schlüssel hinzufügen....430
Schlüssel entfernen....430
Keyslot gezielt löschen....431
Passphrase ändern....431
Rescue-Grundsatz für Keyslots....431
luksFormat....431
Niemals zur „Reparatur“ verwenden....431
Besonders gefährlicher Denkfehler....431
erase / luksErase....432
Kein normales Datenlöschen....432
convert....432
config....432
Tokens....432
Plain dm-crypt....432
Konsequenz bei Plain dm-crypt....433
Keine sichere automatische Erkennung von Verschlüsselung allein anhand „zufälliger Daten“....433
LUKS erleichtert die Verwaltung....433
Keine Passwortwiederherstellung....433
Verschlüsselung und Datenrettung....433
Beispiel 1: LUKS intakt, Dateisystem beschädigt....434
Beispiel 2: LUKS-Header beschädigt....434
Beispiel 3: Passphrase unbekannt....434
Beispiel 4: Physisch defektes verschlüsseltes Laufwerk....434
Beispiel 5: Volume lässt sich öffnen....435
Cryptsetup und GNU ddrescue....435
Cryptsetup und SMART/NVMe Diagnose....435
Cryptsetup und GParted....435
Cryptsetup und GNOME Disks....435
Cryptsetup und GnuPG....435
Typischer sicherer Rescue-Workflow....436

Grenzen....436
Sicherheitshinweise....437
Technische Einordnung in JUMBO BLOCK Rescue....437
A.26 GnuPG....439
Zweck....439
Einsatzgebiet im Rescue-System....439
Cryptsetup und GnuPG unterscheiden....440
Digitale Signatur....440
Was eine erfolgreiche Signaturprüfung technisch aussagt....441
Signaturprüfung ist daher zweistufig....441
Detached Signature....441
Detached Signature prüfen....441
Warum beide Dateien ausdrücklich angeben?....442
Beispiel für eine ISO-Prüfung....442
„Good signature“....442
Fingerprint....442
Warum der Fingerprint wichtig ist....443
Vertrauenswürdiger Vergleich....443
Zentrale Rescue-Regel....443
Öffentliche Schlüssel anzeigen....443
Geheime Schlüssel anzeigen....444
Schlüssel importieren....444
Import ist kein Vertrauensbeweis....444
Schlüssel vor dem Import untersuchen....444
Öffentlichen Schlüssel exportieren....444
--armor....445
Binär und ASCII Armor....445
Datei signieren....445
Detached Signature erzeugen....445
ASCII-armored Detached Signature....445
Signaturschlüssel auswählen....446
Cleartext Signature....446
Cleartext Signatures und Binärdateien....446
Verifikation verändert die Datei nicht....446
GnuPG und Prüfsummen....447
Prüfsumme ohne authentisierte Quelle....447
Herstellerdownloads....447
Bereits vertrauenswürdiges GnuPG verwenden....447
Bedeutung für JUMBO BLOCK Rescue....447
Datei für einen Empfänger verschlüsseln....448
Prinzip der Public-Key-Verschlüsselung....448
Vor dem Verschlüsseln Empfängerschlüssel prüfen....448
Entschlüsseln....448
Ausgabe in Datei....449
Signierte verschlüsselte Datei....449
Symmetrische Verschlüsselung....449
Symmetrische Verschlüsselung im Rescue-Einsatz....449
Passphrase....449
Passphrase nicht in Kommandozeile schreiben....450
gpg-agent....450
Passphrase-Caching....450

Bedeutung im Rescue-System....450
Agent beenden....450
Private Schlüssel....450
Private Schlüssel nicht unnötig importieren....451
Public-Key-Verifikation ist read-only bezüglich des geprüften Artefakts....451
gpgv....451
Unterschied gpg und gpgv....451
Vertrauensmodell von gpgv....452
Konsequenz....452
gpgv mit Keyring....452
Detached Signature mit gpgv....452
Rückgabewert von gpgv....452
gpg und Exitcode....452
Signaturprüfung in Skripten....453
Schlüssel-ID versus Fingerprint....453
User IDs....453
Vertrauen....453
„WARNING: This key is not certified with a trusted signature“....454
Rescue-Workflow ohne aufgebautes Web of Trust....454
Schlüsselserver....454
Schlüsselserver lösen das Authentizitätsproblem nicht....454
Netzwerk und Offline-Betrieb....454
Offline-Verifikation....455
Beispiel: Rescue-ISO prüfen....455
Beispiel: Signatur ist ungültig....455
Beispiel: Public Key fehlt....456
Beispiel: Good Signature, unbekannter Schlüssel....456
Beispiel: Download mit Chromium....456
GnuPG und ClamAV....457
GnuPG und Hashwerte....457
GnuPG und Cryptsetup....457
GnuPG und SSH....457
Metadaten bleiben relevant....458
Sichere Löschung....458
Rescue-System und Schlüsselmaterial....458
Grenzen....458
Sicherheitshinweise....459
Technische Einordnung in JUMBO BLOCK Rescue....459
A.27 SSH / OpenSSH....462
Zweck....462
Baseline-Abgrenzung....462
Grundsyntax....463
Benutzername....463
IP-Adresse statt Hostname....463
DNS und SSH....463
Standardport....463
Anderen Port verwenden....463
SSH besteht aus mehreren Sicherheitsstufen....464
Serverauthentisierung....464
Benutzer-Authentisierung....464
Host Key und Benutzer-Key sind unterschiedliche Schlüssel....464

Erste Verbindung....465
Fingerprint prüfen....465
Warum das wichtig ist....465
known_hosts....465
Wiederholte Verbindung....465
Host Key hat sich geändert....466
Legitime Ursachen einer Host-Key-Änderung....466
Sicherheitsrelevante Ursache....466
Gespeicherten Host suchen....466
Alten Host-Key-Eintrag entfernen....466
Sicherheitsregel für ssh-keygen -R....466
StrictHostKeyChecking....467
Host-Key-Prüfung nicht deaktivieren....467
Public-Key-Authentisierung....467
Privater Schlüssel....467
Öffentlicher Schlüssel....467
Identity-Datei auswählen....468
Passphrase für private Schlüssel....468
Schlüssel ohne Passphrase....468
ssh-agent....468
ssh-add....468
Agentzustand beachten....469
Agent Forwarding....469
Agent Forwarding ist sicherheitskritisch....469
Rescue-Regel....469
Jump Host statt Agent Forwarding....469
Vorteil....470
Remote-Shell....470
Sehr wichtige Rescue-Regel....470
Remote-Kommando....470
Beispielprinzip....470
Vorteil....470
Vorsicht mit Remote-Kommandos....471
Pseudo-Terminal....471
IPv4 erzwingen....471
IPv6 erzwingen....471
Verbose-Modus....471
Was Verbose-Ausgabe hilft zu unterscheiden....472
Debugausgabe kann sensible Informationen enthalten....472
Effektive SSH-Konfiguration anzeigen....472
Warum ssh -G hilfreich ist....472
Benutzerkonfiguration....473
Beispielprinzip....473
Vorteil....473
Systemweite Konfiguration....473
Dateiübertragung mit scp....473
Moderne scp-Übertragung....473
Lokale Datei auf Remote-System kopieren....474
Remote-Datei lokal sichern....474
Verzeichnisse....474
scp verwendet SSH-Sicherheit....474

Port bei scp....474
sftp....474
SFTP ist nicht FTP über TLS....474
Rescue-Einsatz für Dateiübertragung....475
Vor dem Upload auf ein beschädigtes System....475
Port Forwarding....475
Local Forwarding....475
Beispielanwendung....476
Remote Forwarding....476
Sicherheitsrisiko bei Forwardings....476
Local Forwarding bindet standardmäßig lokal....476
Dynamisches Forwarding....476
-N....477
SSH-Tunnel ist kein VPN-Ersatz für jede Situation....477
SSH und Nmap....477
SSH und Dig....477
SSH und MTR....477
SSH und GUFW....477
SSH und TCPDump/Wireshark....477
SSH und Chromium....478
SSH und SSHFS....478
Möglicherweise kompromittiertes Zielsystem....478
Keine privaten Schlüssel auf das Ziel kopieren....478
Kein Agent Forwarding zum kompromittierten Ziel....478
Remote-Ausgaben sind nicht automatisch vertrauenswürdig....479
Wichtige Unterscheidung....479
Typischer Rescue-Ablauf: Server erreichbar, SSH funktioniert nicht....479
Typischer Rescue-Ablauf: neuer Server....480
Typischer Rescue-Ablauf: Host-Key-Warnung....480
Typischer Rescue-Ablauf: Datei vom Server sichern....480
Typischer Rescue-Ablauf: internes Webinterface....480
Typischer Rescue-Ablauf: kompromittierungsverdächtiger Server....480
Grenzen....481
Sicherheitshinweise....481
Technische Einordnung in JUMBO BLOCK Rescue....482
A.28 SSHFS....484
Purpose....484
Use in the rescue system....484
SSHFS builds on SSH....484
SSHFS is not a separate server service....485
SFTP instead of remote shell....485
Basic syntax....485
Example....485
Local Mount Point....485
Omit username....486
Omit remote directory....486
Entire root file system....486
Remote permissions remain decisive....486
Important feature of local laws....486
default_permissions....487
By default, only for the mounting user....487

allow_other....487
Basic rule....487
Host key verification....487
Host key warning....487
Using SSH options....488
Different SSH port....488
Use a specific private key....488
Read-only mount....488
Significance for JUMBO BLOCK Rescue....489
Read-only protects only this access path....489
No forensic snapshot....489
Live system stays live....489
Consequences for securing evidence....490
Copy files....490
Important perspective....490
Writing feels remote....490
Rescue basic rule....490
Graphical file managers....490
Danger from graphical programs....491
Applying programs to remote files....491
The network remains part of file access....491
performance....491
Connection interrupted....492
Caused....492
ServerAliveInterval....492
Example....492
reconnect....492
combination....492
Important limitation of reconnect....492
No protection against data loss during a shutdown....492
Importance for rescue copies....493
Hash check....493
delay_connect....493
sshfs_sync....493
no_readahead....493
sync_readdir....493
Caching....494
SSHFS is not a cluster file system....494
Remote file is modified during reading....494
Database files....494
Unmount....494
Baseline and fusermount3....495
Example....495
Before the unmount....495
"Device or resource busy"....495
Connection interruption and blocked processes....495
consequence....495
Read-only Rescue Workflow....495
Workflow with an unstable network connection....496
Workflow: Backing up logs from a remote server....496
Workflow: Remote backup destination....496

Sensitive Rescue Data....497
Transport encryption is not storage encryption....497
Combination with GnuPG....497
SSHFS and SCP....497
When SCP is easier....498
When SSHFS is useful....498
SSHFS and SFTP....498
SSHFS and Midnight Commander....498
SSHFS and ClamAV....498
SSHFS and hash calculation....498
SSHFS and network diagnostics....499
Diagnostic principle....499
IPv6 address....499
/etc/fstab....499
Rescue classification of /etc/fstab....500
Version 3.7.3....500
Version boundary....500
Limitations....500
Safety instructions....500
Role in JUMBO BLOCK Rescue....501
A.29 HWInfo....503
Purpose....503
Area of application in the rescue system....503
Hardware detection instead of health diagnosis....504
Basic call....504
Scope of the output....505
Hardware classes....505
Combining multiple classes....505
Brief view....506
Quick system overview....506
CPU....506
RAM....506
Important distinction regarding RAM....506
Data carrier....507
Examine individual data carriers....507
Significance in a rescue situation....507
Never simply copy device names from a previous session....507
Block devices....508
Partitions....508
Storage controller....508
SCSI....508
RAID devices....508
Meaning....509
PCI devices....509
USB devices....509
USB controller....509
Network hardware....510
Network and Netcard....510
Wi-Fi....510
Hardware detected, but network still doesn't work....510
graphics card....511

monitor....511
BIOS....511
Sound....511
camera....511
Bluetooth....512
Keyboard and mouse....512
Printers and scanners....512
--all....512
--reallyall....512
Log output....512
Using a hardware log....513
Data protection of the log....513
Save configuration....513
View saved configuration....513
Compare device assignments....514
Importance for a live rescue system....514
Debug output....514
--verbose....514
Hardware database....515
Output database....515
View version....515
Root privileges....515
Do not use root access indiscriminately....515
Detection depends on the running kernel....516
"Not detected by HWInfo" does not necessarily mean "not present"....516
Possible causes of lack of detection....516
HWInfo does not modify partitions....516
HWInfo does not repair data carriers....516
Typical rescue scenario: unknown computer....517
Typical rescue case: Hard drive missing....517
Typical rescue scenario: USB drive is not recognized....517
Typical rescue scenario: Network not working....518
Typical rescue case: incorrect screen resolution....518
Typical Rescue case: Support report....518
HWInfo and Inxi....518
HWInfo and LSHW....519
HWInfo and dmidecode....519
HWInfo and lm-sensors....519
HWInfo and SMART....519
HWInfo and NVMe diagnostics....519
Hardware detection before repair....519
Don't guess the device name....520
Especially critical with ddrescue....520
Hardware report before expansion....520
Live system as an advantage....520
Live system as a boundary....520
Limitations....521
Safety instructions....521
Technical classification in JUMBO BLOCK Rescue....521
A.30 Inxi....524
Purpose....524

Area of application in the rescue system....524
Difference to HWInfo....525
Basic call....525
Base report....525
Importance for Rescue....526
Extended Report....526
Levels of detail with -v....526
-v 0....527
-v 1....527
-v 2....527
Higher levels of detail....527
Rescue Rule....528
CPU....528
CPU flags....528
CPU detection is not a CPU test....528
machine....528
DMI demarcation....529
RAM....529
RAM detection is not a RAM test....529
graphic....529
Rescue operation graphic....530
Audio....530
Network devices....530
Extended network information....530
IP addresses....530
Data protection....531
-z – filter sensitive information....531
Example....531
Central Support Rule....531
-Z....532
Data carrier....532
Data carrier information is not a complete condition diagnosis....532
Partitions....532
RAID....532
RAID output is not RAID repair....532
USB....533
Importance for Rescue....533
Bluetooth....533
battery....533
Rescue rule for notebooks....534
Sensors....534
Distinction from A.33....534
Additional details with -x....534
Example principle....534
More details may contain sensitive data....535
-a – administrative details....535
Meaning....535
Do not automatically use -a....535
Driver information....535
Hardware detected, but incorrect driver....536
Kernel....536

A very important distinction....536
Example....536
The same applies to drivers....536
Desktop environment....536
Operating system information....537
Rescue basic rule....537
Version without colors....537
Benefits for saved reports....537
Redirect output....537
Filtered support report....537
Do not publish report blindly....537
Output for internal documentation....538
Typical rescue process: initial system overview....538
Typical rescue scenario: Network not working....539
Typical rescue case: Graphics problem....539
Typical rescue case: Storage....540
Typical rescue scenario: Hardware report for support....540
Typical rescue case: Notebook....541
Inxi and HWInfo....541
Inxi and LSHW....541
Inxi and dmidecode....541
Inxi and lm-sensors....541
Inxi and SMART/NVMe....542
External utilities....542
Root privileges....542
Rescue Rule....542
Limitations....543
Safety instructions....543
Technical classification in JUMBO BLOCK Rescue....543
A.31 LSHW....546
Purpose....546
Area of application in the rescue system....546
Difference to HWInfo and Inxi....547
Basic call....547
Root privileges....547
Rescue Rule....547
Hierarchical representation....547
Importance in rescue operations....548
Brief view....548
Typical entry point....548
Hardware path....548
Bus information....549
Meaning....549
-short and -businfo....549
Hardware classes....549
Determine classes....550
Combining multiple classes....550
Storage diagnostics....550
Network hardware....550
Network report as HTML....550
Configuring a network device....551

Hardware detection does not mean network is working....551
CPU....551
CPU detection is not a stress test....551
Cache....551
RAM....552
SPD recognition....552
RAM inventory is not a RAM test....552
Firmware and motherboard....552
DMI....552
Differentiation from dmidecode....553
Supported information sources....553
Linux system information....553
Rescue perspective....553
PCI and USB....553
Numerical device identifiers....554
Using numeric IDs....554
Example principle....554
Data carrier....554
Storage controller....554
combination....554
Detected data carrier does not mean data carrier is healthy....554
Condition diagnosis....555
Hardware tree and partitions....555
HTML output....555
XML output....555
JSON output....555
Choose output format deliberately....556
Save report to file....556
HTML report....556
JSON report....556
Sensitive information....556
-sanitize....557
Support report....557
Example principle....557
Check the filtering anyway....557
-notime....557
Benefits for comparative reports....557
-quiet....558
Enable or disable hardware tests....558
Documented tests....558
Meaning of "test"....559
Example....559
Why disable information sources?....559
SQLite dump....559
Meaning....559
Version query....560
Help....560
Graphical user interface....560
Baseline demarcation from the GUI....560
Typical rescue scenario: unknown computer....560
Typical rescue scenario: Data carrier is missing....561

Typical rescue case: Network adapter not working....561
Typical Rescue case: USB device unknown....562
Typical rescue case: Hardware documentation....562
Typical Rescue case: machine comparison....562
HWInfo, Inxi and LSHW together....562
Divergent results....563
Limitations of hardware detection....563
Key conclusion....563
Hardware health....563
Writing effect....564
Nevertheless, use with caution....564
Live system perspective....564
Installed target system....564
Safety instructions....564
Technical classification in JUMBO BLOCK Rescue....565
A.32 DMI Decode....567
Purpose....567
Area of application in the rescue system....567
Key difference to HWInfo and LSHW....568
Basic rule....568
Basic call....568
Root privileges....568
Data source under Linux....569
Direct memory access....569
Structure of a DMI entry....569
DMI types....569
Targeted query with --type....570
Supported type keywords....570
BIOS....571
system....571
Mainboard....571
Housing....571
processor....571
memory....571
Physical Memory Array....571
Memory Device....572
Empty memory banks....572
Important: Maximum capacity is not installed storage....572
RAM data is not a memory test....573
Individual values with --string....573
System manufacturer....573
System model....573
System version....573
Serial number....573
System UUID....573
BIOS manufacturers....573
BIOS version....573
BIOS date....573
Motherboard manufacturer....574
Mainboard model....574
Motherboard serial number....574

Case information....574
Processor information....574
Multiple processors....574
Show available string keys....574
Show available types....575
Show specific handle....575
Using the handles....575
Quiet mode....575
Rescue operation of --quiet....576
Raw data as hex dump....576
Binary DMI fuse....576
Important file rule....576
Offline evaluation....576
Benefits for Rescue documentation....577
A DMI dump is not a disk image....577
OEM Strings....577
OEM data....577
--no-quirks....577
Rescue classification of --no-quirks....578
--no-sysfs....578
Firmware quality is crucial....578
Typical placeholders....578
Firmware specifications and physical labeling may differ....578
Serial numbers....579
System UUID....579
Asset Tags....579
Data protection....579
Better to ask targeted questions....579
BIOS/Firmware diagnostics....579
BIOS version is not a firmware integrity check....580
Mainboard diagnostics....580
System identification....580
Please respect data protection....580
RAM inventory....580
Typical rescue case: RAM upgrade....580
Reason....581
Typical rescue scenario: unknown computer....581
Typical rescue scenario: Preparing for a BIOS update....581
Critical safety notice....582
Typical rescue scenario: Document hardware before replacement....582
After hardware change....582
Typical rescue case: System report....582
Typical rescue scenario: only BIOS version required....582
Typical rescue scenario: only the system model is needed....583
Typical rescue case: Memory problem....583
dmidecode and HWInfo....583
dmidecode and Inxi....583
dmidecode and LSHW....583
Contradictions between tools....584
Physical reality prior to firmware specification....584
SMBIOS support from version 3.6....584

New options in 3.6....584
Newer SMBIOS versions than dmidecode are supported....584
Rescue consequence....585
Limitations....585
Safety instructions....585
Technical classification in JUMBO BLOCK Rescue....585
A.33 Sensors / lm-sensors....588
Purpose....588
Area of application in the rescue system....588
Central demarcation....589
Three separate levels....589
Basic call....589
Typical issue....589
Do not expect a fixed sensor list....590
Temperatures....590
Celsius for technical documentation....590
Chip selection....590
Adapter information....590
To use....591
Raw data....591
Difference from normal output....591
JSON output....591
Benefits of structured output....591
version....592
Help....592
Configuration files....592
Meaning of the configuration....592
Alternative configuration file....593
Temporarily bypass configuration....593
Rescue meaning....593
Sensor limits....593
Limit value does not automatically constitute manufacturer approval....593
Plausibility check....594
Incorrect or unconnected sensor inputs....594
Fan speeds....594
Zero value for the fan....594
tensions....595
Power and current values....595
CPU sensors....595
Multiple CPU cores....595
Sensor values are snapshots....596
Repeated observation....596
No artificial load required....596
sensors detect....596
What sensors-detectinvestigates....596
Interactive operation....597
Automatic operation....597
Important safety notice regarding --auto....597
Fundamental risk of sensors-detect....597
Manufacturer warning....597
Consequences for JUMBO BLOCK Rescue....597

Recommended Rescue Procedure....598
Production systems....598
Rescue principle....598
Kernel modules....598
Live system perspective....598
Missing sensor value....598
A missing value is not proof of a missing sensor....599
sensors -s....599
Important distinction....599
Rescue Rule....599
Configuration instructions....599
label....600
compute....600
ignore....600
set....600
consequence....600
Temperature sensor types....600
Firmware configuration....600
Hysteresis....600
Sensor alarm....601
An alarm does not automatically mean total loss....601
--bus-list....601
Bus numbers....601
Typical rescue scenario: Computer shuts down unexpectedly....601
Typical rescue case: Fan constantly running at high speed....602
Typical rescue scenario: fan is stopped....602
Typical rescue scenario: unusually high CPU temperature....603
Typical Rescue case: obviously incorrect voltage....603
Typical rescue scenario: no sensors found....603
Typical rescue scenario: thermal control before rescue....603
Data carrier temperature....604
Sensor value and SMART temperature....604
lm-sensors and HWInfo....604
lm-sensors and Inxi....604
lm-sensors and LSHW....604
lm-sensors and dmidecode....605
Four different hardware perspectives....605
Writing utilities....605
Security warning regarding isaset....605
isadump....606
JUMBO BLOCK Rescue Principle....606
No automatic write operations....606
Limitations....606
Safety instructions....607
Technical classification in JUMBO BLOCK Rescue....607
A.34 Midnight Commander....609
Purpose....609
Area of application in the rescue system....609
Not data recovery software in the strict sense....610
Two-panel principle....610
Program start....610

Start with a directory....610
Two starting directories....610
Rescue benefits....611
navigation....611
Switch between panels....611
menu....611
Function keys....611
Rescue Advantage....612
Internal viewer....612
Rescue principle....612
Start viewer directly....612
Typical rescue applications of the viewer....612
Internal Editor....613
Start the editor directly....613
Editing means writing....613
Better on a working copy....613
Copy....613
Typical Rescue case....614
Copying does not automatically mean data recovery....614
Why?....614
Move....614
Key Rescue Difference....614
Rename....615
Create directory....615
Delete....615
Critical safety notice....615
No wastebasket as a rescue strategy....615
Select multiple files....615
Pattern-based selection....615
Rescue caution when selecting large numbers....616
File search....616
Content search....616
Rescue benefits....616
Search is not file carving....616
Shell command line....617
Importance for Rescue....617
Caution....617
Subshell....617
Rescue meaning....617
Archives and virtual file system....617
Archive....618
Do not confuse with a real mount....618
Do not unnecessarily modify the archive....618
File types and actions....618
Rescue Rule....618
Executable files....618
Security rule for suspicious files....618
Hidden files....619
File permissions and metadata....619
Rescue meaning....619
A copy is not automatically forensically identical....619

Forensic evidence preservation....620
Read-only source....620
Important limitation....620
Read-only and backup target....620
Check device identity first....620
Don't blindly trust path names....621
Free space....621
Internal editor mcedit....621
Syntax highlighting....621
configuration....621
Rescue significance of user configuration....621
Mouse support....622
black and white representation....622
Color representation....622
View version....622
Help....622
Typical rescue scenario: Backing up user files....623
Typical rescue scenario: Examine the configuration file....623
Typical rescue scenario: Repair configuration....623
Typical rescue scenario: sorting rescued files....623
Important limitation....624
Typical rescue scenario: Back up logs....624
Typical Rescue case: Remote files....624
Security limit....624
Midnight Commander and SSH....624
Midnight Commander and GNU ddrescue....625
Midnight Commander and TestDisk....625
Midnight Commander and PhotoRec/Foremost....625
Midnight Commander and Extundelete....625
Midnight Commander and GParted....626
Midnight Commander and GNOME Disks....626
Midnight Commander and ClamAV....626
Midnight Commander and Binwalk / Hexedit....626
Limitations....626
Safety instructions....627
Technical classification in JUMBO BLOCK Rescue....627
A.35 QPDF....629
Purpose....629
Area of application in the rescue system....629
Check PDF....630
Correctly assessing warnings....630
Rewrite damaged PDF....630
Original preserved....630
Limits of repair....631
Check page number....631
Select and compile pages....631
Split PDF....631
Encrypted PDFs....632
Decode....632
No password cracker....632
Embedded files....632

Safety rule for attachments....632
Digal signatures....633
Typical Rescue case: PDF cannot be opened....633
Typical rescue scenario: PDF from PhotoRec or Foremost....633
Typical rescue scenario: saving individual pages....633
Typical rescue scenario: encrypted PDF....634
QPDF and other rescue tools....634
Limitations....635
Safety instructions....635
Technical classification in JUMBO BLOCK Rescue....635
Block B....637
Tool groups....637
Assignment of areas, programs and versions....638
B.1 MX Boot Repair....639
Purpose....639
Significance for JUMBO BLOCK Rescue....639
Repairing the GRUB configuration....639
Reinstalling the GRUB boot loader....640
UEFI systems....640
Legacy/MBR systems....640
Regenerating initramfs....640
Typical rescue workflow....641
Dual-boot and multi-boot systems....641
Limitations....641
Safety notes....642
Interaction with other rescue tools....642
Technical classification in JUMBO BLOCK Rescue....643
B.2 UEFI Manager....644
Purpose....644
Area of use in the rescue system....644
Prerequisite....644
Displaying existing boot entries....644
Changing the boot order....645
Boot Next....645
Enabling and disabling entries....645
Adding and removing entries....645
Typical rescue scenario: wrong operating system starts....645
Typical rescue scenario: dual boot....646
EFI Stub Installer....646
Frugal EFI Stub Installer....646
Limitations....646
Safety notes....646
Technical classification in JUMBO BLOCK Rescue....647
B.3 Chroot Rescue Scan....648
Purpose....648
Significance for JUMBO BLOCK Rescue....648
What is a chroot environment?....648
Automatic search....648
Several Linux installations....649
Preparing the chroot environment....649
Shell in the target system....649

Typical rescue tasks....650
Network access....650
Damaged initrd....650
Typical rescue workflow....651
Difference from MX Boot Repair....651
Not a second running operating system....651
Limitations with regard to other operating systems....652
Limitations with damaged file systems....652
Safety notes....652
Technical classification in JUMBO BLOCK Rescue....653
B.4 MX Live USB Maker....654
Purpose....654
Significance for JUMBO BLOCK Rescue....654
Basic workflow....654
Full-featured mode....654
Image mode....655
Cloning a live system....655
Dry Run....655
Encrypted live USB....655
Advanced options....656
Typical use....656
Limitations....656
Safety notes....657
Technical classification in JUMBO BLOCK Rescue....657
B.5 MX Snapshot....658
Purpose....658
Area of use in the rescue system....658
Special considerations for compromised systems....659
System snapshot....659
Personal snapshot....659
Exclusions....659
Typical rescue workflow....660
No automatic decontamination....660
Further use....660
Limitations....661
Safety notes....661
Technical classification in JUMBO BLOCK Rescue....661
B.6 ISO Signature Verifier....663
Purpose....663
Significance for JUMBO BLOCK Rescue....663
Operating-system-independent checking....663
Checksum and signature....664
Different signature procedures....664
Basic workflow....665
Signature keys....665
A valid signature does not automatically mean a trustworthy publisher....665
Typical rescue workflow....665
Failed verification....666
Successful verification....666
No malware check....666
Interaction with ClamAV....667

Interaction with MX Live USB Maker....667
Own checksums....667
Limitations....667
Safety notes....668
Technical classification in JUMBO BLOCK Rescue....668
Block C....669
Tool groups....669
Assignment of areas, programs and versions....670
C.1 – Format USB....671
Purpose....671
Use in the rescue system....671
Functions....671
Operation and workflow....672
Choice of file system....672
Typical rescue scenarios....672
Providing a storage device for rescued files....672
Handover of examination results....673
Data exchange between different operating systems....673
Limitations and safety notes....673
Technical classification within JUMBO BLOCK Rescue....673
C.2 – Thunar File Manager....675
Purpose....675
Use in the rescue system....675
Functions....675
Operation and workflow....676
Typical rescue scenarios....676
Taking over rescued files....676
Backing up accessible user data....676
Compiling examination results....676
Data exchange with removable media....677
Limitations and safety notes....677
Technical classification within JUMBO BLOCK Rescue....677
C.3 – Catfish File Search....679
Purpose....679
Use in the rescue system....679
Functions....679
Operation and workflow....679
Typical rescue scenarios....680
Searching for user documents....680
Searching rescued data....680
Locating configuration and log files....680
Preparing a data handover....680
Limitations and safety notes....680
Technical classification within JUMBO BLOCK Rescue....681
C.4 – Engrampa Archive Manager....681
Purpose....681
Use in the rescue system....682
Functions....682
Operation and workflow....682
Typical rescue scenarios....683
Opening a rescued archive....683

Extracting required files....683
Compiling examination results....683
Preparing a data handover....683
Limitations and safety notes....683
Technical classification within JUMBO BLOCK Rescue....684
C.5 – FeatherPad....685
Purpose....685
Use in the rescue system....685
Functions....685
Operation and workflow....686
Typical rescue scenarios....686
Checking configuration files....686
Targeted configuration change....686
Evaluating log files....686
Examining rescued text files....687
Documenting a rescue deployment....687
Limitations and safety notes....687
Technical classification within JUMBO BLOCK Rescue....687
C.6 – GtkHash....689
Purpose....689
Use in the rescue system....689
Functions....689
Operation and workflow....690
Typical rescue scenarios....690
Checking a file transfer....690
Documenting rescued files....690
Checking against a published checksum....690
Checking rescued data....691
Limitations and safety notes....691
Technical classification within JUMBO BLOCK Rescue....691
C.7 – Advanced Network Configuration....693
Purpose....693
Use in the rescue system....693
Functions....693
Operation and workflow....694
Typical rescue scenarios....694
Manual network configuration....694
Preparing an SSH connection....694
Preparing SSHFS....694
Network diagnostics....695
Working in separate network segments....695
Limitations and safety notes....695
Technical classification within JUMBO BLOCK Rescue....695
C.8 – Xfce Terminal....697
Purpose....697
Use in the rescue system....697
Functions....697
Operation and workflow....698
Typical rescue scenarios....698
Starting command-line tools....698
Checking storage devices and file systems....699

Remote work....699
Evaluating technical information....699
Working with administrative rights....699
Limitations and safety notes....699
Technical classification within JUMBO BLOCK Rescue....700
C.9 – Screenshot....701
Purpose....701
Use in the rescue system....701
Functions....701
Operation and workflow....702
Typical rescue scenarios....702
Documenting an error message....702
Documenting the state of a storage device....702
Documenting settings....702
Documenting analysis results....702
Handover of a rescue process....703
Limitations and safety notes....703
Technical classification within JUMBO BLOCK Rescue....703
Block D....705
Program groups....705
Assignment of areas, programs and versions....707
D.1 – Hydro....709
Purpose....709
Role within the Engineering Suite....709
Essential functions....709
Data and data exchange....710
Local operation and significance for JUMBO BLOCK Rescue....711
Data backup....711
Reports....711
Limitations and notes....712
Technical classification in JUMBO BLOCK Rescue....712
Further documentation....713
D.2 – Plan....714
Purpose....714
Role within the Engineering Suite....714
Essential functions....714
Planning via target volume....715
Planning via installation size....715
Manual adjustment....715
Installation layout....715
SABA sealing....715
Data and data exchange....715
Data integrity....716
Local operation and significance for JUMBO BLOCK Rescue....716
Data backup....716
Reports....717
Limitations and notes....717
Technical classification in JUMBO BLOCK Rescue....717
Further documentation....718
D.3 – Budget....719
Purpose....719

Role within the Engineering Suite....719
Essential functions....720
Data and data exchange....720
Import from Plan....720
Import from Static....720
Costing principle....720
Local operation and significance for JUMBO BLOCK Rescue....721
Reports....721
Limitations and notes....721
Technical classification in JUMBO BLOCK Rescue....722
Further documentation....722
D.4 – Static....723
Status within JUMBO BLOCK Rescue....723
Purpose of the engineering service....723
Procedure....723
1. Plan....724
2. Hand over project data....724
3. Receive results....724
Scope of the engineering service....724
Cover plate....724
Floor and side plates....724
Overall system....724
Stability....725
Durability and robustness....725
Reinforcement and optimisation....725
Required project data....725
Results and reports....726
Structural Report....726
Production Printout....726
Customer Report....726
Limitations....726
Significance within JUMBO BLOCK Rescue....727
Further information....727
D.5 – Temp....728
Status within JUMBO BLOCK Rescue....728
Purpose of the engineering service....728
Procedure....729
1. Plan....729
2. Hand over project data....729
3. Receive results....729
Scope of the engineering service....729
Heat storage....729
Heat dissipation....729
Heat inputs....729
Medium temperatures and operation....730
Site effect....730
Site TIB....730
Required project data....730
Results and reports....731
TIB report....731
Detailed balance....731

Climate profiles....731
Infrastructure scenarios....731
Limitations....731
Significance within JUMBO BLOCK Rescue....732
Further information....732
D.6 – Inspect....733
Purpose....733
Role within the Engineering Suite....733
Essential functions....734
Base data from project planning....734
Asset tree....734
Design values and limits....735
Check catalogue....735
Due dates....735
Inspection reasons....736
Carrying out an inspection....736
Photo documentation....736
Defect management....737
Data and data exchange....737
Input data....737
Inspection data....737
Data integrity....738
File hash....738
Chained inspection logs....738
Photos....738
Local operation and significance for JUMBO BLOCK Rescue....738
Data backup....739
Reports....739
Inspection report....739
Maintenance and inspection plan....740
Defect list....740
Limitations and notes....740
Technical classification in JUMBO BLOCK Rescue....740
Further documentation....741
D.7 – Project....742
Purpose....742
Role within the Engineering Suite....742
Essential functions....743
Project master data....743
Project sections and project steps....743
Responsible and Executor....743
Status and dates....744
Dates overview....744
Automatic timeline....744
Partners and roles....744
Document register and file attachments....745
Status log....745
Data and data exchange....746
Additive import and merging....746
Conflict resolution....747
Local operation and significance for JUMBO BLOCK Rescue....747

Data backup....748
Reports....748
Project report....748
Gantt report....748
Data integrity....748
Limitations and notes....749
Technical classification in JUMBO BLOCK Rescue....749
Further documentation....750
D.8 – Tender....751
Purpose....751
Role within the Engineering Suite....751
Essential functions....752
Opening GAEB X files....752
New bill of quantities....752
BoQ structure....753
Items....753
Summary items and subitems....753
Search and navigation....754
Prices and totals....754
Data phases and phase changes....755
X80....755
X82....755
X82 → X83....755
X84....755
X84 bid submission....755
Hidden migration aid for D81–D84....756
Local operation and significance for JUMBO BLOCK Rescue....757
Data backup....757
Printing and PDF....757
Language and number format....758
Limitations and notes....758
Technical classification in JUMBO BLOCK Rescue....759
Further documentation....759
D.9 – Calc....760
Purpose....760
Role within the Engineering Suite....760
Essential functions....761
Basic calculator....761
Scientific functions....761
Memory functions....762
Percentage calculation....762
Geometry calculator....762
JUMBO BLOCK Area Fit....763
Unit conversion....763
JUMBO BLOCK Module Calculator....764
Calculation history....764
Data and data exchange....764
Local operation and significance for JUMBO BLOCK Rescue....765
Data backup....765
PDF report....766
Language and operation....766

Limitations and notes....766
Baseline version note....767
Technical classification in JUMBO BLOCK Rescue....767
Further documentation....767
D.10 – Units....768
Purpose....768
Role within the Engineering Suite....768
Essential functions....768
Categories....768
Mechanics & Physics....768
Electrical....769
Thermodynamics....769
Other....769
Performing a conversion....770
Number formatting....770
Special case: temperature....770
Special case: fuel consumption....771
Calculation history....771
Data and data exchange....771
Import....772
Local operation and significance for JUMBO BLOCK Rescue....772
Data backup....773
PDF report....773
Language....773
Limitations and notes....774
Technical classification in JUMBO BLOCK Rescue....774
Further documentation....774
D.11 – Note....776
Purpose....776
Role within the Engineering Suite....776
Key functions....776
Create and edit notes....776
Notes list....777
Search....777
Save....777
Automatic saving....777
Links and files....778
Left....778
files....778
Data and data exchange....779
import....779
Local working methods and significance for JUMBO BLOCK Rescue....779
Data backup....780
Print and PDF....780
Language....781
Limits and Notes....781
Technical classification in JUMBO BLOCK Rescue....782
Further documentation....782
D.12 – Tables....783
Purpose....783
Role within the Engineering Suite....783

Workbook and worksheets....784
Formulas and calculations....785
Formula functions....785
Error detection....787
Editing and clipboard....787
Undo and Redo....788
Rows and columns....788
Sort, search and replace....789
Cell formatting....789
Conditional formatting....790
Freezing rows and columns....790
Diagrams....790
Data and data exchange....791
Native .jbtab format....791
XLSX....792
CSV....793
Data integrity....793
Local working methods and significance for JUMBO BLOCK Rescue....793
Data backup....794
Print and PDF....794
Language....795
Limits and Notes....796
Technical classification in JUMBO BLOCK Rescue....796
Further documentation....796
D.13 – Writer....798
Purpose....798
Role within the Engineering Suite....798
Document structure....799
letterhead....799
Main text....799
Footfield....799
Side foot....799
Text formatting....800
fonts....801
Sans-serif....801
Serifs....801
Monospace....801
Headings and table of contents....801
Automatic fields....802
tables....802
Pictures....802
Links and files....803
Page settings....803
Templates....804
business letter....804
File note....804
protocol....805
Search and Replace....805
Word and character count....805
Data and data exchange....806
Import and security sanitization....806

Word export....807
Word Import....808
ODT Import....809
Local working methods and significance for JUMBO BLOCK Rescue....809
Data backup....809
PDF export....810
Data integrity....810
View and operation....811
Language....811
Limits and Notes....812
Technical classification in JUMBO BLOCK Rescue....812
Further documentation....813
D.14 – Reader....814
Purpose....814
Role within the Engineering Suite....814
Supported file formats....815
EPUB....815
PDF....815
Markdown....815
text....816
HTML/HTM....816
JSON and XML....816
Open document....816
Document-related settings....816
navigation....817
EPUB....817
PDF....817
Text formats....817
Table of contents and sidebar....817
EPUB....817
PDF....817
Internal and external links....817
Depiction....818
Reading width....818
Light and dark representation....818
Focus mode....818
Full screen mode....819
Search....819
PDF....819
EPUB and text formats....819
Limits of PDF search....820
bookmark....820
PDF....820
EPUB....820
Text formats....820
Markings....820
Keyboard operation....821
Touch operation....821
Data and data exchange....822
Local storage....822
Data protection....823

Significance for JUMBO BLOCK Rescue....823
Limits and Notes....823
Technical classification in JUMBO BLOCK Rescue....824
Further documentation....824
D.15 – Text....825
Purpose....825
Role within the Engineering Suite....825
Key functions....826
Open files....826
New file....826
Edit....826
Syntax highlighting....826
Line numbers....827
Search and Replace....827
Undo and Redo....827
Data and file formats....828
Local working methods and significance for JUMBO BLOCK Rescue....828
Data backup....829
Language....829
Limits and Notes....829
Technical classification in JUMBO BLOCK Rescue....830
Further documentation....830
D.16 – PDF Merge....831
Purpose....831
Role within the Engineering Suite....831
Add PDFs....831
File selection....831
Drag-and-drop....832
Loading the PDF files....832
Shared page list....832
Arrange pages....833
Move one position....833
Free movement....833
Remove pages....833
Save....833
Storage location and browser differences....834
Chrome and Edge....834
Firefox and Safari....834
Data integrity....835
Reset....835
Local working methods and significance for JUMBO BLOCK Rescue....835
Meeting-based working methods....836
Language....836
Limits and Notes....836
No page preview....836
No editing of page content....837
No encrypted PDF files....837
No continuous session....837
Browser-dependent storage location....837
Technical classification in JUMBO BLOCK Rescue....837
Further documentation....838

D.17 – PDF Printer....839
Purpose....839
Role within the Engineering Suite....839
Open PDF....840
Interactive preview....840
Zoom and page resizing....840
Zoom in and out....840
100%....840
Width....840
Page....841
Continuous zoom....841
Paper format....841
Page orientation....841
Print quality....841
Draft....841
Good....841
High....842
Pages per sheet – N-up....842
Brochure printing....842
Duplex note....843
Print output....843
Keyboard operation....843
Page navigation....843
zoom....843
Adjust to width....844
100%....844
Press....844
Data and data exchange....844
Local working methods and significance for JUMBO BLOCK Rescue....844
Importance of the system printer....845
Data protection....845
Technical basis....845
Limits and Notes....846
Technical classification in JUMBO BLOCK Rescue....846
Further documentation....847
D.18 – Convert....848
Purpose....848
Role within the Engineering Suite....848
operation....849
1. Select PDF....849
2. Convert....849
3. Save the Word file....849
Status display....849
How the conversion works....850
Text extraction....850
Header and footer recognition....850
Sales reconstruction....850
Table recognition....851
Pictures....851
Font assignment....851
Data and data exchange....851

Choose storage location....852
Chrome and Edge....852
Firefox and Safari....852
Local working methods and significance for JUMBO BLOCK Rescue....853
No continuous work session....853
Language....853
Limits and Notes....854
No pixel-perfect 1:1 conversion....854
Multi-column layouts....854
Short tables....854
Headers and footers....854
vector graphics....854
No OCR....855
Technical classification in JUMBO BLOCK Rescue....855
Further documentation....855
D.19 – Archive....856
Purpose....856
Role within the Engineering Suite....856
Supported formats....857
Technical basis of the archive formats....857
Packing files and folders....857
File list....858
Archive name....858
compression....858
File names and timestamps....859
ZIP64....859
Password protection with AES-256....860
Compatibility of encrypted ZIP files....860
Visible metadata in ZIP encryption....860
Handling passwords....861
Open and check the archive....861
Unpack everything....861
Chrome and Edge....861
Firefox and Safari....862
Target folder and browser protection....862
Extract individual files....862
Convert archive to ZIP....863
Integrity check during unpacking....863
Safe pathway treatment....863
Open password-protected archives....864
Limits of encrypted 7z and RAR archives....864
Progress and Cancel....864
Local working methods and significance for JUMBO BLOCK Rescue....865
Importance for recovered data....865
RAM and archive size....866
ZIP, TAR and GZIP....866
7z, RAR, TAR.XZ and TAR.BZ2....866
Change data during unpacking....866
Unsupported archive variants....866
Rare ZIP compression procedures....867
Mobile use....867

Language....867
Limits and safety information....867
Technical classification in JUMBO BLOCK Rescue....868
Further documentation....868
D.20 – Files....870
Purpose....870
Role within the Engineering Suite....870
Browser and access rights....871
Chrome, Edge and Opera on the computer....871
Firefox, Safari and most smartphones....871
Reading and writing / Reading only....872
Protected areas....872
Open folder....872
Recently opened folders....873
navigation....874
File list....874
List and tile view....874
Select files and folders....875
Open files and folders....875
File preview....875
Status bar and folder size....876
Create a new folder....876
Create new text file....876
Rename....876
Copy....877
Move....877
Copying between storage media....878
Name conflicts....878
Protection against recursive copying....878
Drag-and-drop from the outside....879
Progress and Cancel....879
Delete....879
Seek....880
Search by name....880
Search by type....880
Search within text content....881
Search results....881
Download files....881
Booting from USB stick....882
Local working methods and data protection....882
Significance for JUMBO BLOCK Rescue....883
File timestamps and attributes....883
Move....884
Performance limits....884
Locked files....884
Language and presentation....884
Limits and safety information....885
Important keyboard shortcuts....885
Technical classification in JUMBO BLOCK Rescue....886
Further documentation....886
The journey continues....887

# Block A

**Block A – JB-Rescue** forms the functional core of JUMBO BLOCK Rescue.

The tools it contains are used to examine, diagnose, back up and restore data and systems. They cover a range of rescue scenarios – from damaged storage devices and lost files through network and security analyses to hardware diagnostics and the technical editing of rescued files.

The tools are not restricted to a particular operating system family. Whether they can be used in a specific case depends on the program, file system, storage device, protocol or target system concerned.

The basic rule is:

- **Identify and diagnose first.**
- **Keep originals and damaged sources as unchanged as possible.**
- **With unstable storage devices, back up first.**
- **Carry out analyses and repairs preferably on images or working copies.**
- **Only carry out writing and destructive operations deliberately and after source and target have been clearly identified.**

## Tool groups

**35 rescue tools in ten functional groups:**

**A.1–A.5 – Data recovery**
TestDisk, PhotoRec, GNU ddrescue, Foremost and Extundelete for backing up damaged storage devices and for recovering lost partitions and files.

**A.6–A.9 – Storage devices**
GParted, GNOME Disks, SMART diagnostics and NVMe diagnostics for examining, managing and assessing the condition of storage devices and partitions.

**A.10–A.14 – Security**
ClamAV Scan, ClamAV signatures, Chkrootkit, Rkhunter and GUFW Firewall for malware checks, system analysis and controlled network protection.

**A.15–A.16 – Analysis**
Binwalk and Hexedit for examining binary data, file structures, firmware images and unknown or damaged files.

**A.17–A.24 – Network**
Nmap, Wireshark, TCPDump, ARP Scan, MTR, Dig, Whois and Chromium for network diagnostics, packet analysis, reachability checks, name resolution and controlled research.

**A.25–A.26 – Encryption**
Cryptsetup and GnuPG for controlled access to encrypted storage devices and for cryptographic file, key and signature tasks.

**A.27–A.28 – Remote**
SSH and SSHFS for encrypted remote connections and for accessing remote file systems during rescue and backup work.

**A.29–A.33 – Hardware**
HWInfo, Inxi, LSHW, DMI Decode and Sensors for identifying and diagnosing the hardware present and its operating parameters.

**A.34 – Files**
Midnight Commander for local and remote file management and for the controlled copying, moving and organising of rescue data.

**A.35 – Documents**
QPDF for the technical checking, structural editing and, where applicable, repair of damaged PDF files.

The 35 tools complement each other within a rescue case. Not every program is suitable for every target system, and not every damage detected should be repaired immediately.

The basic workflow of JUMBO BLOCK Rescue is therefore:

**Identify → assess condition → protect the original → back up or create an image if necessary → examine the working copy → rescue or repair selectively → check the result.**

**JUMBO BLOCK Rescue is not a tool for a single operating system, but an independent rescue environment for examining and rescuing a wide variety of systems, storage devices, files and network environments.**

# Assignment of areas, programs and versions

| No. | Area | Program | Baseline version |
|---|---|---|---|
| **A.1** | Data recovery | TestDisk | 7.2 or later |
| **A.2** | Data recovery | PhotoRec | 7.2 or later |
| **A.3** | Data recovery | GNU ddrescue | 1.29 or later |
| **A.4** | Data recovery | Foremost | 1.5.7 or later |
| **A.5** | Data recovery | Extundelete | 0.2.4 or later |
| **A.6** | Storage devices | GParted | 1.6.0 or later |
| **A.7** | Storage devices | Disks / GNOME Disks | 46.1 or later |
| **A.8** | Storage devices | SMART diagnostics | smartmontools 7.4-3 |
| **A.9** | Storage devices | NVMe diagnostics | nvme-cli 2.13 or later |
| **A.10** | Security | ClamAV Scan | 1.4.3 or later |
| **A.11** | Security | ClamAV signatures | 1.4.3 or later |
| **A.12** | Security | Chkrootkit | 0.58b or later |
| **A.13** | Security | Rkhunter | rkhunter 1.4.6-13 |
| **A.14** | Security | GUFW Firewall | 24.04.0 or later |
| **A.15** | Analysis | Binwalk | 2.4.3 or later |
| **A.16** | Analysis | Hexedit | 1.6 or later |
| **A.17** | Network | Nmap | 7.95 or later |
| **A.18** | Network | Wireshark | 4.4.18 |
| **A.19** | Network | TCPDump | 4.99.5 or later |
| **A.20** | Network | ARP Scan | 1.10.0 or later |
| **A.21** | Network | MTR | 0.95 or later |
| **A.22** | Network | Dig | bind9-dnsutils 1:9.20.29-1 |
| **A.23** | Network | Whois | 5.6.3 |
| **A.24** | Network | Chromium | 154.0.8037.57 or later |
| **A.25** | Encryption | Cryptsetup | 2.7.5 or later |
| **A.26** | Encryption | GnuPG | 2.4.7 or later |
| **A.27** | Remote | SSH | openSSH 10.0p1 or later |

| No. | Area | Program | Baseline version |
|---|---|---|---|
| **A.28** | Remote | SSHFS | sshfs 3.7.3-1.2 |
| **A.29** | Hardware | HWInfo | 21.82 or later |
| **A.30** | Hardware | Inxi | 3.3.38 or later |
| **A.31** | Hardware | LSHW | lshw 02.19.git.2021.06... or later |
| **A.32** | Hardware | DMI Decode | dmidecode 3.6 or later |
| **A.33** | Hardware | Sensors | lm-sensors 3.6.2 or later |
| **A.34** | Files | Midnight Commander | 4.8.33 or later |
| **A.35** | Documents | QPDF | 12.2.0-1 or later |

# A.1 TestDisk

**Area:** Data recovery
**Program:** TestDisk
**Baseline 20261001:** TestDisk 7.2
**Installed package:** `testdisk 7.2-0.1`
**Operation:** text-based, interactive user interface in the terminal

## Purpose

TestDisk is a free open-source tool for recovering lost partitions, repairing certain file system structures and restoring deleted files. In JUMBO BLOCK Rescue, TestDisk is therefore one of the central tools for logical data recovery.

Baseline 20261001 contains the package `testdisk` in version `7.2-0.1`. The official documentation for TestDisk 7.2 names as its main areas of application the recovery of lost partitions, the repair of boot sectors and the restoration of files from damaged or deleted file system structures.

TestDisk does not work like an ordinary file manager. It analyses the structures of a storage device and can reconstruct information that can no longer be reached normally because of a damaged or deleted partition table.

## Typical use cases

In JUMBO BLOCK Rescue, TestDisk can be used in particular when a partition has been deleted by mistake, a partition table is damaged, an existing file system can no longer be reached normally by the operating system or certain boot sector structures have been damaged.

TestDisk also has functions for restoring deleted files. The official documentation names FAT12/16/32, exFAT, NTFS and ext2 for this. With ext3 and ext4, TestDisk can analyse file system structures, but direct restoration of deleted file contents via the undelete function is not possible in the same way because of the metadata available there.

## Starting and basic operation

Within JUMBO BLOCK Rescue, TestDisk is run in the terminal. The user interface is operated entirely by keyboard.

After starting, TestDisk first offers the option of creating a log file. The official documentation normally recommends **Create**. This creates a file named `testdisk.log` in the current directory.

Next, the storage device to be examined is selected. TestDisk shows the detected drives together with their size. This is followed by the selection or detection of the partition table type in use. TestDisk tries to detect the appropriate type automatically; according to the official guide, the suggested default value is usually correct.

### Analysing a partition table

The **Analyse** menu item is used to search for lost partitions.

TestDisk first displays the currently detected partition table. **Quick Search** can then be used to search for existing or lost partitions. Partitions found are displayed while the search is still running.

An important way of checking is to highlight a partition found and press **P** to display its files. This check helps to establish whether the partition being sought has actually been found. The file view is supported for FAT, exFAT, NTFS and ext2/3/4.

If **Quick Search** does not find the partition being sought, **Deeper Search** is available. This deeper search examines the storage device more thoroughly and can accordingly take considerably longer.

### Recovering a partition

TestDisk displays the partitions found together with their status. Partitions marked as deleted are not recovered automatically.

The user must therefore check which of the partitions found actually belong to the original structure of the storage device. Only after size, position and, if possible, the files contained have been checked can the status of a partition to be recovered be set accordingly.

**Write** can then be used to write the selected partition structure to the partition table. This step changes the storage device being examined. The official documentation expressly advises **confirming Write only when the partition structure displayed has been checked completely and all required partitions are listed correctly**.

### Restoring deleted files

In addition to complete partitions, TestDisk can also locate individual deleted files or directories on supported file systems.

The corresponding route leads via **Advanced** and then **Undelete** for the partition concerned. Deleted files found are displayed and can be selected and copied to a target medium.

The choice of target medium is crucial: restored files should **not be written back to the same file system from which they are being rescued**. Every write operation on the affected storage device can overwrite deleted data that is still present and thus prevent further recovery. The TestDisk documentation therefore expressly recommends a different target medium or at least a different file system.

### Boot sectors and file system structures

TestDisk also contains functions for repairing certain file system structures. The documentation for version 7.2 describes, among other things, the repair or reconstruction of FAT, exFAT and NTFS boot sectors and working with ext2/3/4 superblocks.

With FAT32 and NTFS, for example, an existing backup boot sector can be used to restore the main boot sector. For this purpose, TestDisk can detect the backup boot sector and copy it to the main boot sector via **Backup BS**.

These functions intervene directly in file system structures and should therefore only be used after prior analysis and backup.

### Procedure for physically damaged storage devices

TestDisk is primarily a tool for **logical** data recovery. With storage devices that have read errors or defective sectors, you should not work directly on the damaged original for longer than necessary.

For damaged storage devices, the official TestDisk documentation recommends first creating a copy or image and then continuing to work on this clone.

For this purpose, JUMBO BLOCK Rescue contains a separate tool, **GNU ddrescue**, which is covered in detail in section **A.3**.

For a physically conspicuous storage device, this results in the following, safer working sequence:

**Original storage device → sector-by-sector backup/image → analysis of the image → recovery of the data**

TestDisk can also work directly with storage device images. The official documentation expressly lists working with disk images as a supported mode of operation.

### TestDisk and PhotoRec

TestDisk and PhotoRec belong to the same data recovery package but take different approaches.

TestDisk works primarily with existing partitioning and file system structures. If recovery is successful, original partitions, directory structures and file names, for example, can therefore be preserved.

PhotoRec, by contrast, works on the basis of signatures and searches for known file types independently of an intact file system structure. If TestDisk can no longer reconstruct lost files from the existing file system information, the official documentation also recommends using PhotoRec.

PhotoRec is covered separately in the following section, **A.2**.

### Safety notes

In data recovery, protecting the data that still exists is more important than repairing as quickly as possible.

No new data should therefore be written to an affected storage device if at all possible. In particular, formatting, reinstalling an operating system or copying new files to the storage device can overwrite data that could still be recovered.

Particular care is required with TestDisk functions that actually make changes to the storage device. These include, in particular, writing a new partition table via **Write** and overwriting or reconstructing boot sectors.

As long as you are only analysing and searching, you can first check which structures TestDisk detects. Writing repair steps should only be carried out once the storage device and the structures to be recovered have been clearly identified.

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, TestDisk forms the first tool stage for **structure-oriented data recovery**.

It is particularly suitable when the actual data is probably still present but access is prevented by a lost partition, a damaged partition table or certain damaged file system structures.

With a technically unstable storage device, on the other hand, GNU ddrescue should first be used to back up the original. If the file system structure is so badly damaged that structure-oriented recovery is no longer possible, PhotoRec additionally provides signature-based data recovery.

The first three data recovery tools of JUMBO BLOCK Rescue thus complement each other functionally:

**TestDisk – analyse and reconstruct structures**
**PhotoRec – find files again on the basis of their contents**
**GNU ddrescue – back up damaged storage devices as gently as possible**

# A.2 PhotoRec

**Area:** Data recovery
**Program:** PhotoRec
**Baseline 20261001:** PhotoRec 7.2
**Installed package:** `testdisk 7.2-0.1`
**Operation:** text-based, interactive user interface in the terminal

## Purpose

PhotoRec is a data recovery program for the signature-based restoration of files. It belongs to the TestDisk package but takes a different approach to data recovery from TestDisk.

In JUMBO BLOCK Rescue Baseline 20261001, the package `testdisk` is installed in version `7.2-0.1`. In addition to TestDisk, this package also provides PhotoRec.

The developer CGSecurity describes PhotoRec as a **file carver**. The program searches for known file signatures and can therefore find files even when the underlying file system is damaged. As a rule, PhotoRec does not reconstruct the original directory structure or the original file names.

## Typical use cases

PhotoRec is used in particular when conventional recovery via existing file system structures is no longer possible or has not been successful.

Typical use cases are deleted files, damaged file systems, formatted partitions and storage devices on which the original file system structure can no longer be used sufficiently.

Among the data detected by PhotoRec, CGSecurity names ZIP archives, Office documents, PDF and HTML files, JPEG images and numerous other graphics and file formats. According to the developer, PhotoRec 7.2 supports more than **300 file families with more than 480 file extensions**.

## Working principle

For the actual file recovery, PhotoRec does not rely on the original directory entries and file names.

Instead, it examines the data areas for characteristic signatures of known file formats. If such a signature is detected, PhotoRec tries to reconstruct the associated data as a file.

This method has one major advantage: even a badly damaged file system can still contain usable files.

The disadvantage is that the original logical organisation is normally lost. Restored files therefore do not automatically get back their original file names and directory paths.

### Starting and selecting the storage device

Under Linux, PhotoRec is run in the terminal. Appropriate access rights are required for direct access to physical storage devices; for Linux, the documentation for PhotoRec 7.2 describes running it with root privileges for this purpose.

After starting, PhotoRec displays the available storage devices.

The storage device to be examined is selected with the arrow keys and then confirmed. The partition in which lost files are to be searched for is then selected.

Before starting a recovery, check carefully that the correct source storage device has actually been selected.

### Selecting the source data

After a partition has been selected, PhotoRec provides the following functions, among others:

**Search** starts the data recovery.

**Options** opens additional settings for the recovery procedure.

**File Opt** determines which file types PhotoRec is to search for.

**File Opt** is particularly useful with large storage devices when specific file formats are to be searched for.

However, the developer points out that disabling too many file formats can have unexpected effects. For file formats whose size cannot be determined directly from the file header, PhotoRec uses, among other things, the occurrence of a new known file signature to determine where a file ends. If too many signatures are disabled, oversized reconstructed files can result.

### Selecting the file system type

Before the actual search, PhotoRec asks for the file system type.

For **ext2/ext3/ext4**, the corresponding ext file system option is used. For other supported file systems, **Other** is selected in accordance with the developer documentation.

PhotoRec uses this selection to take the organisation of the data blocks into account correctly.

### Entire partition or free space only

Depending on the file system, PhotoRec can examine either the entire partition or only the unallocated space.

Searching the **entire partition** is particularly useful with a damaged file system.

Searching only the **free or unallocated space**, by contrast, concentrates on deleted files. According to the PhotoRec 7.2 documentation, this option is available for ext2/ext3/ext4, FAT12/16/32 and NTFS.

The selection thus directly affects the scope and aim of the data recovery.

### Selecting the file types to be recovered

**File Opt** can be used to enable or disable individual file types or file families.

This makes it possible, for example, to concentrate a rescue on particular document, image, archive or multimedia formats.

PhotoRec also makes it possible to define your own file signatures. Under Linux, a file `.photorec.sig` in the home directory or a `photorec.sig` in the current directory can be used for this. This also allows file formats to be taken into account for which PhotoRec does not know a suitable signature by default.

### Selecting the destination directory

Before the recovery begins, a destination directory must be selected for the rescued files.

One of the most important rules of data recovery as a whole applies here:

**Rescued files must not be written to the source file system being examined.**

New write operations could overwrite areas in which data that has not yet been recovered is located. Files that could previously be rescued could thus be lost for good.

CGSecurity expressly warns against using the source file system as the recovery destination at the same time.

The destination should therefore be on a different storage device or on a suitable different file system.

The developer also points out that FAT32 can be problematic as a destination if files larger than 4 GB are to be recovered.

### Carrying out the recovery

After starting, PhotoRec examines the selected data areas.

The number of files already recovered is displayed continuously during the process. PhotoRec saves the rescued files in directories with names such as:

```
recup_dir.1
recup_dir.2
recup_dir.3
```

These files can be accessed while a recovery is still running.

PhotoRec carries out several processing steps or passes. In the first phase, the block size can be determined, among other things; this is followed by the actual recovery of detected files.

## File names and directory structure

As PhotoRec reconstructs the files on the basis of their contents and signatures, original file names and directory structures are normally no longer available.

By default, the generated file names begin with a letter and a number.

CGSecurity documents the following, among others:

`f` – recovered file
`b` – damaged file, or file retained as damaged
`t` – embedded JPEG thumbnail

With certain file formats, PhotoRec can use existing metadata. Information from documents or time information from images, for example, can thus be used for naming or for chronological assignment. However, the developer expressly warns against regarding such time information as reliable without checking it in forensic examinations.

## Options for difficult cases

PhotoRec 7.2 has several options documented by the developer for special rescue situations.

**Paranoid** checks recovered files and, by default, discards results detected as invalid.

An additional **Bruteforce** option can be used to recover more heavily fragmented JPEG files. According to the developer, this method is very computationally intensive.

**Keep corrupted files** makes it possible to keep files or file fragments detected as damaged as well. These can then possibly be examined further with other tools.

**Low memory** reduces the memory requirement for systems on which PhotoRec runs into problems during a recovery because of insufficient RAM. CGSecurity recommends using this option only when it is actually necessary.

**Expert Mode** allows, among other things, the file system block size and offset to be specified manually and is thus intended for special recovery situations.

## Working with storage device images

PhotoRec can examine not only physical storage devices but also storage device images.

The documentation for version 7.2 expressly names **RAW images (`.dd`)** and **EnCase EWF images (`.E01`)** as well as split EnCase images. With storage device images, PhotoRec does not need direct administrative access to a physical drive.

PhotoRec is therefore particularly well suited to working with previously created backup images.

With a physically damaged or unstable original storage device, an image or clone should therefore be created first wherever possible. Within JUMBO BLOCK Rescue, **GNU ddrescue** is available for this.

### Fragmented files

File carving has technical limits.

Heavily fragmented files in particular can be problematic because their data is not necessarily stored contiguously on the storage device.

CGSecurity expressly describes this using the example of certain video recordings from digital cameras. PhotoRec basically expects files that are not fragmented; certain fragmented video formats can therefore initially be unusable after recovery and require additional reconstruction steps.

A file signature that has been found therefore does not automatically guarantee that a completely intact file can be recovered.

### Resuming an interrupted recovery

If a running PhotoRec recovery is interrupted, PhotoRec can offer to resume the process the next time it is started.

On completion, the program displays a summary of the recovery.

### Safety notes

No further write operations should take place on a storage device with lost data if at all possible.

Every new write operation can overwrite data blocks that still contain parts of deleted files.

The following therefore applies to JUMBO BLOCK Rescue:

**Leave the source as unchanged as possible.**

**Write recovered files to a different storage device.**

**With a physically conspicuous storage device, create a backup image first.**

**Carry out the actual data recovery on the image or clone wherever possible.**

This procedure reduces additional changes to the original medium and corresponds to the basic safety notes in the PhotoRec documentation.

### TestDisk or PhotoRec?

Whether TestDisk or PhotoRec is used depends on the condition of the data structures.

**TestDisk** tries to use existing partitioning and file system structures or to restore certain damaged structures. This makes it possible to preserve original file names and directory structures.

**PhotoRec**, by contrast, searches on the basis of file contents or file signatures. It can therefore still find files even on damaged file systems, but as a rule loses the original file system organisation in the process.

For lost partitions or certain deleted files, CGSecurity recommends trying TestDisk first, as successful structure-oriented recovery can be faster and can preserve original file names. PhotoRec is particularly important when this route is no longer successful.

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, PhotoRec provides **signature-based data recovery**.

The tool complements TestDisk particularly in situations in which partitioning or file system structures can no longer be used sufficiently for structure-oriented recovery.

Together with GNU ddrescue, this results in a graduated rescue strategy:

**TestDisk – examine partitioning and file system structures and, where supported, restore them**

**PhotoRec – reconstruct files on the basis of their signatures, independently of an intact file system structure**

**GNU ddrescue – first copy problematic storage devices as gently as possible to another medium or into an image**

With an intact storage device, analysis can begin immediately. If there are signs of a physical defect, however, backing up the original takes priority over further intensive analysis or rescue attempts.

# A.3 GNU ddrescue

**Area:** Data recovery
**Program:** GNU ddrescue
**Baseline 20261001:** GNU ddrescue 1.29
**Installed package:** `gddrescue 1.29-1`
**Operation:** command line in the terminal

## Purpose

GNU ddrescue is a data recovery tool for copying data from a file or block device to another target. Its focus is on rescuing as much readable data as possible in the event of read errors, initially treating problem areas in such a way that easily readable areas are not blocked for longer than necessary.

In JUMBO BLOCK Rescue, GNU ddrescue therefore performs a different task from TestDisk and PhotoRec: it first tries to **transfer the readable content of a problematic storage device as completely as possible to another medium or into an image file**.

For this purpose, Baseline 20261001 contains the package `gddrescue` in version `1.29-1`.

GNU itself describes ddrescue as a data recovery tool that copies data from one file or block device to another and, in the event of read errors, tries to rescue the good areas first.

## Distinction from `dd` and `dd_rescue`

GNU ddrescue must not be confused with the Unix/GNU tool `dd`.

`dd` is a general tool for copying and converting data. GNU Coreutils itself points out that, for failing storage media, tools such as GNU ddrescue have additional functions designed to back up as much data as possible before a device finally fails.

Nor is GNU ddrescue identical to the separately developed program `dd_rescue`.

In this manual, **ddrescue refers exclusively to GNU ddrescue 1.29** from the baseline package `gddrescue`.

## Typical use cases

GNU ddrescue is intended in particular for storage devices on which errors occur during reading.

These can include, for example, hard disks, SSDs, USB storage devices, memory cards, optical media or individual partitions, provided that the operating system still makes them available as a readable block device.

A typical rescue case is a storage device on which some areas can still be read without problems, while other areas cause I/O errors.

Instead of working intensively on the problem areas straight away, ddrescue first tries to back up as much easily readable material as possible.

## Basic principle

The central principle of GNU ddrescue is not to spend more time than necessary on the most difficult areas of a damaged storage device as long as other data can still be read without problems.

The aim is to back up a large proportion of the data still available on the target medium as early as possible.

Problem areas can then be processed in more detail in further phases.

This distinguishes ddrescue fundamentally from a simple linear copy operation.

## Source and target

A ddrescue call basically consists of:

```
ddrescue [Optionen] Eingabedatei Ausgabedatei [Mapfile]
```

The input can be, for example, a physical storage device or a partition.

Either another storage device or an image file can be used as the output.

Example of an image:

```
sudo ddrescue /dev/sdX rescue.img rescue.map
```

Here:

`/dev/sdX` stands for the source device,

`rescue.img` stands for the target file,

`rescue.map` stands for the mapfile.

**The device names in this example are placeholders. They must be checked against the system actually connected before real use.**

Confusing source and target can lead to serious data loss.

## The mapfile

One of the most important functions of GNU ddrescue is the **mapfile**.

The mapfile documents the state of the rescue process. It records which areas have already been copied successfully and which areas have not yet been processed or could not be processed successfully.

This means that an interrupted rescue process can be resumed later without having to read areas that have already been copied successfully completely again.

A mapfile should therefore be used and kept for serious rescue processes.

The mapfile is not the actual storage device image. Rather, it contains the information that ddrescue needs to manage the progress of the rescue.

### Resuming a rescue process

If ddrescue is terminated or a rescue process is interrupted, the same mapfile can be used when it is called again.

GNU ddrescue can use this information to detect which areas have already been processed.

Rescue attempts can thus be spread over several sessions.

This is particularly important with unstable or slowly failing storage devices, as a complete rescue may not be possible in a single pass.

### Rescue phases

GNU ddrescue does not process a storage device with just a single linear copy operation.

The rescue algorithm works in several phases. The largest possible easily readable areas are copied first. Problem regions are narrowed down and can then be processed further with finer accesses.

GNU ddrescue terminology distinguishes in particular between phases such as **copying**, **trimming**, **scraping** and **retrying**.

The purpose of this strategy is to back up as much easily accessible data as possible first and only then to spend more time on areas that are difficult to read.

### First rescue pass

A frequently used strategy is to dispense with additional retries for faulty areas in the first pass.

A corresponding call might, for example, be:

```
sudo ddrescue -f -n /dev/sdX /dev/sdY rescue.map
```

Or, when writing to an image file:

```
sudo ddrescue -n /dev/sdX rescue.img rescue.map
```

The option `-n` or `--no-scrape` skips the scraping phase.

The aim of such a first pass is to back up as much easily readable material as possible before additional time and additional read accesses are spent on problem areas.

### Further rescue attempts

After a first pass, the areas that have not yet been read successfully can be processed again.

For example:

```
sudo ddrescue -d -r3 /dev/sdX rescue.img rescue.map
```

The following option limits the number of retry passes to three:

```
-r3
```

`-r` or `--retry-passes` controls the number of additional retry passes for areas marked as faulty.

A higher number is not automatically better. With a physically unstable drive, additional attempts also mean additional load and operating time.

### Direct storage device access

The following options allow ddrescue to request direct storage device access for the input file:

`-d`

or

`--idirect`

This mode of operation can be helpful in certain rescue situations because it bypasses the normal kernel cache or reduces its influence.

However, actual support depends on the operating system and the device used.

### Writing to a block device

Particular care is required if ddrescue is to write directly to a physical target device.

The following option is required if the output file is a regular device or an existing device file that ddrescue would otherwise refuse to overwrite for safety reasons:

`-f`

or

`--force`

Example:

```
sudo ddrescue -f /dev/sdX /dev/sdY rescue.map
```

This command can overwrite the content of the target device.

**Source and target must therefore be identified beyond doubt before execution**.

### Image file or physical clone

GNU ddrescue can copy a source either to another storage device or into an image file.

An image offers advantages in many rescue situations.

After a successful backup, analysis tools can work on the image, while the original storage device is no longer needed for every further examination.

In JUMBO BLOCK Rescue, TestDisk or PhotoRec, for example, can then be used on such a storage device image.

A physical clone, on the other hand, can be useful if a duplicate that can be used directly as a storage device is required.

Which variant is used depends on the rescue case, the storage space available and the condition of the source medium.

### Sizing the target medium correctly

When cloning directly to another storage device, the target must be large enough.

With an image file, there must accordingly be enough free space on the target file system.

Before starting a rescue, both the size of the source device and the available storage space on the target medium should therefore be checked.

### Working independently of the file system

GNU ddrescue works at the level of the input data or block devices.

It therefore does not need an intact file system on the source for a sector-oriented rescue.

This is a key difference from tools that depend on file system structures for their work.

A damaged NTFS, FAT, ext or other file system therefore does not fundamentally prevent the still readable blocks from first being transferred with ddrescue into an image or to another medium.

The subsequent interpretation and recovery of the partitions and files contained in it is a separate task.

### Avoid mounting storage devices where possible

In data recovery, unnecessary write use of the damaged source file system should be avoided.

In particular, if the existing data is important, no repair programs, automatic file system repairs or normal write operations should be carried out on the original before a backup has been made.

For a rescue that is as controlled as possible, the original is therefore preferably treated as a source, and further analysis is carried out on the clone or image created.

### No repair by ddrescue

GNU ddrescue repairs neither partition tables nor file systems.

Nor does it restore deleted files on the basis of their file system information or file signatures.

Its task is initially to transfer the still readable data from a source to a target and to manage the progress of the rescue.

The actual logical analysis is then carried out with tools designed for this purpose.

In JUMBO BLOCK Rescue, these include:

**TestDisk** for partitioning and certain file system structures,

**PhotoRec** for signature-based file recovery.

## Handling a failing storage device

A physically conspicuous storage device should be subjected to as little unnecessary load as possible.

Typical warning signs can be recurring I/O errors, strongly fluctuating read rates or hardware problems reported by the drive or operating system.

With such media, the priority should be on backing up the still readable data.

Every additional full scan and every unnecessary repair attempt on the original extends the operating time of the possibly failing device.

In JUMBO BLOCK Rescue, the following sequence is therefore generally advisable for a physically problematic medium:

**Identify the original → prepare the target medium → ddrescue with mapfile → back up image/clone → further analysis on the duplicate**

## Limitations

GNU ddrescue can only copy data that the storage medium and the underlying hardware can still provide.

An area that physically can no longer be read even after repeated read attempts cannot be reliably reconstructed by software.

Even a successful ddrescue pass does not automatically mean that the file system it contains is free of errors or that every file is completely intact.

If individual source areas could not be read, this data is also missing from the image or clone created.

The result must therefore subsequently be checked with suitable analysis and data recovery tools.

## Safety notes

GNU ddrescue works directly with storage devices and can overwrite existing data if used incorrectly.

Before each start, the following points in particular must therefore be clarified beyond doubt:

**Which device is the source?**

**Which device or file is the target?**

**Is the target large enough?**

**May an existing target be overwritten?**

**Is the correct mapfile being used?**

Confusing the device names `/dev/sdX`, `/dev/sdY`, NVMe devices or other block devices is particularly critical.

The device names shown in this manual are therefore examples only and must not be adopted without checking.

## Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, GNU ddrescue is the central tool for **backing up problematic storage devices before the actual logical data recovery**.

It complements TestDisk and PhotoRec but performs a clearly different task.

Together, the three tools form a graduated rescue chain:

**GNU ddrescue – back up as much still readable data as possible**

**TestDisk – examine partitioning and supported file system structures and restore them if necessary**

**PhotoRec – reconstruct files on the basis of their signatures when the original structures can no longer be used sufficiently**

With a storage device showing signs of physical problems, creating as complete a clone or image as possible with GNU ddrescue generally takes priority over extensive repair attempts on the original.

The original remains the last available source. Wherever possible, every further analysis should be carried out on the backed-up image or clone.

# A.4 Foremost

**Area:** Data recovery
**Program:** Foremost
**Baseline 20261001:** Foremost 1.5.7
**Installed package:** `foremost 1.5.7-11+b2`
**Operation:** command line in the terminal

## Purpose

Foremost is a tool for **signature-based file recovery, or file carving**. It searches storage device images or other input data for characteristic headers, footers and internal data structures of known file formats.

JUMBO BLOCK Rescue Baseline 20261001 contains Foremost as the package `foremost` in version `1.5.7-11+b2`.

Foremost was originally developed for forensic examinations. The program documentation describes its purpose as recovering files on the basis of their headers, footers and data structures.

## Typical use cases

Foremost is particularly suitable for situations in which files can no longer be reached via an intact directory or file system structure.

These include, for example:

deleted files,

damaged file systems,

storage device images from a previous rescue,

partitions that can no longer be fully interpreted,

and the targeted search for specific file types within raw data.

As Foremost works on the basis of file signatures, no intact original directory structure is required for the actual file carving.

## Working principle

Foremost searches the input data for defined file headers – known as **headers** – and, where provided for the file type concerned, for associated **footers**.

If a matching structure is found, Foremost tries to extract the corresponding data area as a file.

The detection rules can already be defined in the program or provided via a configuration file.

Like PhotoRec, Foremost thus works at the level of file contents and signatures and not primarily via original file system entries.

### Supported built-in file types

The Foremost 1.5.7 documentation lists a number of predefined file types.

These include, among others:

`jpg` – JPEG/JFIF and Exif,

`gif`,

`png`,

`bmp`,

`avi`,

`exe` – Windows PE files including DLL and EXE,

`mpg`,

`wav`,

`riff`,

`wmv`,

`mov`,

`pdf`,

`ole` – files with an OLE structure, including older Microsoft Office formats,

`doc`,

`zip`,

`rar`,

`htm`,

`cpp`,

`mp4`.

`all` can be used to apply all predefined extraction methods. According to the man page, this is also the default if no file type is specified with `-t`.

### Basic call

The basic program call is:

```
foremost [Optionen]
```

An input file can be specified with the following option:

`-i`

Example:

```
foremost -i rescue.img
```

This examines `rescue.img`.

According to the program documentation, if no input file is specified or the specified input cannot be read, Foremost uses standard input `stdin`.

### Selecting the output folder

The following option specifies the directory to which Foremost writes the recovered files:

`-o`

Example:

```
foremost -i rescue.img -o recovery
```

This examines `rescue.img` and stores the result in the directory `recovery`.

In a real data recovery, source and output should be handled separately. In particular, recovered data must not be written back unnecessarily to a damaged original storage device.

### Selecting specific file types

The following option limits the search to specific file types:

`-t`

Example – the following call searches specifically for JPEG files:

```
foremost -t jpg -i rescue.img -o recovery
```

Several types can be specified together.

The Foremost documentation shows, for example, the following call for searching for GIF and PDF files:

```
foremost -t gif,pdf -i image.dd
```

To search for all built-in types, the following can be used:

```
foremost -t all -i rescue.img
```

### Verbose mode

The following option enables verbose output mode:

```
-v
```

Foremost then displays additional information about the current state of the program.

The program documentation expressly describes this mode as highly recommended.

Example:

```
foremost -v -i rescue.img -o recovery
```

### Quick mode

The following option enables **Quick Mode**:

```
-q
```

In this mode, Foremost only searches for matching headers at the beginning of each sector. This can make processing considerably faster.

However, this speed comes at the price of a restricted search.

Files whose header is not at the expected position can be missed. As an example, the Foremost documentation names embedded JPEG images within Microsoft Word documents.

The explicit note in the documentation is particularly important:

**Quick Mode should not be used with NTFS file systems.**

NTFS can store small files directly within the Master File Table. Such files can be missed in Quick Mode.

### Quiet mode

The following option enables **Quiet Mode**:

```
-Q
```

Most error messages are suppressed in this mode.

For a normal rescue analysis, this mode of operation is not fundamentally advantageous, as error messages can contain important information about problems with the source, output or data structures.

### Audit-only mode

The following option runs Foremost in **write-audit-only mode**:

```
-w
```

No files found are extracted in this mode.

This mode of operation thus allows an examination or logging without actually writing the files found to the output directory.

### Option `-a`

The following option instructs Foremost to process all headers found and not to apply the normal error detection regarding damaged files:

```
-a
```

This option can therefore produce results that would not be extracted in normal operation because of detected inconsistencies.

### Block size

The following option specifies the block size used by Foremost:

```
-b
```

According to the Foremost documentation, the default setting is **512 bytes**.

Example:

```
foremost -b 1024 -i rescue.img
```

The block size is particularly relevant for file naming and for searching in Quick Mode.

### Skipping areas

The following option makes it possible to skip a specific number of blocks at the beginning of the input:

```
-s
```

The documentation gives the following example:

```
foremost -s 100 -t jpg -i image.dd
```

This first skips 100 blocks before the search for JPEG files begins.

This function can be used for targeted examinations of specific areas of an image.

### Indirect block detection

The following option enables Foremost's **indirect block detection**:

```
-d
```

According to the program documentation, this option works particularly well with Unix file systems.

It can be used, for example, together with the search for specific file types.

An example shown in the documentation is, in essence:

```
foremost -vd -t ole,jpg -i image.dd
```

This examines OLE and JPEG structures with verbose output and indirect block detection enabled.

### Time-stamped output folders

The following option adds a time stamp to the output folder:

```
-T
```

This allows several Foremost runs to be carried out without having to remove the same output folder or assign a new name manually before each run.

This is particularly helpful when the same image is analysed repeatedly with different settings.

### Custom file signatures

An important function of Foremost is the ability to adapt the detection rules via a configuration file.

The following option specifies a particular configuration file:

```
-c
```

Example:

```
foremost -c eigene.conf -i rescue.img -o recovery
```

If no configuration file is specified explicitly, Foremost first looks for `foremost.conf` in the current directory and then for `/etc/foremost.conf`.

### Structure of `foremost.conf`

For a file type, the configuration file defines, among other things:

the file extension,

the case sensitivity of the signature check,

the maximum file size,

the header,

and, optionally, the footer.

Header, maximum size, case sensitivity and file extension are required; the footer is optional.

Lines beginning with # are treated as comments and ignored. This allows individual file type definitions to be disabled.

### Binary signatures

Foremost also allows signatures to be defined in hexadecimal or octal notation.

Hexadecimal values can be specified, for example, as follows:

```
\xFF
```

Wildcards are also possible within the signature definition. By default, ? serves as a placeholder for any single character.

This allows custom or special file formats to be described, provided that their characteristic data structures are known.

### Audit file

While working, Foremost creates audit documentation of the run.

Precisely because of its origins in the forensic field, traceable documentation of a carving run is an important part of the program.

The audit information should be kept together with the rescued data if the traceability of the rescue process will be relevant later.

### Foremost and storage device images

Foremost is particularly suitable for examining storage device images.

This fits directly with the rescue strategy of JUMBO BLOCK Rescue:

A physically conspicuous storage device can first be copied as completely as possible with **GNU ddrescue** to another medium or into an image file.

Foremost can then examine the image without having to access the problematic original medium again for each carving attempt.

Example:

```
foremost -v -i rescue.img -o foremost-recovery
```

### Limitations of file carving

Foremost reconstructs files on the basis of detected data structures and signatures.

This does not automatically mean that every file found is complete and free of errors.

Fragmentation in particular can be a problem. If the parts of a file are not stored contiguously, a purely signature-based reconstruction can produce incomplete or faulty results.

Likewise, random byte sequences can match a known signature and thus lead to falsely detected results.

The extracted files must therefore be checked afterwards.

### Original file names and directory structure

As a rule, Foremost does not reconstruct the original file system organisation.

As the files are found on the basis of their content or signatures, original file names and directory paths are normally not available.

This is a fundamental difference from a successful structure-based recovery with TestDisk.

### Foremost and PhotoRec

Foremost and PhotoRec both belong to the group of file carving tools.

Both can find files again on the basis of characteristic file contents without depending on a completely intact original file system structure.

However, they are independent programs with different detection mechanisms, supported formats, configuration options and operating concepts.

In particular, Foremost offers an explicit, editable signature configuration via `foremost.conf`.

PhotoRec, by contrast, has its own extensive database of supported file formats and an interactive text-based workflow.

Within JUMBO BLOCK Rescue, both tools are therefore deliberately provided separately.

### Safety notes

With a damaged original storage device, Foremost should, if possible, not be used as the first step for extensive repeated scans.

If there are signs of a physical defect, as complete an image or clone as possible should first be created with GNU ddrescue.

Foremost can then work on this image.

Sufficient storage space must be available for the output. A broad search for numerous file types can produce a large number of files.

The output directory should also be on a safe target medium and not on the source storage device to be rescued.

### Role in JUMBO BLOCK Rescue

Foremost adds a second, independent procedure for **signature-based file recovery** to JUMBO BLOCK Rescue.

Within the data recovery area, this results in:

**TestDisk – recovery and repair of supported partitioning and file system structures**

**PhotoRec – extensive signature-based file carving**

**GNU ddrescue – backing up problematic storage devices to another medium or into an image**

**Foremost – header/footer and structure-oriented file carving with configurable signatures**

Foremost is particularly useful when specific file types are to be searched for or custom detection rules via `foremost.conf` are required.

# A.5 Extundelete

**Area:** Data recovery
**Program:** Extundelete
**Baseline 20261001:** Extundelete 0.2.4
**Installed package:** `extundelete 0.2.4-3+b2`
**Operation:** command line in the terminal

## Purpose

Extundelete is a specialised data recovery tool for restoring deleted files from **ext3 and ext4 file systems**.

JUMBO BLOCK Rescue Baseline 20261001 contains the package `extundelete` in version `0.2.4-3+b2`.

Extundelete uses information from the file system journal to try to restore previously deleted files. The program documentation expressly points out that the successful recovery of a particular file cannot be guaranteed.

## Area of application

Extundelete is not a general file carving program.

Its area of application is expressly geared to **ext3 and ext4**. It tries to use file system and journal information that is still present to restore deleted files.

This distinguishes Extundelete from PhotoRec and Foremost, which search for files primarily on the basis of file signatures.

If the required metadata is still present, Extundelete can restore files under their original name or with reference to their original paths. With `--restore-all`, files whose original name can no longer be determined are output under a name following the pattern `file.NNNN`.

## Basic call

The syntax is:

```
extundelete [Optionen] device-file
```

An ext3 or ext4 partition can be specified as the source.

Example:

```
sudo extundelete /dev/sdXN --restore-all
```

Here, `/dev/sdXN` and all comparable device names in this manual are placeholders only. The device actually affected must be identified beyond doubt before use.

Extundelete can also **work with a file containing a copy of the partition**, for example a previously created raw image. The man page expressly names a copy created with `dd` as a possible input.

This means that Extundelete can also be used on a previously backed-up partition image.

### Restoring a single file

The following option makes a targeted attempt to restore a specific deleted file:

```
--restore-file
```

Example:

```
sudo extundelete /dev/sdXN --restore-file home/user/dokument.txt
```

The path is specified relative to the root of the file system concerned.

Extundelete then tries to restore the specified file on the basis of the information still available.

### Restoring a directory

The following option attempts to restore the files of a specific directory:

```
--restore-directory
```

Example:

```
sudo extundelete /dev/sdXN --restore-directory home/user/Documents
```

The man page describes this function as an attempt to restore all files that can be assigned to the specified directory. Where possible, the names the files had before they were deleted are used.

### Restoring all files that can be found

With the following option, Extundelete tries to restore all files that can be rescued with the information still present:

```
--restore-all
```

Example:

```
sudo extundelete /dev/sdXN --restore-all
```

Where the original names can be determined, they are used. For other recovered files, names following this pattern can be generated:

```
file.NNNN
```

This function can be useful after a larger deletion, but potentially produces a correspondingly large set of results.

### Restoring several selected files

The following option processes a list of files to be restored:

```
--restore-files
```

To do this, first create a text file containing the required paths in the same form as they would be specified with `--restore-file`.

You can then run, for example:

```
sudo extundelete /dev/sdXN --restore-files dateiliste.txt
```

The man page points out that this method reduces redundancy compared with many individual calls, because the journal does not have to be processed completely again for each individual file.

### Restoring on the basis of an inode number

Extundelete can also try to restore data on the basis of specific inodes.

The following option is available for this:

```
--restore-inode
```

Example:

```
sudo extundelete /dev/sdXN --restore-inode 2569
```

Several inode numbers can also be specified, separated by commas:

```
--restore-inode 2569,5692,6925
```

This function is particularly useful for a targeted technical examination when the relevant inodes are already known.

### Examining the superblock

With the following option, Extundelete can output information about the superblock of the file system:

```
--superblock
```

Example:

```
sudo extundelete /dev/sdXN --superblock
```

This function is used to examine the file system and does not itself restore any file.

### Examining the journal

The following combination outputs information about the journal on the basis of its superblock:

```
--journal --superblock
```

As Extundelete uses information from the journal for its recovery attempts, the journal is a central part of its working principle.

### Displaying inode information

The following option outputs information on a specific inode number:

```
--inode
```

Example:

```
sudo extundelete /dev/sdXN --inode 2
```

This is an analysis function and can help when examining existing file system structures.

### Displaying block content

The following option outputs the content of a specific block:

```
--block
```

Example:

```
sudo extundelete /dev/sdXN --block 9652
```

This too is an analysis function for targeted examinations of the file system.

### Restricting by time

Extundelete can restrict recovery attempts in time on the basis of the time of deletion.

The following two options are available for this:

```
--before
```

and

```
--after
```

The time is specified as the number of seconds since the Unix epoch.

The following option takes into account files that were deleted before the specified time:

```
--before
```

The following option takes into account files that were deleted after the specified time:

```
--after
```

Under Linux, a Unix time value can be generated, for example, with `date`:

```
date -d "2026-10-01 10:00" +%s
```

This allows a recovery attempt to be restricted specifically to a known period.

### External journal

Ext file systems can use an external journal.

The following option specifies such an external journal device for Extundelete:

```
-j journal_dev
```

This option is only relevant if the file system to be examined has actually been configured accordingly.

### Backup superblock

The following option uses a specific backup superblock when the file system is opened:

```
-b block_number
```

In addition, the following option specifies the block size to be used:

```
-B block_size
```

These options are among the more technically advanced functions and should only be used if the structure and condition of the file system concerned are sufficiently known.

### Why acting quickly is important

Extundelete can only successfully restore a deleted file if the information or data required for this is still available.

After a deletion, released data blocks can be reused by the file system. Relevant journal information is not available indefinitely either.

After an accidental deletion, unnecessary write accesses to the affected file system should therefore be avoided.

The program documentation expressly states that the successful recovery of a particular file **cannot be guaranteed**.

### Original or image

For JUMBO BLOCK Rescue, working with a previously backed-up image is particularly important.

The Extundelete man page expressly allows not only a partition as input but also a file containing a copy of this partition.

A sensible rescue workflow can therefore be:

**Back up the original storage device → create a partition image → use Extundelete on the image**

With a physically conspicuous storage device, GNU ddrescue can be used for the prior backup.

This reduces further analysis accesses to the possibly unstable original.

### No guarantee of recovery

Extundelete has no way of restoring file data that has already been physically overwritten.

Nor does existing journal information guarantee that all required data blocks of a deleted file are still available.

The man page therefore expressly states that there is **no guarantee that a particular file can be restored**.

A failed recovery attempt therefore does not necessarily mean that Extundelete has malfunctioned. The information needed for a successful reconstruction may simply no longer be present.

### Distinction from TestDisk

TestDisk and Extundelete partly overlap in the area of deleted files but have different focuses.

TestDisk is a more comprehensive tool for examining and restoring partitioning and certain file system structures.

Extundelete, by contrast, is specifically geared to restoring deleted files from **ext3 and ext4 file systems** and uses information from their journal for this.

### Distinction from PhotoRec and Foremost

PhotoRec and Foremost work primarily on the basis of signatures.

They search for characteristic data structures of known file formats and can therefore deliver results even when original file system information is no longer sufficiently available.

Extundelete, by contrast, tries to use information from the ext3/ext4 file system and its journal.

The advantage of a successful structure-based recovery is that original names and relationships can be preserved.

If this recovery is no longer possible, PhotoRec or Foremost can be used as alternative file carving procedures.

### Distinction from GNU ddrescue

GNU ddrescue does not restore deleted files.

Its task is to copy still readable data from a problematic storage device as completely as possible to another medium or into an image.

Extundelete can then work on such a copy, provided that it contains a suitable ext3 or ext4 file system.

The two programs thus complement each other directly:

**GNU ddrescue backs up the still readable raw data.**

**Extundelete then tries to reconstruct deleted files from ext3/ext4 structures.**

### Limitations

Extundelete is specialised in **ext3 and ext4**. It is not the intended tool for NTFS, FAT, exFAT and other file systems.

Its recovery capabilities also depend on which file system, journal and file data are still present.

It is therefore no substitute for a backup.

The developer and the program documentation also expressly point out that no particular file is guaranteed to be restored.

### Safety notes

After important files have been deleted accidentally, the affected ext3/ext4 file system should, if possible, not continue to be used normally.

New write operations can reoccupy released data blocks and thus reduce the chances of recovery.

Restored data should therefore be stored on a **different file system or target medium**.

If there are signs of a hardware defect, you should also not start with repeated intensive analysis attempts on the original. In this case, backing up with GNU ddrescue first is preferable.

### Role in JUMBO BLOCK Rescue

Extundelete adds to the data recovery tools of JUMBO BLOCK Rescue a specialised procedure for deleted files on Linux file systems of the **ext3 and ext4** families.

The **data recovery** area of Baseline 20261001 thus comprises five different tools or procedures:

**TestDisk** – examine and restore partitioning and supported file system structures.

**PhotoRec** – reconstruct files on the basis of their signatures.

**GNU ddrescue** – back up problematic storage devices as gently as possible.

**Foremost** – configurable file carving on the basis of headers, footers and data structures.

**Extundelete** – restore deleted files from ext3/ext4 file systems using existing journal information.

The tools therefore do not replace one another. They cover different levels and situations of a data recovery and can build on one another within a controlled rescue workflow.

# A.6 GParted

**Area:** Storage devices
**Program:** GParted
**Baseline 20261001:** GParted 1.6.0
**Installed packages:** `gparted 1.6.0-2`, `gparted-common 1.6.0-2`
**Operation:** graphical user interface

## Purpose

GParted is the **GNOME Partition Editor** for the graphical management of partitions and partition tables.

JUMBO BLOCK Rescue Baseline 20261001 contains GParted in version `1.6.0-2`.

The official GParted manual for version 1.6.0 describes the program as a tool for creating, reorganising and deleting partitions. The documented functions include in particular creating partition tables and creating, deleting, resizing, moving, checking, labelling, copying and pasting partitions.

## Significance in the rescue system

In JUMBO BLOCK Rescue, GParted is used for the graphical examination and – where expressly required – editing of the partitioning of connected storage devices.

The program can be used, for example, to identify existing storage devices and partitions, to view their size and position, to detect unallocated space and to examine existing file systems or partition properties.

However, GParted is **not a read-only diagnostic tool**.

Many of its central functions directly change the partitioning or the file systems on it. A strict distinction must therefore be made between merely examining a storage device and changes that are actually to be carried out.

## Program interface

After starting, GParted displays the selected storage device and its partitioning both graphically and in table form.

The device selection can be used to switch between detected storage devices.

The display provides a quick overview of existing partitions and unallocated space.

Before any editing, check which physical storage device is selected. In a rescue system in particular, internal drives, USB storage devices and other storage media may be connected at the same time.

## Examining partitions

An existing partition can be selected and its information displayed.

For this purpose, the official manual expressly lists the **“Information”** function of a partition.

This examination must be distinguished from changes to the partition.

As long as no modifying operation is carried out, GParted can therefore initially serve to provide orientation on the existing structure of the storage device.

## Supported operations

GParted 1.6.0 supports different operations depending on the file system concerned and the auxiliary programs installed.

The basically documented partition operations include:

**Create**

**Delete**

**Resize**

**Move**

**Check**

**Set file system label**

**Copy**

**Paste**

**Manage partition flags**

GParted can also create partition tables on storage devices.

Not every operation is equally available for every file system. The actual possibilities depend on the file system and the tools available on the system.

## Creating a partition table

GParted can create a new partition table on a storage device.

This is a **destructive, potentially data-destroying operation**.

A new partition table must not be created lightly on a storage device with data to be rescued.

In particular, creating a new partition table is **not a repair measure for an existing partition table that is merely damaged or lost**.

If an existing partitioning structure is to be restored, a tool intended for this purpose, such as TestDisk, must first be considered within JUMBO BLOCK Rescue.

## Creating a new partition

Unallocated space can be used to create a new partition.

Depending on the selected partition table and the available file system tools, various properties can be specified.

Creating a new partition changes the structure of the storage device.

On a rescue original with lost data, new space should therefore not be partitioned or formatted without a prior backup.

### Deleting a partition

GParted can delete existing partitions.

Deleting removes the corresponding partition entry from the structure of the storage device.

This function is useful in normal administration, but in a data recovery case it represents a writing change to the original medium.

A partition that has been deleted by mistake or damaged should therefore not be “repaired” through further experimental changes to the partition table.

### Formatting a partition

GParted can format partitions with supported file systems, provided that the tools required for this are present.

Formatting creates new file system structures and can overwrite existing data structures.

**A partition with data to be rescued must therefore not be formatted.**

This also applies if an operating system reports that a partition must be formatted before it can be used.

Such a message is no proof that there is no longer any recoverable data.

### Resizing a partition

GParted can change the size of supported partitions and file systems.

**Partition** → **Resize/Move** is used for this.

According to the GParted manual, unmounted or inactive partitions offer the most options for resizing. Online resizing is possible for some active file systems, but it is often limited to growing.

To grow a partition, suitable unallocated space must be available.

### Moving a partition

GParted can move a partition within the available space.

This is also done via **Resize/Move**.

Moving can take considerably longer than merely changing partition boundaries, as data has to be transferred physically or logically to other positions.

The official manual also expressly warns that moving an operating system boot partition can result in the operating system no longer starting.

Such an operation must therefore not be regarded as a non-binding test.

### Copying and pasting a partition

GParted can copy an unmounted partition and paste it into suitable unallocated space.

The source partition is first selected with **Partition** → **Copy** and then transferred to the target area via **Partition** → **Paste**.

The official manual points out that the copy initially has the same file system label and UUID as the source partition.

This must be taken into account if the original and the copy are used in one system at the same time.

### Checking a partition

For supported file systems, GParted provides a **Partition** → **Check** function.

The file system tools available in each case are used for this.

Such a check is not to be equated with a purely passive display. Depending on the file system and the tool used, repair measures can be part of the operation.

With a storage device containing important data to be rescued, a backup or image of the original should therefore be created first before a file system repair.

### Partition flags

GParted can enable and disable partition flags.

The official manual names, for example, the flags `boot` and `hidden`.

Which flags are actually available depends on the partition table used and the selected partition.

Changing such flags modifies partition metadata and should therefore also be done deliberately.

### Pending operations

A particularly important feature of GParted is the management of **pending operations**.

Many changes are not carried out immediately at first but are queued.

This allows planned operations to be checked once more before they are actually carried out.

The queue can be corrected as long as the changes have not yet been applied.

The official manual describes functions for undoing the last operation and for clearing all pending operations. Only **“Apply All Operations”** actually processes the queued changes.

### Check before applying

The decisive safety moment is applying the queued operations.

Before this step, the following must be checked in particular:

the selected storage device,

the partition affected,

the planned operation,

size and position,

the target of a copy operation,

and the order of several pending operations.

GParted itself expressly warns that editing partitions can lead to **data loss** and recommends a data backup before use.

### Mounted partitions

Many partition operations require an unmounted or inactive partition.

This is one of the reasons why an externally booted rescue system is useful for partition work: the system partitions of the installed operating system do not necessarily have to be in active use.

GParted 1.6.0 does support individual operations in the mounted state for certain file systems, but the possibilities are limited.

### Encrypted partitions

GParted supports certain operations with encrypted partitions.

For LUKS, for example, the official manual states that an encrypted partition can only be moved if the encryption mapping is closed.

Particular care is required with encrypted storage devices.

The GParted manual expressly points out that a data backup is recommended before partition changes and emphasises this particularly for encrypted data.

### File system support

GParted supports numerous file systems, but not every operation for every file system.

The official feature overview therefore distinguishes for each file system individually between functions such as:

detect,

read,

create,

grow,

shrink,

move,

copy,

check,

change label,

change UUID.

In the JUMBO BLOCK Rescue manual, it should therefore not be assumed across the board that every GParted function is available for every detected file system.

What counts is the availability displayed by GParted on the specific system.

### Storage device types

According to the project documentation, GParted can work with different storage media.

These include conventional hard disks as well as flash memory such as USB sticks and SSDs. The GParted documentation also names NVMe devices and various RAID configurations.

For rescue use, regardless of device type, it remains crucial to identify source and target unambiguously.

### Special feature of GParted 1.6.0

For version 1.6.0, which is used in JUMBO BLOCK Rescue, one change compared with older GParted versions is particularly important:

With GParted **1.6.0, the former “Attempt Data Rescue” function was removed**, as was the use of `gpart` for this function. This is expressly stated in the official release announcement for GParted 1.6.0.

For this manual, this means:

**GParted 1.6.0 is not documented as a tool for recovering lost partitions.**

For the actual partition recovery, other tools are available in JUMBO BLOCK Rescue, in particular TestDisk.

### GParted and TestDisk

The tasks of GParted and TestDisk must be clearly separated in rescue operation.

**GParted** is used to display and deliberately manage existing partitioning structures and file systems.

**TestDisk** is geared to examining and restoring lost or damaged partitioning and certain file system structures.

If a partition has merely been lost or the partition table is damaged, a new partition or partition table should therefore not be created hastily with GParted.

This could change structures that TestDisk might still be able to use for a recovery.

### GParted and GNU ddrescue

GParted and GNU ddrescue also perform different tasks.

**GParted manages partitions and partition tables.**

**GNU ddrescue backs up as much still readable raw data as possible from a problematic storage device.**

With a physically conspicuous storage device, you should therefore not begin with extensive GParted operations.

Instead, it makes sense first to create as complete an image or clone as possible with GNU ddrescue and to carry out further examinations on the duplicate wherever possible.

### Typical rescue workflow

With an unknown or damaged storage device, GParted can first be used for visual orientation.

The storage device is selected and the partitioning displayed is checked.

If the existing structures are plausible and only administrative changes are needed, suitable GParted operations can be planned.

If, on the other hand, lost partitions, damaged file system structures or a hardware defect are suspected, you should **not intervene hastily with write operations**.

In that case, the specialised data recovery and diagnostic tools of JUMBO BLOCK Rescue must be used first.

### Limitations

GParted is no substitute for a backup.

It is not a tool for the physical data recovery of failing storage devices.

And GParted 1.6.0 no longer has the former “Attempt Data Rescue” function.

Furthermore, the successful completion of a partition operation does not guarantee that a damaged operating system will start again afterwards or that a previously damaged file system is completely free of errors.

### Safety notes

The GParted project expressly warns:

Editing partitions can lead to data loss. Software errors, hardware problems or a power failure during an operation can also cause data loss. A data backup is therefore recommended before partition changes.

The following therefore applies to JUMBO BLOCK Rescue:

**Identify first, then act.**

Before any writing operation, the storage device and partition must be clearly determined.

With valuable data or data that is not backed up elsewhere, a backup, image or clone should be created before structural changes wherever possible.

With physically conspicuous media, backing up with GNU ddrescue takes priority.

With lost partitions, TestDisk should be used first before new partition structures are written with GParted.

## Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, GParted is the central **graphical tool for displaying and managing partitions and partition tables**.

It adds the administrative level of the storage device structure to the data recovery programs documented so far.

A clear functional separation applies:

**GNU ddrescue – back up storage devices**

**TestDisk – examine and restore lost or damaged partitioning structures**

**PhotoRec and Foremost – rescue files on the basis of signatures**

**Extundelete – restore deleted files from ext3/ext4**

**GParted – graphically display and deliberately manage existing partitions and partition tables**

Precisely because GParted has powerful writing functions, it is both an important tool in the rescue system and a tool that must be used with particular care.

# A.7 Disks / GNOME Disks

**Area:** Storage devices
**Program:** Disks / GNOME Disks
**Baseline 20261001:** GNOME Disks 46.1
**Installed package:** `gnome-disk-utility 46.1-2+b1`
**Operation:** graphical user interface

## Purpose

**Disks**, known internationally as **GNOME Disks**, is the graphical storage device management tool of the GNOME project.

JUMBO BLOCK Rescue Baseline 20261001 contains `gnome-disk-utility` in version `46.1-2+b1`.

GNOME describes the application as a tool for **inspecting, formatting, partitioning and configuring disks and block devices**. In addition, SMART data can be displayed, physical drives can be benchmarked and storage device images can be used.

GNOME Disks thus combines several functions that are relevant to a rescue system:

identifying storage devices,

examining partitions and file systems,

checking drive condition,

displaying SMART data and starting self-tests,

measuring drive performance,

using storage device images,

managing partitions,

checking and repairing file systems,

and formatting storage devices or partitions.

Some of these functions are purely diagnostic, while others change the storage device. This distinction is crucial in rescue operation.

## Storage device overview

After starting, GNOME Disks displays the detected storage devices in a device list.

After a device has been selected, information on the storage device concerned and its existing volumes or partitions is displayed.

The program can therefore first be used to **identify a connected storage device**.

This is important before any further rescue measure.

If several drives are connected, it must be established unambiguously which device is the source and which, if applicable, is the target of an operation.

### Storage devices and block devices

GNOME Disks does not work exclusively with conventional hard disks.

The official GNOME description refers generally to **disks and block devices**. The application is thus designed as a management tool for corresponding storage and block devices detected by the Linux system.

In rescue operation, internal and external drives and USB storage media, for example, can therefore be examined via the graphical interface, provided that they are detected by the system as corresponding devices.

### Partitions and volumes

GNOME Disks displays the existing structure of a selected storage device graphically.

This makes it possible to distinguish existing partitions or volumes from one another and to examine basic properties of the individual areas.

The GNOME documentation expressly lists the management of **volumes and partitions** as a task of disk management.

This makes GNOME Disks a suitable first graphical tool for gaining an overview of an unknown storage device.

### Mounting and unmounting

File systems or volumes can be mounted and unmounted via the graphical interface, provided that the device and file system concerned allow this.

For administrative operations, it may be necessary to unmount a file system first.

In rescue operation, it must also be borne in mind that mounting a damaged file system or one that is to be backed up need not automatically be the first sensible step.

If a hardware defect is suspected, first check whether backing up the medium with GNU ddrescue is preferable.

### SMART data

With supported drives, GNOME Disks can display **SMART data**.

SMART stands for **Self-Monitoring, Analysis and Reporting Technology**.

GNOME describes SMART as a technology integrated in drives for monitoring possible problems. In GNOME Disks, **SMART Data & Self-Tests** can be used to view the overall assessment of the drive and to access individual SMART attributes.

GNOME Disks thus enables an initial graphical assessment of the condition of a supported drive.

### SMART self-tests

In addition to displaying existing SMART data, GNOME Disks can be used to start SMART self-tests.

For this, the GNOME documentation describes calling **SMART Data & Self-Tests** and then **Start Self-test**.

The result supplements the existing SMART attributes and the overall drive assessment.

However, SMART is no guarantee that a drive is free of errors or cannot fail at short notice.

A conspicuous condition should be taken as a reason to back up the data as quickly as possible, especially where important data is concerned.

### Distinction from A.8 – SMART diagnostics

In this chapter, the SMART function of GNOME Disks is only covered as part of graphical drive management.

JUMBO BLOCK Rescue also contains the separate menu item and manual section **A.8 – SMART diagnostics**.

There, the actual SMART diagnostics using the `smartmontools` tool set included in the baseline are documented separately and in considerably more detail.

The separation thus remains clear:

**GNOME Disks – graphical drive management including SMART display**

**SMART diagnostics – specialised drive diagnostics**

### Performance test

GNOME Disks has an integrated **drive benchmark**.

After the drive has been selected, the performance test can be started via the drive menu. Parameters for transfer rate and access time can be adjusted.

A pure read test determines the speed at which data is read from the medium.

On completion, GNOME Disks displays the measurement results graphically.

The results documented include:

minimum transfer rate,

maximum transfer rate,

average transfer rate,

and average access time.

The benchmark can thus be used for the technical assessment of drive performance.

### Write benchmark

The benchmark also has an option for a **write test**.

If this is enabled, GNOME Disks measures the write speed as well as the read speed.

This distinction is essential for a rescue system:

**In terms of safety, a read test and a write test are not equivalent.**

A write benchmark generates write accesses to the medium and should therefore not be used rashly on an original storage device with data to be rescued.

With a source medium that is damaged or must be kept unchanged for forensic or technical reasons, a write benchmark is not suitable as a non-binding diagnosis.

### Storage device images

GNOME Disks supports working with **storage device images**.

The GNOME project expressly names creating and using images of USB storage devices as a function of the application.

This allows data from a suitable storage device to be transferred to an image file, or existing images to be written back to a storage device.

The two directions must be strictly distinguished:

**Storage device → image file**

and

**Image file → storage device**

The second direction overwrites data on the target medium.

### Creating a storage device image

Creating a storage device image can be used to back up the content of a storage device in an image file.

For an intact storage device that can be read without problems, this can be a practical graphical function.

Such an image can then be kept separately from the original medium or examined further.

### Restoring a storage device image

GNOME Disks can write an existing storage device image back to a selected drive.

This operation is **destructive for the previous content of the target device**, because the target is written with the data of the image.

Before starting, at least three things must therefore be checked unambiguously:

the selected image file,

the selected target storage device,

and the fact that existing data on this target may be overwritten.

Confusing source and target storage devices can lead to considerable data loss.

### Distinction from GNU ddrescue

The image function of GNOME Disks must not be equated with GNU ddrescue.

GNOME Disks provides a convenient graphical way of working with storage device images.

**GNU ddrescue**, by contrast, is designed specifically for data recovery from media that are difficult to read; it works with a mapfile, documents areas already rescued or problem areas and enables further rescue passes based on this.

With a **physically conspicuous, unstable or faultily readable storage device**, GNU ddrescue is therefore the specialised tool for this within JUMBO BLOCK Rescue.

### Creating and managing partitions

GNOME Disks can manage partitions or volumes and partition storage devices. The GNOME project expressly names partitioning as a core function of the application.

Such operations change the structure of the storage device.

In rescue operation, new partitions should therefore not be created on a medium with lost data without a clear technical reason.

If a partition has merely disappeared or a partitioning structure is damaged, a specialised recovery tool such as TestDisk should be used first.

### Formatting

GNOME Disks can format partitions or suitable storage devices.

GNOME expressly describes the formatting function as a way of removing existing files from a removable storage device and setting up a file system for further use.

The following therefore applies to data recovery:

**A partition with data to be rescued must not be formatted just because the operating system no longer detects the existing file system correctly.**

Formatting creates new file system structures and thus changes the medium.

### Formatting is not secure erasure

GNOME expressly points out that normal formatting is **not a method of secure data erasure**.

After formatting, files or data may still be found with special recovery tools.

This property is relevant from a rescue perspective:

Accidental formatting does not necessarily mean that all data is irretrievably lost.

In this case, further write accesses should be avoided and suitable data recovery tools used.

### Changing the file system size

With suitable file systems, GNOME disk management supports changing the size of a file system and its partition.

The official GNOME documentation expressly lists **shrinking and growing a file system and its partition** as a function of disk management.

As with GParted, this is a structural change to the storage device.

Such an operation should not be used as the first repair attempt on a damaged original medium.

### Checking and repairing a file system

GNOME Disks, or GNOME disk management, offers functions for checking and repairing supported file systems.

The GNOME documentation expressly lists the task **“Repair a damaged filesystem”**.

A file system repair must be distinguished from a purely reading diagnosis.

Repair tools can change file system structures in order to restore a consistent state.

With important data, an image or other backup of the initial state should therefore be created first wherever possible.

### Powering off a drive

With supported devices, GNOME Disks can provide functions for safely stopping or powering off a drive.

This is particularly useful with externally connected storage media that are to be disconnected from the system in a controlled manner once work is complete.

Before a medium is physically removed, running write operations must be completed and file systems in use must be properly unmounted.

### GNOME Disks and GParted

GNOME Disks and GParted overlap in several tasks but are not identical.

**GNOME Disks** combines drive information, device management, SMART display, benchmarks, image functions and basic partition and file system management in one graphical interface.

**GParted** concentrates much more strongly on the graphical editing of partitions and file system structures.

Within JUMBO BLOCK Rescue, the two programs therefore complement each other.

GParted is available for extensive partition work, while GNOME Disks is particularly suitable for a quick drive overview and additional device-related functions.

### Typical rescue workflow

With an unknown storage device, GNOME Disks can first be used for identification and overview.

Initially, only the following should be examined:

Which storage device has been detected?

What capacity does it have?

Which partitions or volumes are displayed?

Which drive condition is reported?

Is SMART information available?

Only then should it be decided whether further diagnostic, backup or repair steps make sense.

With a physically conspicuous storage device, the priority should be on data backup.

With a lost partition, TestDisk should be considered.

GParted is available for extensive partition changes.

For specialised SMART diagnostics, the separate **SMART diagnostics** tool follows within JUMBO BLOCK Rescue.

### Limitations

GNOME Disks is not a specialised data recovery program.

Unlike TestDisk, the application does not reconstruct lost partitioning structures, nor does it carry out signature-based file carving like PhotoRec or Foremost.

With physically damaged or poorly readable media, the image function does not replace the specialised rescue mechanisms of GNU ddrescue.

SMART values, in turn, are a diagnostic aid but no guarantee of the current or future error-free condition of a storage device.

### Safety notes

GNOME Disks combines reading and writing functions in the same interface.

Before every operation, it must therefore be established whether it merely reads information or changes the storage device.

Particular care is required with:

formatting,

partitioning,

changes to file systems,

file system repairs,

restoring an image to a storage device,

and write benchmarks.

With a storage device containing data to be rescued, a backup of the initial state should be created before such operations wherever possible.

With a physically unstable medium, GNU ddrescue is intended for the backup.

**Source and target must be identified unambiguously before every writing operation.**

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, GNOME Disks is a versatile **graphical tool for managing storage devices and block devices**.

It adds device-related diagnostic and management functions to GParted and provides quick graphical access to drive information.

The functional classification is:

**GParted – detailed graphical partition management**

**Disks / GNOME Disks – drive overview, device management, SMART display, benchmarks, images and basic partition/file system management**

**SMART diagnostics – specialised SMART and drive diagnostics**

**GNU ddrescue – specialised backup of storage devices that are difficult to read**

**TestDisk – recovery of lost or damaged partitioning structures**

GNOME Disks is thus particularly suitable for the **initial technical inventory of a storage device**, but at the same time has powerful writing functions that may only be used in a controlled manner in rescue operation.

# A.8 SMART diagnostics

**Area:** Storage devices
**Program:** SMART diagnostics
**Technical basis:** smartmontools / `smartctl`
**Baseline 20261001:** smartmontools 7.4
**Installed package:** `smartmontools 7.4-3`
**Operation:** command-line-based drive diagnostics; integrated as the SMART diagnostics function in JUMBO BLOCK Rescue

## Purpose

SMART diagnostics is used for the technical condition analysis of supported hard disks and solid-state drives on the basis of the diagnostic information provided by the drive.

For this purpose, JUMBO BLOCK Rescue Baseline 20261001 contains the package `smartmontools` in version `7.4-3`.

Smartmontools provides in particular the programs `smartctl` and `smartd`. `smartctl` is used for directly querying and controlling the SMART functions of a drive, while `smartd` is intended for continuous monitoring.

For interactive rescue use, **`smartctl`** is the most relevant.

## What is SMART?

SMART stands for:

**Self-Monitoring, Analysis and Reporting Technology**

The technology is integrated in many modern storage devices and provides information on drive reliability and device condition.

`smartctl` can read this information and – depending on the device concerned – trigger internal self-tests of the drive.

SMART does not replace a data backup and cannot guarantee that a drive will not fail in future.

## Supported device classes

Smartmontools supports different storage protocols and device classes.

These include in particular:

**ATA / SATA**

**SCSI / SAS**

**NVMe**

Support also exists for other device and controller configurations.

Which SMART or diagnostic information is actually available depends on the drive, its firmware, the protocol used, the controller and, where applicable, an intervening USB or RAID interface.

### Identifying the device

Before any diagnosis, the correct device must be determined unambiguously.

Typical Linux device names can be, for example:

`/dev/sda`

`/dev/sdb`

Or, for NVMe:

`/dev/nvme0`

or

`/dev/nvme0n1`

These names are examples only.

The storage device actually to be examined must be identified beyond doubt before the call.

### Searching for devices automatically

`smartctl` has the following function for searching for supported devices:

`--scan`

Example:

```
sudo smartctl --scan
```

This outputs the detected devices including the determined device type or protocol.

The following variant is also available:

`--scan-open`

This variant also tries to open the devices found, which allows device detection to be made more specific.

### Displaying device information

The following options output basic information about the drive:

`-i`

or

`--info`

Example:

```
sudo smartctl -i /dev/sda
```

With an ATA device, this can include, among other things:

device model,

serial number,

firmware version,

capacity,

information on the ATA standard,

and information on whether SMART is supported and whether SMART is enabled.

This call is therefore suitable first of all for identifying the drive.

### SMART health status

The following options query the SMART health status:

```
-H
```

or

```
--health
```

Example:

```
sudo smartctl -H /dev/sda
```

This status is an overall assessment provided by the drive or its SMART implementation.

A positive result does not mean that all individual values are inconspicuous or that a future failure is ruled out.

The diagnosis should therefore not be reduced solely to this one status.

### Comprehensive SMART information

A central diagnostic call is:

```
sudo smartctl -a /dev/sda
```

The following option outputs a comprehensive collection of the available SMART information:

```
-a
```

or

```
--all
```

Which information appears depends on the device type and its capabilities.

In a rescue diagnosis, this call is often a suitable starting point for a more detailed examination.

### Extended full output

The following options request an even more comprehensive output of the available SMART and other device-related information:

`-x`

or

`--xall`

Example:

```
sudo smartctl -x /dev/sda
```

`-x` thus goes beyond the normal `-a` output.

This output can be valuable for a detailed technical analysis, but produces a correspondingly large amount of diagnostic information.

### SMART attributes with ATA devices

With ATA/SATA devices, the following options display the SMART attributes:

`-A`

or

`--attributes`

Example:

```
sudo smartctl -A /dev/sda
```

The attributes output come from the SMART implementation of the drive concerned.

Depending on the manufacturer and model, the meaning and encoding of individual raw values can vary.

`smartctl` uses, among other things, its drive database to interpret known models and attributes correctly.

### Normalised values and raw values

With conventional ATA SMART attributes, normalised values and thresholds can be present in addition to a raw value.

The raw values must not be interpreted across the board as physical units.

The smartmontools documentation expressly points out that the conversion of raw values into physical quantities is not specified by the SMART standard and that manufacturers can use different conventions.

An individual raw value should therefore always be assessed in the context of the drive model and the attribute concerned.

### Error logs

SMART-capable devices can provide error logs.

With suitable devices, these can be read with corresponding `-l` options.

A classic example is:

```
sudo smartctl -l error /dev/sda
```

The error log can contain information about errors registered by the drive.

However, the absence of a logged error does not prove that the medium is completely free of errors.

### Self-test log

The following option queries the log of self-tests carried out:

```
-l selftest
```

Example:

```
sudo smartctl -l selftest /dev/sda
```

Results of earlier internal drive self-tests can be listed there.

After a newly started self-test, its result can be checked later via this log.

### Internal SMART self-tests

With supported drives, `smartctl` can trigger internal self-tests.

The tests are carried out by the drive itself.

Which test types are available depends on the device and protocol concerned.

Short and longer or extended self-tests are particularly relevant for diagnosis.

### Short self-test

With a supported device, a short self-test can be started, for example, as follows:

```
sudo smartctl -t short /dev/sda
```

The test is carried out internally by the drive.

With supported devices, `smartctl` typically reports an expected test duration or the time at which the result should be available.

The self-test log can then be queried again.

### Long or extended self-test

With suitable devices, a more extensive self-test can be started as follows:

```
sudo smartctl -t long /dev/sda
```

This test usually takes considerably longer than a short self-test.

Here too, the actual check takes place within the drive.

The result can then be checked via the self-test log.

### Aborting a self-test

With supported devices, a running SMART self-test that is not being carried out in captive mode can be aborted with the following options:

`-X`

or

`--abort`

Example:

```
sudo smartctl -X /dev/sda
```

Whether and how a particular device responds to this depends on its capabilities.

### Captive mode

For certain ATA self-tests, there is what is known as **captive mode**.

The smartmontools documentation expressly warns that such a test can block the drive for the duration of the test.

Captive tests should therefore only be carried out on drives on which **no partitions are mounted**.

For the normal initial rescue diagnosis, there is generally no reason to force a captive test without a specific technical need.

### Enabling or disabling SMART

With suitable devices, the following option enables SMART:

`-s on`

The following option disables it:

```
-s off
```

Example:

```
sudo smartctl -s on /dev/sda
```

This is already a control function and not a mere diagnostic query.

In rescue operation, the device configuration should therefore not be changed unnecessarily.

For an initial examination, first establish whether SMART is supported and what state it is in.

### Specifying the device type

Normally, `smartctl` tries to detect the device type automatically.

The following options specify a device type explicitly:

```
-d TYPE
```

or

```
--device=TYPE
```

This function is relevant, for example, with certain USB adapters, RAID controllers or other intermediate layers.

However, it should not be used experimentally with arbitrary type specifications.

For individual device types, the smartmontools documentation expressly warns of possible undesirable side effects if used incorrectly.

### USB enclosures and adapters

With external SATA drives behind a USB adapter, SMART availability depends on whether the bridge chip used forwards the required commands correctly and is supported by smartmontools.

A drive can therefore physically support SMART while the data is not accessible, or not fully accessible, via a particular USB adapter.

A lack of SMART access must therefore not automatically be interpreted as a lack of SMART support in the actual drive.

### RAID controllers

RAID controllers can also change direct access to the underlying physical drives.

Depending on the controller, special device types or access methods are required.

With a RAID system, it must therefore first be clarified whether `smartctl` is addressing the logical volume or actually a physical drive.

Incorrect assignments can lead to misleading diagnostic results.

### NVMe and SMART

Smartmontools also supports NVMe devices.

However, NVMe diagnostic information does not simply correspond to the conventional ATA SMART attribute table.

`smartctl` displays the health and error information offered by the NVMe device in accordance with the NVMe protocol.

For JUMBO BLOCK Rescue, there is also **A.9 – NVMe diagnostics**, based on `nvme-cli`.

Two different diagnostic routes are thus available for NVMe drives:

SMART/health evaluation via smartmontools,

and native NVMe diagnostics via `nvme-cli`.

Native NVMe diagnostics are covered separately in the following chapter.

### Temperature

Many drives provide temperature information.

Depending on the device, this can be output via SMART attributes or other device-specific SMART/health structures.

An elevated temperature can be diagnostically relevant.

However, interpretation must take account of the manufacturer-specific limits and the design of the device concerned.

A single temperature value alone is not a complete condition diagnosis.

### SMART is not a surface scan

SMART diagnostics is not to be equated with a complete sector-by-sector read test of the storage device.

SMART supplies health, error and operating information provided by the drive itself and enables internal self-tests.

A successful SMART query therefore does not automatically mean that every logical block of the storage device can be read without problems.

### SMART is not a data recovery program

`smartctl` does not restore deleted files.

It does not reconstruct lost partitions.

Nor does it create a rescue clone of a damaged storage device.

Its task is the **diagnosis and control of the SMART and related monitoring functions provided by the drive**.

## SMART diagnostics and GNU ddrescue

SMART diagnostics and GNU ddrescue complement each other.

SMART can provide indications that a drive is technically conspicuous.

GNU ddrescue can then be used to transfer as much still readable data as possible from a problematic medium to another medium or into an image.

With a storage device that is already clearly conspicuous, you should not try to force a complete fault analysis through numerous diagnostic and stress tests before the important data has been backed up.

If data preservation has priority, backing up the still readable content is decisive.

## SMART diagnostics and GNOME Disks

GNOME Disks can display SMART data graphically and start SMART self-tests.

The **SMART diagnostics** menu item within JUMBO BLOCK Rescue, by contrast, is based on the specialised smartmontools tool set.

Two access levels are thus available:

**GNOME Disks – graphical overview and basic SMART functions**

**smartctl – detailed command-line-based SMART diagnostics**

Both draw on diagnostic information provided by the drive but differ in operation and level of detail.

## JSON output

Smartctl supports structured output.

The following options output diagnostic information in JSON format:

`-j`

or

`--json`

Example:

```
sudo smartctl -a -j /dev/sda
```

This is particularly useful for automated evaluations or for further processing of diagnostic information.

For a manual rescue examination, the normal text output is usually easier to read directly.

### Drive database

Smartmontools uses a drive database to take account of known device models and manufacturer-specific features.

With known ATA drives, this allows model families to be recognised, for example, and suitable defaults to be used for interpreting certain attributes.

This database improves the interpretation of device-specific SMART information.

### Limitations of SMART diagnostics

SMART can provide valuable indications of the condition of a storage device but has clear limitations.

Not every drive provides the same values.

Not every manufacturer interprets raw values identically.

USB adapters or RAID controllers can restrict access to diagnostic information.

A drive can have problems or fail later despite inconspicuous SMART values.

And a drive that is already badly damaged should not be subjected to unnecessary load from extensive tests if important data has to be rescued first.

### Safety notes

Before any diagnosis, the correct physical drive must be identified unambiguously.

Pure information queries such as `-i`, `-H`, `-A`, `-a` or `-x` must be distinguished from commands that control drive functions or trigger self-tests.

With an unstable or already failing storage device, it must be weighed up whether an additional self-test makes sense or whether data recovery should be carried out first.

A long self-test in particular can keep the drive busy for an extended period.

With mechanically conspicuous drives, severe read errors or repeated connection drop-outs, backing up important data should have priority.

### Typical rescue workflow

A controlled workflow is advisable for an initial SMART examination:

First, the storage device is identified unambiguously.

Device information and existing SMART data are then read.

For example:

```
sudo smartctl -i /dev/sdX
```

Then:

```
sudo smartctl -H /dev/sdX
```

And for a more comprehensive diagnosis:

```
sudo smartctl -a /dev/sdX
```

If required, the following can also be used:

```
sudo smartctl -x /dev/sdX
```

Only after reviewing this information should it be decided whether an internal self-test makes sense.

If there are signs of a serious hardware problem and important data to be rescued, backing up the storage device should be considered first instead.

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, SMART diagnostics forms the specialised level for the **condition analysis of SMART-capable storage devices**.

The tools in the storage devices area thus complement each other functionally:

**GParted – graphically manage partitions and partition tables**

**Disks / GNOME Disks – graphical drive overview, device management, SMART display, benchmarks and images**

**SMART diagnostics / smartctl – detailed SMART and drive diagnostics**

**NVMe diagnostics / nvme-cli – native diagnostics and management of NVMe devices**

In the rescue workflow, SMART diagnostics can therefore help in particular to decide whether a storage device appears technically inconspicuous or shows indications of a problematic condition.

However, the SMART evaluation is always a **diagnostic aid and no guarantee against failure**.

With data at risk, backing up the medium as early as possible remains the decisive step.

# A.9 NVMe diagnostics

**Area:** Storage devices
**Program:** NVMe diagnostics
**Technical basis:** `nvme-cli`
**Baseline 20261001:** nvme-cli 2.13
**Installed package:** `nvme-cli 2.13-2`
**Operation:** command line

## Purpose

NVMe diagnostics is used for the direct examination of **NVMe storage devices** via the management and diagnostic functions provided by the NVMe protocol.

For this purpose, JUMBO BLOCK Rescue Baseline 20261001 contains:

`nvme-cli 2.13-2`

The central command is:

`nvme`

`nvme-cli` can be used to identify NVMe controllers and namespaces, read device information, query SMART/health information, examine error logs and – where supported by the device – run internal self-tests.

The tool also has numerous administrative functions that can change NVMe devices.

In rescue operation, a strict distinction must therefore be made between **pure diagnosis** and **modifying NVMe commands**.

## NVMe

NVMe stands for:

**Non-Volatile Memory Express**

NVMe is an interface and command specification for non-volatile storage devices.

Under Linux, NVMe devices typically appear with device names such as:

`/dev/nvme0`

Associated namespaces appear, for example, as:

`/dev/nvme0n1`

These names are examples only.

Before any diagnosis, the device actually to be examined must be identified unambiguously.

## Controller and namespace

With NVMe, a distinction must be made between **controller** and **namespace**.

A controller can provide one or more namespaces.

The following device name typically designates the NVMe controller:

`/dev/nvme0`

The following device name designates a namespace of this controller:

`/dev/nvme0n1`

This distinction is essential for the correct use of `nvme-cli`.

Some commands accept both an NVMe character device such as `/dev/nvme0` and a namespace block device such as `/dev/nvme0n1`. Which form makes sense or is required depends on the command concerned.

### Listing detected NVMe devices

A central first command is:

```
sudo nvme list
```

It is used to list the NVMe devices detected by the system.

This makes it possible first to establish which NVMe storage devices or namespaces are present.

Before any further operations, this information should be used to ensure that the correct device has been selected.

### Identifying the controller

For detailed information about an NVMe controller, nvme-cli 2.13 provides the following command:

```
nvme id-ctrl
```

Example:

```
sudo nvme id-ctrl /dev/nvme0
```

The command sends an **Identify Controller** command to the NVMe device and outputs the controller data structure returned.

This contains numerous technical properties of the controller.

Depending on the device, these can include identification, firmware and capability information, for example.

For a rescue examination, this is an important source for the technical identification of the NVMe device.

### Identifying the namespace

The information of an NVMe namespace can be retrieved with the following command:

`nvme id-ns`

Example:

`sudo nvme id-ns /dev/nvme0n1`

This outputs the **Identify Namespace** data structure provided by the NVMe device.

Controller and namespace information serve different purposes and should therefore not be confused with each other.

### SMART/health information

NVMe provides its own **SMART / Health Information Log Page**.

In nvme-cli 2.13, it is retrieved with the following command:

`nvme smart-log`

Example:

`sudo nvme smart-log /dev/nvme0`

The command requests the NVMe SMART log page and outputs the structure returned.

In version 2.13, an NVMe controller device can be specified, such as:

`/dev/nvme0`

A namespace block device can also be specified, such as:

`/dev/nvme0n1`

### Global and namespace-specific SMART data

`nvme smart-log` supports the following option:

`--namespace-id`

or

`-n`.

This allows a specific namespace ID to be requested.

Without such a specification, nvme-cli 2.13 uses the namespace ID `0xffffffff` by default for the global SMART query.

Whether a device actually provides SMART/health information on a per-namespace basis depends on the capabilities of the controller concerned.

### Typical health information

The SMART/health log page of an NVMe device contains standardised health and operating information.

Depending on the device and implementation, information on the following areas, among others, can be output:

critical warnings,

temperature,

available spare capacity,

lifetime used,

data volumes read and written,

power-on hours,

power cycles,

unsafe shutdowns,

media and data integrity errors,

and error log entries.

These values are provided by the NVMe device or controller.

They are diagnostic information and no guarantee that the drive is free of errors or cannot fail at short notice.

### Critical Warning

The **Critical Warning** field of the NVMe SMART/health information deserves particular attention.

It is used to signal certain critical states detected by the controller.

A set warning bit should not be considered in isolation but evaluated together with the remaining health information and the technical situation of the storage device.

With a drive containing important data and indications of a critical state, data backup should be given priority.

### Temperature

NVMe controllers can provide temperature information via the SMART/health log page.

An unusual or critical temperature can be diagnostically relevant.

However, the temperature alone does not allow a complete assessment of the drive's condition.

It must be considered together with the remaining health data and the warning states reported by the device.

### Available Spare

NVMe can provide information on the spare capacity of the non-volatile memory still available.

This includes, among other things, values for the available spare and the associated threshold.

A critical state of the spare can be signalled via the health information.

### Percentage Used

The NVMe health log contains the **Percentage Used** field.

This describes an estimate provided by the controller of the lifetime or endurance of the NVM subsystem already used.

This value must not be confused with the percentage of storage space occupied in a file system.

It relates to the use or wear of the storage medium as assessed by the controller.

### Data Units Read and Data Units Written

NVMe can provide counters for the data volumes read and written by the host.

These values allow the previous use of the drive to be assessed.

They are operating counters and not directly an error indicator.

### Power Cycles and Power On Hours

NVMe devices can report, among other things, the number of power cycles and power-on hours.

These values provide additional information about the previous use of the device.

Here too, the following applies:

A high power-on hours counter alone does not prove a defect.

It must be interpreted in connection with the remaining health and error data.

### Unsafe Shutdowns

The SMART/health information can contain a counter for **Unsafe Shutdowns**.

This documents corresponding shutdowns recorded by the NVMe subsystem.

When a system is examined, the value can provide indications of its operating history.

On its own, however, it does not yet constitute a complete fault diagnosis.

### Media and Data Integrity Errors

**Media and Data Integrity Errors** reported by the NVMe device are particularly relevant to a rescue case.

A corresponding counter indicates that the device has registered such errors.

If important data is involved at the same time, this should be treated as a serious diagnostic indication, and further load on the drive should be weighed against the need to back up the data as quickly as possible.

### Error log

NVMe has its own error log.

In nvme-cli 2.13, it can be retrieved with the following command:

```
nvme error-log
```

Example:

```
sudo nvme error-log /dev/nvme0
```

The command requests the NVMe **Error Information Log Page** and outputs the corresponding entries.

The error log can contain additional information on NVMe errors registered by the controller.

### Number of error log entries

In version 2.13, `nvme error-log` supports specifying the number of entries to be retrieved.

The following options are available for this:

```
--log-entries
```

or

```
-e
```

Example:

```
sudo nvme error-log /dev/nvme0 -e 16
```

This allows several entries of the NVMe error log to be requested specifically.

### SMART log in JSON format

nvme-cli 2.13 supports different output formats for `smart-log`.

The following options request structured JSON output:

`-o json`

or

`--output-format=json`

Example:

`sudo nvme smart-log /dev/nvme0 -o json`

This is particularly suitable for automated evaluations and machine processing of diagnostic information.

In version 2.13, `smart-log` also supports normal and binary output.

### Outputting raw data

The following options switch to raw output:

`--raw-binary`

or

`-b`

`nvme smart-log` then outputs the raw SMART log data structure.

For this mode of operation, the official documentation recommends, in essence, redirecting the binary output to a file or a suitable recipient.

The human-readable output is much more suitable for normal manual diagnosis.

### Device Self-Test

NVMe defines a **Device Self-Test**.

nvme-cli 2.13 provides the following command for this:

`nvme device-self-test`

Example:

`sudo nvme device-self-test /dev/nvme0 -s 1`

The command triggers a test carried out by the NVMe device itself.

Whether this function is available depends on the capabilities of the device concerned.

### Short self-test

In nvme-cli 2.13, the following self-test code starts a **Short Device Self-Test**:

`1`

Example:

```
sudo nvme device-self-test /dev/nvme0 -s 1
```

If required, a namespace can be specified with `-n` or `--namespace-id`.

### Extended self-test

The following self-test code starts an **Extended Device Self-Test**:

```
2
```

Example:

```
sudo nvme device-self-test /dev/nvme0 -s 2
```

An extended self-test can take considerably longer and keep the device busy for correspondingly longer.

With a drive that is already unstable and has important data to be rescued, it should therefore first be checked whether carrying out such a test really makes sense or whether data backup must take priority.

### Self-test status

The following self-test code queries the current state of a device self-test operation:

```
0
```

This makes it possible to establish whether a corresponding self-test is running and what state it is in.

### Aborting a self-test

The following self-test code aborts a running device self-test operation:

```
0xf
```

Alternatively, `fh` can be used.

Example:

```
sudo nvme device-self-test /dev/nvme0 -s 0xf
```

### Waiting for the self-test

For the device self-test, nvme-cli 2.13 also supports the following option:

```
--wait
```

or

```
-w.
```

With this option, the program waits for the device self-test to finish before it exits.

Without this option, the test can be started while the result is queried separately later.

### Self-test log

In nvme-cli 2.13, the results of device self-tests can be retrieved with the following command:

```
nvme self-test-log
```

Example:

```
sudo nvme self-test-log /dev/nvme0
```

This allows the information provided by the device on self-tests carried out to be examined.

### Distinction from SMART diagnostics

A.8 and A.9 partly overlap, because smartmontools also supports NVMe devices.

However, the tools work at different levels.

**SMART diagnostics / smartctl**

This provides a uniform diagnostic interface for various drive types such as ATA, SCSI and NVMe.

**NVMe diagnostics / nvme-cli**

This accesses the NVMe-specific management, identify, log and diagnostic functions directly.

For a targeted NVMe examination, `nvme-cli` therefore offers information and commands that go beyond a general SMART query.

### Diagnosis versus administration

`nvme-cli` is not exclusively a diagnostic program.

The tool also contains administrative and potentially destructive commands.

These include, for example, functions for:

formatting,

namespace creation and deletion,

namespace attachment,

firmware download and activation,

write operations,

Write Zeroes,

Data Set Management,

feature changes,

controller reset,

and other NVMe administration commands.

These functions are part of the scope of nvme-cli but are **not part of a non-binding rescue diagnosis**.

### Format

Particular care is required with the following command:

`nvme format`

It carries out an NVMe format operation.

Depending on the options used and the device capabilities, such an operation can change or erase the user data of a namespace.

**`nvme format` must therefore never be treated as a diagnostic command.**

On an original storage device with data to be rescued, such an operation is not permissible without a clear technical intention.

### Namespace management

Among other things, nvme-cli has commands for creating, deleting, attaching and detaching namespaces.

These functions change the logical NVMe configuration.

They are administrative tools and are not required for a pure condition diagnosis.

In rescue operation, they must not be used experimentally on a storage device with important data.

### Write commands

nvme-cli contains direct I/O commands such as:

`nvme write`

and

`nvme write-zeroes`.

Such commands write to the NVMe medium.

They must therefore be fundamentally distinguished from purely reading diagnostic commands.

With an original medium containing data to be rescued, such commands must not be run without an explicit technical need.

### Firmware functions

nvme-cli also supports NVMe firmware operations.

These include functions for downloading and activating or committing firmware.

Firmware operations change the operating state of a device and can cause additional risks.

They are not part of a normal initial rescue diagnosis.

### Controller reset

An NVMe controller reset is also possible with nvme-cli.

Although a reset does not change the stored user data with the same aim as formatting, it actively intervenes in the operating state of the controller.

Such a command should therefore not be equated with a pure information query.

### Safe initial diagnosis

Reading or diagnostic commands are particularly suitable for an initial examination of an NVMe device.

A typical workflow is as follows. First, identify the existing NVMe devices:

```
sudo nvme list
```

Then identify the controller, for example:

```
sudo nvme id-ctrl /dev/nvme0
```

After that, retrieve the health information:

```
sudo nvme smart-log /dev/nvme0
```

And retrieve the error log:

```
sudo nvme error-log /dev/nvme0
```

Only after evaluating this information should it be decided whether a device self-test makes sense.

### Handling conspicuous NVMe drives

If an NVMe device shows critical warnings, media errors, repeated I/O errors, connection problems or other clear abnormalities, the purpose of further work must be taken into account.

If important data is to be rescued, extensive diagnosis is not necessarily the highest priority.

Every additional test puts strain on the device.

With an unstable medium, it may make more sense to back up the still readable data first.

### NVMe diagnostics and GNU ddrescue

GNU ddrescue and nvme-cli perform different tasks.

**nvme-cli** examines NVMe-specific device information, health data, error logs and other controller functions.

**GNU ddrescue** tries to transfer readable data from a problematic block device to another medium or into an image.

nvme-cli thus diagnoses the condition.

ddrescue is used for the actual block-by-block data backup or data recovery.

### Limitations

NVMe diagnostics cannot reconstruct deleted files.

It does not restore lost partitions.

It does not replace a rescue clone.

And even an inconspicuous SMART/health log does not guarantee that all data on the drive is readable or that the device will not fail in future.

Conversely, a single high operating counter does not automatically have to mean a defect.

The diagnostic information must be assessed in context.

### Safety notes

Before every `nvme` command, the target device must be identified unambiguously.

It is particularly important to distinguish between the controller device:

`/dev/nvmeX`

and the namespace device:

`/dev/nvmeXnY`

In rescue work, before running any unknown command, you should also check whether it merely reads information or changes the controller or the medium.

Format, namespace, firmware, write and other administrative commands are particularly critical.

With a damaged original medium, the following applies:

**Diagnosis must not unnecessarily endanger the rescue of the data still available.**

## Version note

This chapter expressly documents **nvme-cli 2.13**, in accordance with JUMBO BLOCK Rescue Baseline 20261001.

The command forms used in this chapter correspond to the documentation of nvme-cli 2.13. They include:

`nvme id-ctrl`

`nvme smart-log`

`nvme error-log`

and

`nvme self-test-log`

Later nvme-cli versions have changed parts of the command structure.

These later changes are deliberately **not applied retroactively to Baseline 20261001**.

## Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, NVMe diagnostics forms the specialised diagnostic level for **NVMe storage devices**.

The four tools in the storage devices area thus complement each other:

**GParted – graphical partition and file system management**

**Disks / GNOME Disks – graphical device overview, storage device management, images, benchmarks and basic diagnostics**

**SMART diagnostics / smartctl – cross-protocol drive diagnostics for ATA, SCSI and NVMe, among others**

**NVMe diagnostics / nvme-cli – native NVMe controller, namespace, health, error and self-test diagnostics**

For NVMe systems, `nvme-cli` thus provides the most detailed direct interface to the diagnostic information provided by the NVMe protocol.

At the same time, it is a powerful administration tool.

Precisely for this reason, the strict separation between **reading diagnosis and modifying commands** is mandatory in rescue use.

# A.10 ClamAV Scan

**Area:** Security
**Program:** ClamAV Scan
**Technical basis:** ClamAV / `clamscan`
**Baseline 20261001:** ClamAV 1.4.3
**Installed core packages:** `clamav 1.4.3+dfsg-1`, `clamav-base 1.4.3+dfsg-1`, `libclamav12 1.4.3+dfsg-1`
**Operation:** command-line-based malware scanner

## Purpose

Within JUMBO BLOCK Rescue, ClamAV Scan is used to examine files and directories for malware and other content that can be identified by the loaded ClamAV signatures or detection mechanisms.

JUMBO BLOCK Rescue Baseline 20261001 contains ClamAV in version 1.4.3.

The following command is particularly relevant for an immediate, one-off scan:

`clamscan`

`clamscan` is a standalone command-line scanner. It does not require a running `clamd` daemon. On each call, a scan engine is created, the existing virus database is loaded, the specified target is examined, a scan report is output and the process then exits.

This way of working is particularly suitable for a rescue system in which storage devices, directories or individual files are to be examined selectively.

## Signature database

ClamAV needs an existing signature database for the scan engine to work.

By default, `clamscan` looks for the databases in the location where the ClamAV signatures were installed or updated by `freshclam`.

Two functions must therefore be distinguished:

**ClamAV Scan / `clamscan`** examines files.

**ClamAV signatures / `freshclam`** updates the official ClamAV signature databases.

Signature management is therefore documented separately in **A.11 – ClamAV signatures**.

## Basic syntax

The basic syntax is:

```
clamscan [Optionen] [Datei/Verzeichnis/-]
```

This allows individual files, directories or other suitable inputs to be examined.

### Scanning a single file

A single file can be specified directly.

Example:

```
clamscan /pfad/zur/datei
```

ClamAV examines the specified file and then outputs the result.

This is particularly suitable when a suspicious file is to be checked specifically.

### Scanning a directory

A directory can also be specified directly.

Example:

```
clamscan /pfad/zum/verzeichnis
```

Without an additional recursion option, `clamscan` does not automatically examine all subdirectories.

For a complete directory tree, recursive search must be enabled explicitly.

### Recursive scan

The following options search subdirectories recursively:

```
--recursive
```

or

```
-r
```

Example:

```
clamscan -r /pfad/zum/verzeichnis
```

Or:

```
clamscan --recursive /pfad/zum/verzeichnis
```

This is a central function for examining a mounted third-party storage device.

If a file system to be examined is mounted under a suitable mount point, for example, its directory tree can be scanned recursively in a targeted manner.

### Scanning the entire system

Technically, `clamscan` can also be applied to the root directory.

Example:

```
clamscan --recursive /
```

This triggers a very extensive scan.

ClamAV expressly points out that a full system scan can take a long time.

ClamAV does not have a special “quick scan” mode. The user determines which files and directories are examined.

For JUMBO BLOCK Rescue, a targeted scan of the storage device or directory that is actually relevant is therefore often more expedient than an unselective scan of the entire running rescue system.

### Scanning a mounted storage device

A typical rescue scenario is to mount a third-party file system and then examine its mount point.

The exact path depends on the mount point actually used.

The basic principle is:

```
clamscan -r /pfad/zum/mountpunkt
```

Before the scan, it must be ensured that the intended storage device is actually being examined.

The name of the mount point must not be guessed.

### Detection reports

If ClamAV identifies a file as malicious or suspicious on the basis of the loaded detection data, this is reported in the scan result.

Such a report must then be assessed.

ClamAV itself points out that **false positives** are possible.

A scan detection therefore does not mean that a file should be deleted immediately without further checking.

### Scan summary

On completion of a `clamscan` run, a summary is output.

This is used to assess the course and result of the scan.

With extensive rescue examinations, it can be useful to save a scan log to a file as well.

### Writing the scan report to a file

The following option saves the scan report to a file:

```
--log=FILE
```

Example:

```
clamscan -r --log=/pfad/scan.log /pfad/zum/scan
```

This is particularly useful for rescue and analysis work, because the result can then be documented and evaluated separately.

The location of the log should be chosen deliberately.

### Displaying infected files only

`clamscan` supports the following option:

`--infected`

or

`-i`.

This reduces the normal output so that only infected files are output.

Example:

```
clamscan -r -i /pfad/zum/scan
```

With very large data sets, this can improve clarity.

### Quiet mode

The following option reduces the output further:

`--quiet`

ClamAV documents this option as outputting only error messages.

This mode is not always useful for interactive analysis but can be useful in automated workflows or for targeted logging.

### Audible detection alert

With the following option, ClamAV can sound an audible signal when a virus is detected:

`--bell`

This function does not change the scan itself but merely serves as a notification during an interactive scan.

### Alternative signature database

The following option explicitly loads a specific signature file or a directory containing supported ClamAV databases:

`--database=FILE/DIR`

This allows `clamscan` to be operated with a specifically selected signature base if required.

For normal JUMBO BLOCK Rescue use, the database installed by default or maintained with `freshclam` is the intended starting point.

### Official signatures only

`clamscan` supports the following option:

`--official-db-only`

This makes it possible to specify that only official ClamAV signatures are loaded.

This is particularly relevant if additional signature files were present alongside the official databases.

### PUA detection

ClamAV supports the detection of what are known as **Potentially Unwanted Applications (PUA)**.

The following option is available for this with `clamscan`:

`--detect-pua`

In addition, PUA categories can be included or excluded with the following options:

`--include-pua=KATEGORIE`

or

`--exclude-pua=KATEGORIE`

PUA must be distinguished from a clear malware classification. The term covers software that is classified as potentially unwanted because of certain properties or behaviour.

A corresponding report must therefore be assessed in the relevant context.

### Archives and containers

The ClamAV scan engine can analyse supported container and archive formats.

With such files, it must be borne in mind that the amount of data actually to be examined can be considerably larger than the outer archive file.

ClamAV therefore has size and scan limits that restrict the processing of such content.

### Maximum file size

The following option specifies the maximum size of a file that ClamAV scans:

`--max-filesize=GRÖSSE`

Files above this limit are skipped and assumed to be clean by `clamscan`.

This is important for interpreting a scan result:

**“Not reported as infected” is not automatically the same as “completely examined”.**

Scan limits must be taken into account in a security-relevant assessment.

### Maximum scan size per container

The following option limits the maximum amount of data examined within a container file:

`--max-scansize=GRÖSSE`

This affects, for example, the processing of complex or large container structures.

Here too, a set limit can mean that not all data theoretically contained is examined.

### Using a file list

The following option specifies a file containing the paths to be examined:

`--file-list=DATEI`

This enables targeted scan lists without specifying each path individually as a command-line argument.

### File system boundaries

For recursive scans, `clamscan` supports the following options:

`--cross-fs=yes`

or

`--cross-fs=no`.

These control whether the scan switches to other file systems when traversing a directory tree recursively.

`--cross-fs=yes` is enabled by default.

This option is particularly relevant for a rescue system.

If a directory tree is examined below which further file systems are mounted, the scan can otherwise extend beyond the storage device originally intended.

For targeted examinations, it should therefore be decided deliberately whether file system boundaries are to be crossed.

### Moving infected files

With the following option, ClamAV can move files that trigger a detection to a specified directory:

```
--move=VERZEICHNIS
```

Example principle:

```
clamscan -r --move=/pfad/quarantaene /pfad/zum/scan
```

This function changes the data being examined.

It is therefore not to be equated with a purely reading diagnosis.

With a third-party original storage device or in an evidence preservation/rescue scenario, such a change should only be made deliberately.

### Copying infected files

The following option copies files with a detection report to a specified directory:

```
--copy=VERZEICHNIS
```

Unlike with `--move`, the original file remains in its original location.

Here too, it must be borne in mind that potentially malicious files are being handled.

The target directory should be treated in a correspondingly controlled manner.

### Deleting files automatically

ClamAV has the following option for deleting files that trigger an alarm during a scan:

```
--remove
```

This option is particularly critical for rescue use.

The official ClamAV documentation expressly warns against using this function generally, because **false positives can occur**.

Instead, ClamAV rather recommends moving or copying files and checking a detection afterwards.

The following therefore applies to JUMBO BLOCK Rescue:

**Automatic deletion of detected files is not a suitable standard procedure.**

The diagnosis should first be preserved and the detection assessed.

### Heuristic alerts

ClamAV supports heuristic detection mechanisms.

The following options enable or disable heuristic alerts:

```
--heuristic-alerts=yes
```

or

```
--heuristic-alerts=no
```

Heuristic alerts are enabled by default.

A heuristic report is not necessarily based on the same kind of clear signature match as a conventional signature detection.

Such results must therefore also be assessed appropriately.

### Encrypted archives and documents

The following option configures ClamAV to report encrypted archives or documents:

```
--alert-encrypted
```

Encrypted content can be problematic for a scanner, because its actual content cannot be fully analysed without decryption.

However, a corresponding report does not automatically mean that the file contains malware.

It initially indicates content that is encrypted and therefore cannot be examined normally.

### Temporary files

The following option prevents temporary files created by ClamAV during processing from being removed afterwards:

```
--leave-temps
```

This function is primarily relevant for special analysis and debugging purposes.

In normal scanning, there is generally no reason to leave temporary scan artefacts behind permanently.

### clamscan and clamdscan

ClamAV provides several ways of scanning.

`clamscan` is a standalone scanner.

Each time it starts, it loads the signature database into a newly created scan engine and does not require a running `clamd`.

`clamdscan`, by contrast, is a client for the ClamAV daemon `clamd`, which must already be running.

The daemon keeps its scan engine and database in memory and can accept scan jobs from clients.

For a targeted one-off scan in a rescue system, `clamscan` is particularly straightforward, because no previously configured and running scan daemon is required.

### Not a complete endpoint security system

ClamAV describes itself as a **malware detection toolkit** and not as a complete endpoint security suite.

In particular, a manual `clamscan` does not provide permanent real-time protection.

ClamAV does have separate on-access functions for Linux, but these work via `clamd` and `clamonacc` and must be distinguished functionally from a one-off `clamscan`.

A.10 therefore expressly documents the targeted rescue scan and not the setting up of a permanent endpoint protection system.

### Scanning a Windows storage device

As `clamscan` examines files and directories, a file system of another operating system mounted under Linux can also be used as a scan target, provided that Linux can access its files.

This is particularly useful for a rescue system.

A Windows system does not itself have to be started in order to examine its accessible files with ClamAV from JUMBO BLOCK Rescue.

A possibly compromised installed operating system can thus be examined offline.

### Advantage of an externally booted rescue system

If JUMBO BLOCK Rescue is started independently of the installed target operating system, the scan does not run within this possibly compromised operating system.

Instead, the files of the target system can be examined from the rescue environment.

However, this does not automatically prevent every conceivable manipulation or detection problem and does not replace a complete forensic examination.

But it offers a clear practical advantage:

The programs of the operating system being examined do not have to be run for a file-based ClamAV scan.

### A detection is not proof

A ClamAV detection is a technical scan result.

It should be documented and then assessed.

False positives are possible.

Likewise, a scan without detections does not mean that a system is free of malware with absolute certainty.

Detection depends, among other things, on the signatures used, the scan engine, the scan options, file access rights, supported formats, size limits and the fundamental detectability of a threat.

### Access rights

ClamAV can only examine content that the scan process can access.

Missing read permissions can therefore mean that files are not examined.

When evaluating a scan report, error messages and files that were not examined must be taken into account.

A scan must not be judged solely on the number of detections reported.

### Scanning and damaged storage devices

ClamAV is not a tool for physical data recovery.

With a drive that has read errors or other hardware problems, you should not start by forcing an extensive malware scan.

A full scan can read large parts of a storage device and put a corresponding strain on it.

If important data has to be rescued from an unstable medium, backing up the still readable content – for example with GNU ddrescue – takes priority.

A created image or its accessible content can then be examined.

### ClamAV and data recovery

The tasks must be clearly separated:

**GNU ddrescue** backs up as many readable blocks of a problematic medium as possible.

**TestDisk** examines and reconstructs partitioning or certain file system structures.

**PhotoRec and Foremost** search for files on the basis of their contents or signatures.

**ClamAV Scan** examines accessible files for malicious or suspicious content that is known or can be detected by the ClamAV engine.

ClamAV therefore does not replace a data recovery tool.

### Typical rescue workflow

With a storage device that is technically intact but suspicious from a security perspective, a controlled workflow is advisable.

First, the correct file system is identified and made accessible under controlled conditions.

Then check whether the existing ClamAV signature database is suitable or up to date enough for the intended scan.

The relevant directory tree can then be examined recursively, for example:

```
clamscan -r /pfad/zum/ziel
```

For clearer output of detections:

```
clamscan -r -i /pfad/zum/ziel
```

For a documented examination, a log can also be written:

```
clamscan -r -i --log=/pfad/scan.log /pfad/zum/ziel
```

Files found should be assessed first.

Automatic deletion is not a suitable standard procedure.

### Safety notes

Before a scan, the correct target must be identified unambiguously.

With original media containing important data, an unchanged or controlled backed-up initial state should be preserved first wherever possible.

The following options change the data being examined:

```
--move
```

and, in particular:

```
--remove
```

They must therefore be distinguished from a purely diagnostic scan.

The absence of a detection is no guarantee of freedom from malware.

Conversely, a detection is not automatically proof of an actually active infection.

Scan errors, access problems and scan limits must be taken into account when interpreting the result.

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, ClamAV Scan forms the first specialised level for **file-based malware examination**.

Its particular strength in the rescue context is that an installed target operating system does not have to be started.

An accessible file system can be examined from within the independent rescue environment.

The functional classification is:

**ClamAV Scan / `clamscan` – one-off malware examination of files and directories**

**ClamAV signatures / `freshclam` – updating the ClamAV detection databases**

**Chkrootkit – examination for indications of certain rootkits and suspicious system characteristics**

**Rkhunter – additional host-based security and rootkit checks**

**GUFW Firewall – graphical management of the host-based firewall**

ClamAV Scan is therefore not a repair tool and not a complete endpoint security system.

Its task within JUMBO BLOCK Rescue is the **controlled examination of accessible files for threats detectable by ClamAV**, preferably without having to start the possibly compromised target operating system itself.

# A.11 ClamAV signatures

**Area:** Security
**Program:** ClamAV signatures
**Technical basis:** ClamAV / `freshclam`
**Baseline 20261001:** ClamAV 1.4.3
**Installed packages:** `clamav-freshclam 1.4.3+dfsg-1`, `clamav-base 1.4.3+dfsg-1`, `clamav 1.4.3+dfsg-1`
**Operation:** command-line-based updating of the malware signature databases

## Purpose

Within JUMBO BLOCK Rescue, ClamAV signatures is used to maintain the detection databases used by the ClamAV scanner for malware examination.

The central program for this is:

`freshclam`

FreshClam is the official automatic database update tool of ClamAV.

It downloads the official ClamAV signature databases and updates databases that already exist.

The tasks of A.10 and A.11 are thus clearly separated:

**A.10 – ClamAV Scan / `clamscan`**
examines files and directories.

**A.11 – ClamAV signatures / `freshclam`**
provides and updates the official detection databases required for this.

## Significance of the signature databases

Before the ClamAV scan engine can be used with `clamscan` or `clamd`, ClamAV virus databases must be present in the location intended for them.

These databases form an essential part of the scanner's detection basis.

The version of the installed ClamAV software and how up to date the signature databases are, are two different things.

An installed ClamAV package as shown below therefore does not automatically mean that the existing signature databases are also up to date:

```
ClamAV 1.4.3
```

Both must be taken into account when assessing a malware scan.

### FreshClam

FreshClam is basically called with the following command:

```
freshclam
```

A typical manual call is:

```
sudo freshclam
```

FreshClam requires a valid configuration file `freshclam.conf`.

The exact location of this file depends on the installation and distribution.

ClamAV expressly points out that packaged installations may already prepare the configuration in a distribution-specific way.

### Update process

On a normal call, FreshClam tries to establish a connection to the ClamAV distribution network for the virus signature databases.

If no databases exist yet in the configured database directory, FreshClam downloads the required databases in full.

If databases are already present, FreshClam tries to update them.

This means that, during an update, the complete database does not generally have to be downloaded again every time.

### Incremental update

FreshClam supports what are known as **CDIFF updates**.

Differences between the existing and the current database version are transferred and applied to the local database.

ClamAV expressly describes FreshClam as a tool that, instead of the complete CVD file, can transfer only the differences between the existing and the current database with each update.

This reduces the amount of data to be transferred.

### Damaged database

FreshClam checks the condition of existing databases during the update process.

If an existing database is detected as damaged, it is not simply continued incrementally.

ClamAV documents that a damaged database is instead replaced by a complete new download.

FreshClam thus has a mechanism for replacing a local database that can no longer be used with a complete database version again.

## Database types

ClamAV uses various database files and signature formats.

The central official databases include the virus databases provided by ClamAV.

ClamAV refers to the complete signed database containers as **CVD – ClamAV Virus Database**.

FreshClam manages the database versions published by ClamAV automatically.

For normal JUMBO BLOCK Rescue use, no manual editing of the individual signature files is therefore required.

## CVD

CVD files are signed ClamAV database archives.

They contain signature data and metadata that can be used by the ClamAV engine.

FreshClam handles obtaining and updating these databases.

Manual modification of official CVD files is not intended for normal rescue operation.

## CDIFF

FreshClam uses CDIFF files or CDIFF content for incremental updating.

Instead of always transferring a complete new database, changes relative to the existing database version can be applied.

A distinction must therefore be made between two situations:

**Initial installation or missing database:**
full download.

**Existing database that can be updated:**
FreshClam attempts an incremental update.

If the local database is damaged, FreshClam can fall back on a full download.

## Digital signatures

FreshClam supports the verification of digital signatures of the ClamAV databases.

Checking the integrity and authenticity of the official database files is an essential part of a secure signature supply.

A malware database should not be taken into a rescue system from an arbitrary or unverified source.

## Manual update

For an immediate update attempt, FreshClam can be started manually:

```
sudo freshclam
```

The process and any error messages are output directly in the terminal.

This is particularly useful for JUMBO BLOCK Rescue when it is to be checked before a malware scan whether the existing signatures can be updated.

### Interactive operation

FreshClam can be run **on demand**, i.e. as required via the command line.

This is the most obvious mode of operation for a rescue system.

The user decides deliberately when an internet connection is established and an update is attempted.

After a successful update, the relevant storage devices can then be examined with `clamscan`.

### Daemon mode

FreshClam also supports background operation.

The following option runs FreshClam in daemon mode:

```
freshclam -d
```

In this mode, the program can check regularly for new database versions.

This mode of operation is mainly relevant for systems that run permanently.

A rescue system that is only started for a specific maintenance or emergency deployment does not necessarily need a permanently running FreshClam daemon.

### DNS-based version check

FreshClam supports database version checks via DNS.

This allows the program to establish whether newer database versions are available without necessarily having to transfer the complete database for each check.

This function is part of FreshClam's normal update mechanism.

### Proxy support

FreshClam can work via an HTTP proxy.

Proxy authentication is also supported.

The corresponding settings are configured via `freshclam.conf`.

This can be relevant if JUMBO BLOCK Rescue is operated in a corporate or KRITIS environment in which internet access is only permitted via a proxy.

The specific proxy data must match the environment concerned and must not be specified across the board.

### Configuration file

FreshClam requires:

`freshclam.conf`

The configuration controls, among other things, the update operation.

ClamAV points out that the exact location can depend on the installation.

With packaged Linux installations, the configuration is usually prepared by the package or the distribution.

The configuration actually installed should therefore be used and not replaced by an example configuration taken from another installation.

### Checking the configuration

ClamAV provides the following tool for examining the ClamAV system configuration:

`clamconf`

Calling the following command shows important build, platform, engine, database and configuration information, among other things:

`clamconf`

`freshclam.conf` and other ClamAV configuration files can also be taken into account.

This is helpful if a signature update does not work because of a configuration deviation.

### Database directory

FreshClam writes the downloaded signature databases to the configured ClamAV database directory.

The user under which FreshClam runs needs the necessary write permissions there.

The programs that then use the databases for scanning need at least sufficient read access.

The specific path should be determined from the installed configuration or the system's ClamAV configuration.

It should not be assumed in the rescue manual as a universal path without checking.

### Database ownership and permissions

ClamAV supports the following setting in `freshclam.conf`:

`DatabaseOwner`

This makes it possible to specify the user context under which the database is managed.

ClamAV points out that this user must own the database directory or have the necessary permissions so that FreshClam can save updates.

Missing write permissions can prevent a signature update.

### Update log

FreshClam can log its activities.

An update log file can be specified via the configuration.

Such a log can be particularly helpful if updates fail or the last successful update is to be traced.

For a rescue system, this can be useful for troubleshooting.

### Network connection

For a normal online update, FreshClam needs a working network connection to the ClamAV distribution network.

Without a reachable update source, FreshClam cannot download new official signatures.

However, this does not automatically mean that ClamAV cannot scan at all.

If usable local databases are already present, `clamscan` can work with these existing databases.

The scan is then carried out with exactly the signature version available locally.

### Offline operation

JUMBO BLOCK Rescue can also be used in an environment in which no internet connection is available or permitted.

In this case, FreshClam cannot carry out a normal online comparison with the ClamAV distribution network.

ClamAV Scan can continue to use existing local signature databases, provided that they are valid and usable for the installed engine.

When assessing the scan result, however, it must then be taken into account that the signatures may not correspond to the latest published version.

### Offline does not mean ineffective

A malware scan with an older but valid signature database is not automatically ineffective.

The existing detection data basically remains available.

However, signatures and database changes that were only published after the last update are missing.

An offline scan must therefore be described correctly as:

**A scan with the existing local signature version.**

It must not be described as a scan with guaranteed up-to-date signatures.

### Updating before a scan

If a network connection is available and permitted under the security rules of the site, the following can first be run before an important malware scan:

```
sudo freshclam
```

After a successful update, the updated local database version is available to the `clamscan` started afterwards.

A sensible rescue workflow is therefore:

check or update the signature version,

then identify the target medium,

then carry out the malware scan,

and document the results.

### Do not force an internet connection

In certain security, KRITIS or forensic scenarios, it may be expressly undesirable to connect the rescue system to a network.

How up to date the malware signatures are must be weighed against this security requirement.

A signature update is no reason to connect a deliberately isolated environment to an external network without approval.

In such a case, work is carried out with the existing local database version, or a controlled offline update procedure of the organisation concerned is used.

### Controlled offline update

ClamAV databases can basically also be provided locally.

For a controlled offline scenario, the required database files must be transferred via a trustworthy process to the rescue system or into the intended database directory.

At least the following must be taken into account:

origin of the databases,

integrity,

compatibility with the ClamAV version used,

file permissions,

and the database path actually used.

An organisation-specific offline update procedure is not part of the baseline package list and must therefore not be assumed to be an already implemented JUMBO BLOCK function.

### FreshClam and local mirrors

Via its configuration, FreshClam can use different update infrastructures.

In larger or isolated organisations, the signature supply can therefore basically be organised via a controlled internal infrastructure.

However, whether such an infrastructure exists depends on the installation or organisation.

JUMBO BLOCK Rescue does not require such a server.

### Signatures and engine compatibility

Signature databases and scan engine must be compatible with each other.

New database functions can require certain minimum engine capabilities.

The signature supply must therefore not be considered completely separately from the installed ClamAV version.

For Baseline 20261001, the installed engine that is authoritative is:

**ClamAV 1.4.3**

### Update errors

A failed FreshClam update can have various causes.

These can include, among other things:

no network connection,

unreachable update infrastructure,

faulty configuration,

proxy problems,

missing write permissions,

or problems with the local database.

The specific FreshClam error message should therefore be evaluated instead of interpreting a failed update attempt across the board as a defect in ClamAV.

### Repeated update attempts

FreshClam should not be restarted repeatedly and uncontrollably at very short intervals if an update fails because of a network or server problem.

A failed update should first be analysed on the basis of the message output.

In particular, network, DNS, proxy or configuration problems are not resolved by simply restarting the same command repeatedly.

### Signatures are no guarantee

Even completely up-to-date ClamAV signatures do not guarantee that every existing piece of malware is detected.

Signature-based and other detection methods used by ClamAV can only detect threats for which corresponding detection characteristics exist or can be captured by the engine.

Conversely, false positives can occur.

Updating the signatures therefore improves the quality and currency of the detection basis but does not replace a comprehensive security analysis.

### Signature version and scan report

In a professional examination, the database version with which the scan was carried out should be taken into account in addition to the actual scan result.

This improves traceability.

A later scan with updated signatures can produce a different result from an earlier scan of the same files.

This is possible in particular with newly added detections.

### Separating update and scan

FreshClam itself does not examine any target directories for malware.

Its task is to maintain the detection databases.

Conversely, `clamscan` is not the tool for regularly updating these databases.

The functional separation thus remains clear:

`freshclam`
**signature supply**

`clamscan`
**malware scan**

In JUMBO BLOCK Rescue, this separation is deliberately reflected in two separate manual chapters.

### Typical rescue workflow with an internet connection

If an internet connection is permitted and available, the workflow can look like this, for example:

```
sudo freshclam
```

Then check whether the update has been completed successfully.

The relevant storage device can then be examined with the `clamscan` described in A.10.

The update should take place before the scan so that the scan works with the database version currently available locally.

### Typical rescue workflow without an internet connection

If no internet connection is available or permitted, a successful online update is not assumed.

Instead, work is carried out with the valid signature databases already present locally.

The resulting scan is then expressly a:

**Scan with the signature version available on JUMBO BLOCK Rescue.**

How current this version is must be taken into account in the interpretation.

### Safety notes

Signature databases should only be obtained from trustworthy sources.

A rescue system should not be connected to a network without checking just for a signature update if the deployment requires an isolated environment.

A successful FreshClam update does not mean that every piece of malware can be detected afterwards.

Conversely, a failed update does not necessarily mean that ClamAV can no longer scan.

What matters is whether a valid local database is present.

In offline deployments, the existing signature version must be documented as such and taken into account when assessing the scan result.

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, ClamAV signatures forms the **supply level for the malware detection databases**.

The functional chain is:

**FreshClam → update signature databases**

**ClamScan → examine files with the locally available signature databases**

A.10 and A.11 thus complement each other directly.

ClamAV Scan answers the question:

**“What is detected in the selected files?”**

ClamAV signatures answers the preceding question:

**“Which detection database is the scanner working with?”**

This distinction is essential for a portable rescue system in particular, because JUMBO BLOCK Rescue can be used both online with previously updated signatures and in isolation with the existing local database version.

The signature version is therefore an essential part of the technical validity of a ClamAV scan.

# A.12 Chkrootkit

**Area:** Security
**Program:** Chkrootkit
**Baseline 20261001:** Chkrootkit 0.58b
**Installed package:** `chkrootkit 0.58b-5+b8`
**Operation:** command line
**Task:** local examination of a Unix/Linux system for signs of rootkits and manipulation

## Purpose

Chkrootkit is used for the local examination of a system for **signs of rootkits and certain traces of a system compromise**.

The name stands for:

**Check Rootkit**

JUMBO BLOCK Rescue Baseline 20261001 contains:

`chkrootkit 0.58b-5+b8`

Chkrootkit is not a conventional virus scanner like ClamAV.

Instead of examining files generally for malware signatures, Chkrootkit carries out a series of specialised checks aimed at typical manipulations, files, processes, network states and other characteristics of certain rootkits or compromised systems.

## What is a rootkit?

A rootkit is software or a collection of components that, after a system has been compromised, can be used to maintain privileged access or to conceal activities and components of an attack.

Rootkits can, for example, try to hide processes, files, network connections or login traces from the administrator.

Chkrootkit searches for certain known characteristics and inconsistencies that can indicate such manipulations.

## Components of Chkrootkit

Chkrootkit does not consist of just a single test.

The project comprises several components.

The central component is:

`chkrootkit`

A shell script that examines system programs and other system components for indications of rootkit manipulation.

Other components documented by the project include:

`ifpromisc`
checks network interfaces for promiscuous mode.

`chklastlog`
checks for deleted or conspicuous entries in connection with `lastlog`.

`chkwtmp`
checks `wtmp` for indications of deleted entries.

`chkproc`
looks for signs of certain LKM trojans or hidden processes.

`chkdirs`
looks for further signs of certain LKM manipulations.

`chkutmp`
checks for deleted or conspicuous `utmp` entries.

The individual components are used by the main check according to their task.

### Basic call

The basic call is:

```
sudo chkrootkit
```

If no individual test is specified, Chkrootkit carries out its intended checks against the target system.

Elevated privileges are advisable for a comprehensive system check, as certain system areas may otherwise not be fully accessible.

### Output

Chkrootkit outputs the results of its individual checks in the terminal.

Different states or messages can appear.

The crucial point is:

**A conspicuous Chkrootkit message is initially a pointer for investigation.**

It must be assessed in the context of the system.

Chkrootkit itself and the Debian integration expressly take account of the problem of possible false positives and provide ways of excluding known false alarms from the output of certain tests.

### Displaying available tests

The following option lists the available tests:

```
chkrootkit -l
```

This makes it possible to establish which individual checks are offered by the installed Chkrootkit version.

The tests comprise both general checks and examinations of certain system programs and known rootkit characteristics.

### Running a single test

Chkrootkit allows a test name to be specified after the options.

Basic syntax:

```
chkrootkit [OPTIONEN] [TEST]
```

This means that the complete set of tests does not necessarily have to be run for every examination.

The test names actually available can be determined with the following option:

```
chkrootkit -l
```

### Quiet mode

The following option runs Chkrootkit in quiet mode:

```
-q
```

Example:

```
sudo chkrootkit -q
```

In this mode, the output of tests that have found nothing suspicious is suppressed.

This can make the output of an examination considerably more compact.

However, quiet mode does not mean that fewer tests are carried out.

It primarily changes how the results are displayed.

### Expert mode

The following option enables expert mode:

```
-x
```

Example:

```
sudo chkrootkit -x
```

For many checks, this mode produces additional information about what the test concerned has found.

The output can therefore become considerably more extensive.

Expert mode is particularly suitable for a more detailed analysis of a check that has previously proved conspicuous.

### Debug mode

The following option enables debug mode:

```
-d
```

This shows in detail what Chkrootkit does during its execution.

The mode is primarily intended for troubleshooting and technical analysis.

It is not required for a normal initial rootkit check.

### Do not combine options

With Chkrootkit, it should be noted that options should not be combined as with many other Unix programs.

For example, instead of:

```
-qn
```

They must be specified separately:

```
-q -n
```

This corresponds to the documented operation of Chkrootkit.

### Examining an external system

One option is particularly important for JUMBO BLOCK Rescue:

```
-r DIR
```

This allows another directory to be used as the root directory of the system to be examined.

Example principle:

```
sudo chkrootkit -r /mnt/zielsystem
```

The specific mount point `/mnt/zielsystem` is merely an example here.

It must be replaced by the actual mount point of the system to be examined.

This function is particularly valuable for a rescue system.

In this way, a possibly compromised Linux system can be mounted and examined from JUMBO BLOCK Rescue without having to start the installed target system itself.

### Advantage of external examination

A rootkit can try to manipulate the view of the running compromised system.

If Chkrootkit is run within precisely this system, it must therefore be taken into account that the environment used for the examination could itself be compromised.

The following option enables a different approach:

`-r`

The suspicious file system is mounted under an independent environment and then examined as the target root.

This is exactly the way of working that fits the concept of JUMBO BLOCK Rescue.

The target operating system does not have to be started for such a file-system-based examination.

### Using alternative trustworthy programs

Another security-relevant option is:

`-p DIR1:DIR2:...`

This allows an alternative search path to be specified for external programs that Chkrootkit needs for its checks.

Chkrootkit itself uses standard programs such as `find` and `grep`.

The tool must therefore basically take into account that such programs could have been manipulated on a compromised system.

`-p` can be used instead to specify a path to trustworthy copies of these programs.

Example principle:

```
sudo chkrootkit -p /media/usb
```

The Debian documentation names exactly this purpose: trustworthy copies of the required programs can be stored in a location where they cannot be modified by the system being examined.

### Significance for JUMBO BLOCK Rescue

This way of working in particular illustrates a central advantage of an independent rescue system.

If a suspicious installed system is examined from JUMBO BLOCK Rescue, the programs of the rescue environment are available independently of the possibly manipulated target operating system.

Nevertheless, with every examination it must be checked precisely which programs and data Chkrootkit actually uses.

An external rescue environment makes an examination more robust but does not replace a complete forensic analysis.

### Skipping NFS directories

The following option instructs certain checks to ignore NFS-mounted directories:

```
-n
```

This can prevent network file systems from being included unnecessarily in a local examination.

### Excluding file system types

The Debian integration of Chkrootkit also supports:

```
-T FSTYPE
```

This allows file systems of a specified type to be excluded from certain checks.

This is based on the corresponding file system type check of `find`.

For a targeted rescue scan, this option can help to exclude irrelevant file systems from certain search runs.

### Excluding known false positives

Chkrootkit supports:

```
-e "DATEI1 DATEI2 ..."
```

This allows known files or directories to be excluded from the results of certain tests.

The list is specified separated by spaces and therefore usually has to be put in quotation marks in the shell.

The option can also be used several times.

It is important to note:

`-e` should not be used simply to make an unknown warning disappear.

An exception should only be set up once a reported finding has actually been examined and identified as an expected or known false positive.

### Sniffer test and false positives

Chkrootkit contains a test for network interfaces or processes that can indicate packet sniffing.

Legitimate network programs can also be flagged.

The Debian documentation names, for example, regular network managers such as the following as possible triggers of corresponding messages:

`systemd-networkd`

`NetworkManager`

`wpa_supplicant`

This is an important example of why a Chkrootkit indication does not automatically prove a compromise.

### Sniffer exceptions

For the sniffer test, Chkrootkit supports the following option:

`-s REGEXP`

This allows expected results to be filtered out of the output of this test using a regular expression.

Here too, the following applies:

An exception should only be set up for a legitimate process that has been technically verified.

Unknown sniffer messages should be examined first.

### Promiscuous mode

The `ifpromisc` component checks whether network interfaces are operating in **promiscuous mode**.

This mode basically allows an interface to receive more network frames than only those addressed directly to this interface.

Packet sniffers can use this mode.

However, promiscuous mode is not automatically proof of malware.

Legitimate network diagnostic, monitoring and virtualisation solutions can also use corresponding modes of operation.

The result must therefore be assessed in the context of the system being examined.

### Checking system programs

A central part of Chkrootkit is examining system programs for known indications of rootkit manipulation.

The list of available tests contains numerous typical Unix/Linux programs and services.

These include, for example, checks relating to:

`login`

`ls`

`ps`

`netstat`

`passwd`

`sshd`

`syslogd`

`tcpdump`

`top`

Other system components are also checked.

Chkrootkit looks for characteristics known from known rootkit manipulations or certain compromised versions.

### LKM checks

Chkrootkit has components for searching for signs of certain **Loadable Kernel Module (LKM)**-based trojans or rootkits.

These include:

`chkproc`

and

`chkdirs`.

These checks try to detect certain inconsistencies or indications that can be associated with hidden processes or directories.

Such results are also to be understood as indicators and must be checked technically.

### Login traces

Rootkits or attackers can try to remove traces of logins from system logs.

Chkrootkit therefore has check programs for the corresponding login databases.

These include:

`chklastlog`

`chkwtmp`

`chkutmp`

These look for certain abnormalities or indications of deleted entries.

### chkwtmp

`chkwtmp` examines the following file for entries that have been overwritten or replaced with null bytes:

`/var/log/wtmp`

If such areas are found, the tool can use neighbouring time stamps to output a time range for the conspicuous entry.

However, the documentation expressly names technical limitations of this check.

A conspicuous result is therefore a pointer for further investigation and not sole proof of an attack.

### chklastlog

`chklastlog` compares login information and looks for users whose login activity does not appear consistently in the login data intended for this.

Such an inconsistency can indicate that login traces have been removed.

This check also has documented limitations and must be interpreted in the context of the system concerned.

### Known rootkit characteristics

Chkrootkit contains checks for characteristics of numerous known rootkits, worms and trojans.

The developer's website lists a large number of detected or checked rootkit families and variants.

However, this does not mean that Chkrootkit can detect every existing or future rootkit technique.

Detection is based on the implemented tests and known characteristics.

### Chkrootkit is not a virus scanner

Chkrootkit and ClamAV perform different tasks.

**ClamAV**

This examines files with its malware detection engine and the loaded signature databases.

**Chkrootkit**

This examines the Unix/Linux system for certain rootkit characteristics, manipulations and inconsistencies.

Both tools can therefore be useful within a security examination, but they do not replace each other.

### No automatic clean-up

Chkrootkit is a **detection and analysis tool**.

Its main task is to identify and report suspicious states.

It is not a general tool for automatically removing rootkits found.

A confirmed system compromise requires a separate assessment and suitable response measures.

### False positives

False positives must expressly be taken into account with Chkrootkit.

Legitimate files, network services, monitoring programs or other system configurations can trigger certain tests.

The Debian integration therefore even contains its own documentation of known false positives and supports exclusion options.

A message should therefore initially be treated as a:

**“finding requiring examination”**

Only the subsequent technical analysis decides whether a manipulation actually exists.

### No finding is not proof either

Even the message that Chkrootkit has not found any suspicious characteristics does not prove that the system is guaranteed to be free of compromise.

Chkrootkit can only examine the characteristics for which corresponding check procedures are implemented and to which it has access.

New, unknown or technically different rootkits can lie outside this detection logic.

### Trust problem of a compromised system

Rootkit examinations involve a fundamental problem:

If the operating system itself has been compromised, programs used for the examination may also have been manipulated.

Precisely for this reason, Chkrootkit supports alternative trustworthy programs with `-p` and the examination of an externally mounted target system with `-r`.

For JUMBO BLOCK Rescue, this is a central conceptual advantage.

The examination can be carried out from a separate rescue environment.

### Typical rescue workflow

A sensible first test of the running environment is:

```
sudo chkrootkit
```

For more compact output:

```
sudo chkrootkit -q
```

If a check is conspicuous, the test concerned can then be examined further, specifically or in expert mode.

For an externally mounted suspicious Linux system, the following can basically be used:

```
sudo chkrootkit -r /pfad/zum/zielsystem
```

The actual mount point must be established unambiguously beforehand.

### Procedure in the event of a warning

A Chkrootkit warning should not lead immediately to changes to the original system.

Instead, the following should be established first:

Which test triggered the warning?

Which file, process or system state was flagged?

Is the state concerned to be expected on the system?

Is a known false positive documented for this test?

Can the finding be confirmed independently with other tools?

Only then should further measures be decided on.

### Evidence preservation

If there is a serious suspicion of compromise, it should be borne in mind that changes to the original system can alter potentially relevant traces.

In a forensically relevant case, backing up the initial state is therefore more important than immediate “clean-up”.

With its data recovery and analysis tools, JUMBO BLOCK Rescue provides ways of backing up storage devices or data first.

However, Chkrootkit itself is not a forensic imaging tool.

### Chkrootkit on a damaged storage device

With a physically unstable storage device, a distinction must be made between security analysis and data recovery.

Chkrootkit needs access to numerous files and system areas.

An extensive examination therefore generates read accesses.

If a drive is already showing hardware errors and important data has to be rescued, you should not start by forcing a complete rootkit examination.

In this case, backing up the still readable data takes priority.

## Chkrootkit and ClamAV

The two previous chapters and Chkrootkit form different examination levels:

**ClamAV signatures**
provides the malware detection databases.

**ClamAV Scan**
examines files for malware detectable by ClamAV.

**Chkrootkit**
looks for certain signs of rootkits and system manipulation on Unix/Linux systems.

A more comprehensive security examination can therefore combine several of these tools.

## Limitations

Chkrootkit has clear limitations.

It does not detect every possible compromise.

It does not replace a complete forensic examination.

It does not automatically prove the presence of a rootkit on the basis of a single warning.

It can produce false positives.

And if it is used within an already compromised running system, it must be taken into account that parts of the examination environment could have been manipulated.

## Safety notes

Chkrootkit results must be interpreted.

A message should not be treated as a confirmed infection on the basis of its wording alone.

Known false positives may only be hidden after technical checking.

When examining a third-party system, the correct mount point should be determined unambiguously.

If there is a serious suspicion of compromise, unnecessary changes to the original system should be avoided.

With physically damaged storage devices, data backup takes priority over extensive security scans.

## Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Chkrootkit forms a specialised examination level for **indications of rootkits and manipulation on Unix/Linux systems**.

The ability to examine a suspicious Linux system from an independent rescue environment is particularly relevant here.

The security tools thus complement each other:

**ClamAV Scan – file-based malware examination**

**ClamAV signatures – maintenance of the ClamAV detection databases**

**Chkrootkit – search for indications of rootkits and manipulation**

**Rkhunter – additional host-based rootkit and security check**

**GUFW Firewall – management of the host-based firewall**

Chkrootkit does not provide an automatic decision of “system secure” or “system compromised”.

Its task is more precise:

**It looks for technically defined signs that can justify further investigation.**

It is precisely in this function that Chkrootkit is a diagnostic tool within JUMBO BLOCK Rescue.

# A.13 Rkhunter

**Area:** Security
**Program:** Rootkit Hunter (Rkhunter)
**Baseline 20261001:** Rkhunter 1.4.6
**Installed package:** `rkhunter 1.4.6-13`
**Operation:** command line
**Task:** examination of local Linux/Unix systems for indications of rootkits, malware and suspicious system changes

## Purpose

Rkhunter stands for:

**Rootkit Hunter**

The program carries out various security checks on a local Linux/Unix system. It tries to detect known rootkits and malware as well as certain suspicious changes and system states.

JUMBO BLOCK Rescue Baseline 20261001 contains:

`rkhunter 1.4.6-13`

The underlying upstream version is therefore Rkhunter 1.4.6.

Rkhunter is not a conventional virus scanner and does not replace ClamAV.

The program combines several types of security check. These include examinations for known rootkit characteristics, changes to important system commands and startup files, and checks of network interfaces and listening applications.

## Basic check principle

Rkhunter does not work with just a single rootkit test.

The program carries out a series of different checks and summarises their results.

These include in particular:

checks of system commands,

checks of stored file properties,

search for known rootkit files and directories,

search for suspicious files,

checks for certain malware characteristics,

checks of system startup files,

checks of network states,

and other system-specific security checks.

Not every test can be run on every system.

If programs required by a particular test are missing, Rkhunter can skip this test.

### Basic system check

The central check is started with the following command:

```
sudo rkhunter --check
```

The short form is:

```
sudo rkhunter -c
```

Rkhunter then runs its enabled checks and displays the results in the terminal.

If something is classified as suspicious, Rkhunter generates a warning.

In addition, a log of the tests carried out and their results is created by default.

### Interactive process

With a normal `--check`, Rkhunter can wait after certain test sections for the user to continue by pressing Enter.

For an uninterrupted scan, the following options are therefore available:

```
--skip-keypress
```

or

```
--sk
```

Example:

```
sudo rkhunter --check --skip-keypress
```

Or:

```
sudo rkhunter -c --sk
```

The check then runs through to the end without these manual interruptions.

### Warnings

An Rkhunter warning is initially a **technical indication of a state requiring examination**.

It is not automatically proof of a rootkit or a successful compromise.

This applies in particular to checks for changed files.

A file can change for legitimate reasons, for example through a regular software update.

Rkhunter can establish that certain properties deviate from a stored reference state. However, this does not automatically tell you why this change took place.

### No finding is no guarantee of security

Conversely, a completed Rkhunter check without warnings does not mean that a system is guaranteed to be free of compromise.

Rkhunter can only search for the characteristics and states for which corresponding check procedures are implemented.

Unknown, new or technically different attacks can lie outside this detection.

Rkhunter is therefore a diagnostic tool and not a mathematical proof of the integrity of a system.

### Displaying supported checks

The following option displays the available test names:

```
rkhunter --list tests
```

This makes it possible to trace which tests are provided by the installed version.

The following option displays the list of rootkits that Rkhunter searches for:

```
rkhunter --list rootkits
```

Other `--list` variants can output, among other things, information about languages, Perl support and the files used for the file properties database.

### Enabling individual checks

The following option enables specific tests selectively:

```
--enable
```

Basic principle:

```
rkhunter --check --enable TEST
```

This allows an examination to be limited to selected test groups.

The test names actually available should first be determined with:

```
rkhunter --list tests
```

### Disabling checks

The following option disables specific tests:

```
--disable
```

Basic principle:

```
rkhunter --check --disable TEST
```

Here too, only test names that actually exist should be used.

Selectively switching off a warning should not serve merely to remove an unknown finding from the output.

First, it must be clarified why the check concerned generates a warning.

### Configuration

Rkhunter uses a central configuration file:

```
rkhunter.conf
```

The program can also use a local configuration file:

```
rkhunter.conf.local
```

It can also use a directory:

```
rkhunter.d
```

This directory contains additional `.conf` files.

Command-line options can override corresponding settings from the configuration files.

The actual configuration of the installed system is relevant for interpreting a scan.

### Checking the configuration

The following option checks the configuration:

```
rkhunter --config-check
```

Short form:

```
rkhunter -C
```

Rkhunter checks the configuration options used and reports any configuration problems detected.

Such a check makes sense after changes to the configuration.

### File properties

An important function of Rkhunter is monitoring certain properties of important system files and commands.

For this purpose, Rkhunter stores reference values in its file properties database and later compares them with the current state.

Hash values, among other things, can be used for this.

By default, Rkhunter uses SHA-256 for this function unless the configuration or system environment specifies otherwise.

### Hash methods

Rkhunter supports several hash methods for the file properties check.

These include:

MD5,

SHA-1,

SHA-224,

SHA-256,

SHA-384,

SHA-512.

The following option can be used to influence the method used:

```
--hash
```

For normal rescue use, however, the existing configuration should not be changed without a technical reason.

### Properties database

The stored file properties form a reference state.

In later checks, Rkhunter can establish that properties of a monitored file have changed relative to this reference state.

This function is only meaningful if the stored reference state is trustworthy.

An already compromised state must not simply be declared the new trustworthy reference.

### `--propupd`

The following command updates the stored file properties database:

```
rkhunter --propupd
```

This command is security-critical.

Rkhunter expressly points out that the program can only detect **that** a file has changed. From this fact alone, it cannot establish what caused the change.

If the following command is run after a change, Rkhunter adopts the current properties as the new reference state:

```
rkhunter --propupd
```

This implicitly assumes that the file concerned is trustworthy.

### Do not use `--propupd` blindly

For this reason, the following command must not simply be run to eliminate warnings after a scan:

```
rkhunter --propupd
```

First, it must be clarified why the file has changed.

Examples of legitimate causes can be regular package updates.

However, a change can just as well be part of a manipulation.

Only when the new state has been assessed as trustworthy may it be adopted as the new reference.

This is one of the most important safety notes when working with Rkhunter.

### Selective `--propupd`

Rkhunter can also update the properties database selectively for individual files, directories or package names.

This means that the entire database does not necessarily have to be recreated.

However, the following also applies to a selective update:

**Only a verified state may be stored as the new reference.**

### Package manager support

For the file properties check, Rkhunter can use information from a supported package manager.

The following option supports, among others, the package managers listed below:

```
--pkgmgr
```

```
DPKG
```

Other package managers are also supported.

This allows file properties to be handled in connection with the information provided by the package manager.

Its use depends on the configuration and the system being examined.

### Updating Rkhunter data

Rkhunter has the command:

```
rkhunter --update
```

This command does **not update the Rkhunter program itself**.

Rather, it checks whether newer versions of the text data files used by Rkhunter are available and downloads them if necessary.

This function requires a suitable network connection and a supported command-line download tool.

### Difference between `--update` and a software update

The terms must be clearly separated. The following command updates Rkhunter data files:

```
rkhunter --update
```

It does not carry out a general Debian/MX Linux package update and does not replace the installed package `rkhunter`.

The installed program version of the baseline remains:

**Rkhunter 1.4.6 / Debian package 1.4.6-13.**

### Version check

The following option checks whether a newer program version is known:

```
rkhunter --versioncheck
```

This function also requires network access.

It must be distinguished from the following option:

```
rkhunter --update
```

`--versioncheck` checks the program version.

`--update` checks the Rkhunter data files.

### Offline operation

JUMBO BLOCK Rescue can be operated without an internet connection.

In this case, the local Rkhunter checks can basically still be run with the existing data and the existing configuration.

By contrast, online functions such as the following cannot be carried out successfully without a reachable update source:

```
--update
```

or

```
--versioncheck
```

An offline check is therefore an examination with the locally available Rkhunter version.

### No obligation to connect to a network

In isolated security, KRITIS or forensic environments, a network connection must not be established merely because Rkhunter could update data.

The security requirements of the deployment take priority.

An existing local Rkhunter version can still be used for examinations, although how up to date it is must be taken into account in the assessment.

### Logging

By default, Rkhunter creates a log file during a normal check.

The documented default path is:

```
/var/log/rkhunter.log
```

The following options specify a different location:

```
--logfile
```

or

```
-l
```

The log contains much more information than just the final summary and should be evaluated if there are warnings.

### Previous log

By default, Rkhunter creates a new log file on each new run.

An existing log file is renamed with the following extension:

```
.old
```

The following option appends to an existing log file instead:

```
--appendlog
```

For documented rescue examinations, it should be decided deliberately how scan logs are kept.

### Displaying warnings only

The following options restrict the terminal output to warning messages:

```
--report-warnings-only
```

or

```
--rwo
```

This is particularly helpful with automated or extensive checks.

For an initial detailed analysis, however, the full output or the log file can be more informative.

### Summary

After a normal `--check`, Rkhunter displays a short summary of the results by default.

The following options suppress this summary:

```
--nosummary
```

or

```
--ns
```

For manual rescue use, the summary is generally helpful and should not be switched off without reason.

### Colours

By default, Rkhunter uses colours to display its test results.

Successful checks, warnings and skipped tests can thus be distinguished visually.

The following option switches off colour output:

```
--nocolors
```

This can be useful in particular for logging or automated processing.

### Quiet mode

The following options suppress the normal output:

```
--quiet
```

or

```
-q
```

This mode is mainly useful when only return values or selectively enabled output are to be evaluated.

For an interactive security examination, visible output is usually more helpful.

### Network checks

Rkhunter carries out various checks in the network area.

The program documentation expressly names examinations of network interfaces and listening applications.

Such checks can provide indications of unexpected services or network states.

However, an open or listening port is not automatically proof of a rootkit.

It must be compared with the system state actually expected.

### System startup files

Rkhunter also examines certain system startup files for changes or suspicious states.

This is relevant because persistent malware can try to anchor itself permanently in a system via startup mechanisms.

Here too, the following applies:

A change can be legitimate and must be assessed in context.

### Hidden files and directories

Rkhunter has checks for certain hidden or suspicious files and directories.

However, Unix/Linux systems regularly use numerous files and directories whose names begin with a dot.

“Hidden” therefore does not automatically mean “malicious”.

What matters is whether a finding is expected and explainable for the system concerned.

### Rootkit-specific checks

Rkhunter contains tests for numerous known rootkits and malware characteristics.

Among other things, version 1.4.6 added checks for further malware or backdoors and a test for the Diamorphine LKM technique.

The Ebury `sshd` backdoor complex was also taken into account in this version through additional checks.

These tests are version-specific and part of the detection logic of Rkhunter 1.4.6.

### Shared memory check

Rkhunter 1.4.6 contains checks of suspicious shared memory segments.

The version introduced additional configuration options to allow certain PIDs or users specifically in this check and to set a minimum size for shared memory segments to be checked.

Here too, legitimate applications can produce conspicuous states.

A warning must therefore be examined.

### Additional auxiliary programs

Some Rkhunter checks can use additional external programs.

If required programs are not installed, the corresponding tests can be skipped.

The Rkhunter project names, for example, `unhide` as an optional tool for additional checks.

When interpreting a report, it is therefore important to look not only at warnings but also at which tests were actually run or skipped.

### Chkrootkit and Rkhunter

Chkrootkit and Rkhunter pursue a similar goal but do not work identically.

**Chkrootkit**

This carries out specialised checks for known rootkit characteristics, manipulated programs, hidden processes and certain login or network abnormalities.

**Rkhunter**

This additionally combines rootkit and malware checks with extensive monitoring of certain system file properties, system configurations, startup files and other security states.

The programs complement each other.

A finding by one program is not automatically reproducible by the other.

### Rkhunter and ClamAV

Rkhunter does not replace a ClamAV scan either.

**ClamAV** examines accessible files with its malware detection engine.

**Rkhunter** examines the state and certain properties of a Linux/Unix system.

The two tools thus cover different security aspects.

### Special feature of a rescue system

According to its own documentation, Rkhunter is primarily designed for checks of the **local system**.

In one important respect, this distinguishes it from the Chkrootkit function `-r` described in A.12, with which an alternative root directory can be expressly specified.

For Rkhunter 1.4.6, there is no equivalent general `--rootdir` option in the documented command-line interface.

It must therefore not be claimed that Rkhunter can fully examine any mounted third-party Linux system as an alternative root system in the same way as Chkrootkit with `-r`.

### Consequence for JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Rkhunter is therefore particularly suitable for checking the **currently running Linux environment or a system in whose system context Rkhunter is run correctly**.

For the direct examination of a third-party Linux file system that is merely mounted under a mount point, Chkrootkit with its documented `-r` function is more flexible.

This restriction is important so that the manual does not attribute an unsubstantiated offline root function to Rkhunter.

### Handling warnings correctly

In the event of an Rkhunter warning, the following should be established first:

Which test generated the warning?

Which file or state is affected?

Can the deviation be attributed to a legitimate package or system update?

Can the current state be traced from a trustworthy source?

Are there further independent indications of a compromise?

Only then should it be decided how to deal with the finding.

### Never use `--propupd` to eliminate warnings

Particular care applies once again to:

```
rkhunter --propupd
```

This command must not be understood as a means of “repairing” warnings.

It updates the reference state.

If it is run on a compromised system, manipulated files can be adopted as the new normal state.

The following therefore applies:

**Verify first, then update the reference.**

### Physically damaged storage devices

Rkhunter is not a data recovery tool.

With a physically unstable drive, data backup and imaging take priority.

An extensive security check generates numerous read accesses and should not be forced before an urgently needed backup of a storage device at risk of failure.

GNU ddrescue, for example, is intended for the actual rescue.

### No automatic clean-up

Rkhunter is a checking and analysis tool.

A warning does not automatically lead to a safe clean-up of the system.

In the event of a confirmed compromise, it must be decided separately whether a forensic backup, reinstallation, restoration from a trustworthy backup or another response measure is required.

Rkhunter does not make this decision.

### Limitations

Rkhunter does not detect every possible piece of malware or every rootkit.

Detection depends on its implemented tests, data files, configuration and the auxiliary programs available.

False positives are possible.

A changed hash does not prove the cause of the change.

An inconspicuous scan does not prove complete system integrity.

And a warning is initially an indicator that must be examined technically.

### Safety notes

For comprehensive system checks, Rkhunter should be run with the required privileges.

Warnings must not be interpreted as a confirmed infection without checking.

Equally, they must not simply be eliminated by whitelisting or `--propupd`.

Before `--propupd`, the current file state must be verified as trustworthy.

Online updates must not be forced in isolated or forensic environments contrary to the security requirements.

In the case of conspicuous results, the scan log should be kept and evaluated.

### Typical rescue workflow

For a normal check of the running system, the configuration can first be checked:

```
sudo rkhunter --config-check
```

A complete check without interruptions can then be carried out:

```
sudo rkhunter --check --skip-keypress
```

With extensive output, the log can then be evaluated.

For a targeted display of warnings, the following is available:

```
sudo rkhunter --check --skip-keypress --report-warnings-only
```

If changes to important files are reported, they are first verified independently.

**Only once their legitimacy has been established may an update of the properties database be considered.**

### Role in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Rkhunter forms an additional level of **host-based Linux/Unix security diagnostics**.

The security tools complement each other:

**ClamAV Scan – file-based malware examination**

**ClamAV signatures – supplying the ClamAV engine with detection data**

**Chkrootkit – search for indications of rootkits and manipulation, including the documented examination of an alternative root directory**

**Rkhunter – rootkit, malware, file property, startup file and other host-based security checks**

**GUFW Firewall – graphical management of the host-based firewall**

The particular strength of Rkhunter lies in the combination of known rootkit/malware checks and the monitoring of security-relevant system properties.

At the same time, its most important methodological limitation is:

**Rkhunter reports deviations and suspicious states – it does not by itself prove their cause.**

Within JUMBO BLOCK Rescue, Rkhunter is therefore to be understood as a diagnostic and control instrument whose warnings must then be assessed technically.

# A.14 GUFW Firewall

**Area:** Security
**Program:** GUFW Firewall
**Baseline 20261001:** GUFW 24.04.0
**Installed package:** `gufw 24.04.0-3`
**Firewall basis:** UFW 0.36.2
**Installed package:** `ufw 0.36.2-9`
**Operation:** graphical user interface
**Task:** configuration of the host-based firewall of JUMBO BLOCK Rescue

## Purpose

GUFW provides a graphical user interface for managing the host-based firewall.

The name stands for:

**Graphical Uncomplicated Firewall**

GUFW is not itself the underlying firewall technology.

It provides a graphical interface for **UFW – Uncomplicated Firewall**.

Both components are installed in JUMBO BLOCK Rescue Baseline 20261001:

`gufw 24.04.0-3`

and

`ufw 0.36.2-9`.

The functional chain thus consists of:

**GUFW → graphical configuration**

**UFW → management of the firewall rules**

**Linux network filtering → technical enforcement of the rules in the operating system**

## Area of use in the rescue system

A rescue system is often used in network environments whose trustworthiness cannot be assumed.

Examples are:

third-party corporate networks,

emergency networks,

temporary maintenance networks,

direct Ethernet connections,

Wi-Fi networks,

or networks at a possibly compromised site.

GUFW makes it possible to control the local firewall of JUMBO BLOCK Rescue graphically and to define rules for incoming and outgoing network traffic if required.

GUFW is therefore not a malware detection tool.

It controls network access to and from the running rescue system in accordance with the configured firewall rules.

## GUFW and UFW

GUFW and UFW must be clearly distinguished.

**GUFW**

This is the graphical user interface.

**UFW**

This is the underlying firewall configuration tool.

Changes made via GUFW therefore affect the firewall configuration managed by UFW.

GUFW provides the essential functions in a graphically accessible form, so that no manual entry of UFW commands is required for typical firewall tasks.

## Enabling and disabling the firewall

GUFW has a central status switch for the firewall.

This can be used to enable or disable the firewall.

At command-line level, the corresponding UFW function is basically:

```
sudo ufw enable
```

or

```
sudo ufw disable
```

Disabling the firewall does not necessarily remove the stored rule definition.

It disables their application by UFW.

For rescue use, the firewall should not be disabled without a specific reason.

## Checking the status

At command-line level, the status can be checked with the following command:

```
sudo ufw status
```

A more detailed display is provided by:

```
sudo ufw status verbose
```

This makes it possible to establish, among other things, whether UFW is active and which default policies or rules are in effect.

GUFW displays the corresponding information graphically.

## Basic principle of a host-based firewall

A host-based firewall protects the computer on which it runs.

It can allow or block network traffic on the basis of defined rules.

The following, among other things, can be taken into account:

direction of the traffic,

protocol,

port,

IP address,

subnet,

and specific rule definitions.

GUFW or UFW must therefore be distinguished from an external network firewall.

It directly protects the running JUMBO BLOCK Rescue system.

## Incoming and outgoing traffic

GUFW distinguishes between:

**Incoming – incoming traffic**

and

**Outgoing – outgoing traffic.**

Incoming traffic comprises connection attempts or data traffic that reaches the rescue system from outside.

Outgoing traffic is initiated by the rescue system towards other systems or networks.

Policies can be specified for both directions.

## Default policies

GUFW allows default policies to be selected for incoming and outgoing traffic.

UFW also works with such default policies.

The basic principle is:

If no more specific rule applies to a data flow, the relevant default policy decides.

Default policies thus form the basis of the firewall configuration.

### Typical security approach

A common host firewall approach is:

**to block incoming connections by default**

and

**to allow outgoing connections by default.**

However, this is not a universally prescribed configuration.

In particularly restrictive or isolated deployment environments, outgoing traffic can also be restricted.

The firewall policy actually required depends on the rescue scenario and the security requirements of the network concerned.

### Rules

GUFW allows custom firewall rules to be created.

Depending on the available rule type and configuration, a rule can handle network traffic as follows, for example:

**Allow – permit**

**Deny – deny or block**

**Reject – reject**

**Limit – limit**

The exact effect must be understood in accordance with the selected UFW rule.

### Allow

An allow rule permits the matching network traffic.

At UFW level, a simple example is:

```
sudo ufw allow 22
```

This would allow traffic in accordance with the UFW rule for port 22.

Such a rule should only be set up if the service concerned is actually needed.

### Deny

A deny rule blocks matching network traffic.

Example principle:

```
sudo ufw deny 22
```

With a blocked packet, the remote party does not necessarily receive the same immediate feedback as with a reject rule.

### Reject

A reject rule refuses the traffic and signals the rejection to the remote party.

`reject` thus differs in its network behaviour from a simple `deny`.

Which variant is appropriate depends on the use case.

### Limit

UFW has a limit function.

It can be used in particular to limit frequent new connection attempts to a service.

The UFW documentation names SSH as a typical example.

Basic principle:

```
sudo ufw limit ssh
```

UFW thus limits new connection attempts on the basis of its implemented rate limit logic.

This can additionally protect a service against simple repeated connection attempts.

However, it does not replace a complete authentication or attack protection solution.

### Ports

Firewall rules can refer to specific network ports.

Examples of typical port numbers are:

22 for SSH,

53 for DNS,

80 for HTTP,

443 for HTTPS.

However, a port number alone does not say with certainty which specific process is actually listening on it.

Before a port is opened, it should therefore be known which service uses it and why opening it is needed.

### Protocols

Rules can be differentiated by network protocol.

The following protocols are particularly relevant here:

TCP

and

UDP

A service can use TCP, UDP or both protocols.

A firewall rule should therefore match the actual network protocol of the service.

### IP addresses

UFW rules can be restricted to specific source or destination addresses.

This means that a port does not necessarily have to be opened for every reachable system.

A service can, for example, be permitted only for a specific administrative remote party or a specific network.

For maintenance and rescue scenarios, such a restriction can be considerably more secure than general opening.

### Subnets

Rules can also be applied to IP networks or subnets.

This makes it possible, for example, to specify that a service should only be reachable within a specific maintenance network.

The network ranges used must correspond to the actual infrastructure.

They must not be assumed across the board in the manual.

### Rule order

Firewall rules have an order.

This order can be relevant if several rules affect the same or overlapping network traffic.

At UFW command-line level, rules can therefore also be inserted specifically at certain positions.

With more complex configurations, the rule base should be checked as a whole rather than looking at a single rule in isolation.

### Deleting rules

Rules that are no longer needed can be removed.

At UFW level, this can be done, for example, on the basis of a rule definition or a numbered rule list.

The following command displays the rules with numbers:

```
sudo ufw status numbered
```

A rule can then be deleted specifically.

Before deleting, it should be checked whether the rule is needed for an existing network function.

### Profiles

GUFW supports profiles.

This allows different firewall configurations to be managed for different deployment situations.

This is particularly useful for mobile systems that are used in different network environments.

A restrictive configuration for an unknown network can differ, for example, from a maintenance environment with specifically opened access.

However, the profiles actually used depend on the installation.

The baseline package list alone does not reveal which individual GUFW profiles are preconfigured in JUMBO BLOCK Rescue.

### Logging

UFW supports the logging of firewall events.

Logging can be enabled, among other ways, with the following command:

```
sudo ufw logging on
```

UFW supports different logging levels.

Firewall logging can be helpful for network diagnostics and security analysis.

However, with high network traffic it can generate considerable amounts of log data.

The logging level should therefore match the actual analysis requirement.

### GUFW and logging

GUFW makes the UFW logging function available graphically.

This allows firewall logging to be controlled without direct command-line configuration.

The logs generated are particularly useful when it is to be traced whether network traffic was blocked because of a firewall rule.

### Firewall and network diagnostics

An active firewall can influence the results of other network tools.

This affects, for example:

Nmap,

MTR,

ping-based diagnostics,

certain Wireshark scenarios,

SSH,

and other network services.

If an expected network connection is not established, it must therefore also be checked whether the local firewall is blocking the traffic concerned.

However, the firewall should not be disabled completely too hastily.

First, the specific rule or the required network traffic should be identified.

### GUFW and SSH

JUMBO BLOCK Rescue contains OpenSSH.

If an SSH service on the rescue system is to be made reachable, the firewall configuration must be taken into account as well as the service itself.

However, opening the firewall does not start an SSH server.

Likewise, a running SSH service does not automatically mean that it is reachable through the firewall.

Service status and firewall rule are two different levels.

### GUFW and Nmap

Nmap examines reachable network services and ports.

The local firewall of the rescue system can influence both the reachability of its own services and certain scan or response situations.

When interpreting Nmap results, it must therefore be taken into account whether GUFW/UFW is active and which rules apply.

### GUFW and Wireshark

Wireshark records network traffic that is visible on the monitored interfaces.

GUFW or UFW, by contrast, decides on the host's network traffic on the basis of the firewall rules.

The two programs thus perform different tasks:

**GUFW/UFW – controls traffic**

**Wireshark – analyses or records traffic**

Wireshark does not replace a firewall, and GUFW does not replace a packet analyser.

### GUFW and ClamAV

There is also a clear division of tasks between firewall and malware scanner.

GUFW does not examine files for malware.

ClamAV does not control network ports as a firewall.

The security functions complement each other:

**ClamAV → malware detection**

**Chkrootkit/Rkhunter → rootkit and system check**

**GUFW/UFW → network access control**

### Reset

UFW supports:

```
sudo ufw reset
```

This command resets the firewall configuration to the installation default.

Existing user-defined rules are disabled and reset in the process.

This function is therefore security-critical.

It must not be used as a general troubleshooting step if only a single connection is not working.

Before a reset, it must be known which rules will be lost or changed as a result.

### Disabling the firewall is not a diagnosis

If a network service cannot be reached, switching off the firewall completely is a possible technical test, but not a suitable standard procedure.

It is better first to:

check the firewall status,

check the rule list,

determine the required port and protocol,

establish the source or destination,

and then correct the specific rule.

In this way, the protective effect for other network connections is preserved.

### A firewall does not protect against all attacks

A host-based firewall is only one security layer.

It can block unwanted network traffic in accordance with its rules.

However, it does not automatically prevent, for example:

malware in files that already exist,

attacks via explicitly permitted services,

vulnerabilities within permitted applications,

manipulations via physical storage devices,

or user errors.

GUFW is therefore part of a multi-layered security concept and not a complete security system.

### Outgoing traffic

With a rescue system, outgoing traffic also deserves attention.

If, for example, a potentially compromised storage device is examined, this does not automatically mean that malware on it will be executed.

However, if unknown programs are deliberately started, they can basically initiate network communication.

A more restrictive outgoing policy can be an additional security measure in special analysis environments.

At the same time, it can block legitimate functions such as:

DNS,

web access,

FreshClam updates,

SSH,

package downloads

or other network services.

Such a policy must therefore be planned deliberately.

### Unknown networks

If JUMBO BLOCK Rescue is connected to an unknown or untrustworthy network, the local attack surface should be kept as small as possible.

Incoming services that are not needed should not be opened unnecessarily.

GUFW offers a graphically controllable way of managing the host-based firewall for this.

### Rescue system as a maintenance computer

In certain scenarios, JUMBO BLOCK Rescue must actively reach network services or provide them itself.

Examples can be:

SSH connections,

DNS queries,

web access,

signature updates,

network diagnostics

or file transfers.

Firewall rules must take these requirements into account.

The rule should be tailored as precisely as possible to the service actually needed.

### Document firewall rules before changes

In more complex deployment situations, the current state should be documented before extensive changes.

At UFW level, the following can be used to take stock, for example:

```
sudo ufw status verbose
```

or:

```
sudo ufw status numbered
```

This makes it possible to trace which rules existed before a change.

### Typical rescue workflow

Before connecting to an unknown network, first check whether the firewall is active.

This is done graphically via GUFW.

Alternatively, the status can be checked with:

```
sudo ufw status verbose
```

Then check whether the default policies match the intended use.

If additional services are needed, only the rules that are actually necessary should be created.

After changes, the status is checked again.

In the event of network problems, the specific rule base is examined first before the firewall is disabled completely.

### Safety notes

GUFW changes the active firewall configuration of the rescue system.

An incorrect rule can block necessary network communication.

An allow rule that is too broad, on the other hand, can enable unnecessary network access.

Ports should only be opened if the associated service is known and required.

Where possible, openings should be restricted to the source addresses or networks actually needed.

`ufw reset` deletes or resets the existing user-defined rule configuration and must not be used rashly.

A disabled firewall provides no filtering managed by UFW.

Firewall rules do not replace malware checks, system hardening or secure service configuration.

### Limitations

GUFW is a graphical firewall manager and not an intrusion detection system.

It does not detect rootkits.

It does not scan files for malware.

It does not analyse network packets like Wireshark.

It does not carry out port scans like Nmap.

It does not eliminate vulnerabilities in network services.

Its task is to make the host-based firewall configuration controllable.

### Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, GUFW forms the graphical level of **local network access control**.

The security tools of the rescue system thus cover different tasks:

**A.10 ClamAV Scan**
examination of files and directories for malware detectable by ClamAV.

**A.11 ClamAV signatures**
maintenance of the ClamAV detection databases.

**A.12 Chkrootkit**
search for certain indications of rootkits and manipulation on Unix/Linux systems.

**A.13 Rkhunter**
additional rootkit, malware, file property and host-based security checks.

**A.14 GUFW Firewall**
graphical management of the host-based network firewall.

GUFW thus completes the security area not as a further detection tool but as a **preventive and controlling network component**.

The clear technical assignment is:

**GUFW = graphical user interface**

**UFW = firewall configuration layer**

**Firewall rules = control of the incoming and outgoing network traffic of the rescue system**

For JUMBO BLOCK Rescue, this means in particular that the system has a directly operable local firewall configuration even when deployed in unknown or untrustworthy networks.

# A.15 Binwalk

**Area:** Analysis
**Program:** Binwalk
**Baseline 20261001:** Binwalk 2.4.3
**Installed package:** `binwalk 2.4.3+dfsg1-2+deb13u1`
**Operation:** command line
**Task:** analysis of binary files and images for embedded files, data structures and executable code

## Purpose

Binwalk is a tool for analysing binary files and images.

Its primary area of application is the analysis of firmware images. However, the tool can generally search binary data for known file signatures and embedded data structures.

Among other things, Binwalk can:

detect known file signatures,

determine the offsets of detected structures,

extract embedded content,

examine extracted content further recursively,

search for specific byte sequences,

search for typical executable opcodes,

and analyse the entropy of binary data.

This clearly distinguishes Binwalk from the data recovery and security programs documented so far.

Binwalk primarily answers the question:

**“Which recognisable data structures are located within this binary file, and at which position do they begin?”**

## Typical areas of use

Binwalk was developed in particular for the analysis and reverse engineering of firmware.

Typical objects of examination are, for example:

firmware images,

binary images,

memory dumps,

composite binary files,

files with embedded archives,

compressed data areas,

file system images

and other binary containers.

Within JUMBO BLOCK Rescue, Binwalk thus serves as a specialised analysis tool for files and images whose internal structure is to be examined.

## Basic call

The basic syntax is:

```
binwalk [OPTIONEN] DATEI
```

A simple call is, for example:

```
binwalk firmware.bin
```

Binwalk searches the specified file for known signatures.

For detected structures, their positions within the file, among other things, are output.

## Offsets

An essential result of a Binwalk scan is the **offset**.

The offset designates the position within the examined file at which a detected structure or signature begins.

Binwalk typically displays such positions in both decimal and hexadecimal form.

This allows further examination to be carried out specifically at this position, for example with a hex editor.

This is exactly where the following complement each other:

**Binwalk – find structures and positions**

and

**Hexedit – examine and edit binary data directly.**

## Signature analysis

The central analysis function of Binwalk is the signature scan.

It can be requested explicitly with the following options:

```
-B
```

or

```
--signature
```

Example:

```
binwalk -B firmware.bin
```

Binwalk searches the target file for known file signatures.

Such signatures are based on characteristic byte sequences or magic values and additional detection rules.

### A signature match is not complete proof

A detected magic value or signature initially means that the examined bytes match a known structure.

This is not in every case equivalent to a completely validated or intact embedded file.

Binary data can contain random byte sequences that match known signatures.

Damaged or truncated data can also match a signature only partially.

The results must therefore be assessed in the context of the examined file.

### Output of invalid matches

The following options also display results that Binwalk has marked as invalid:

```
-I
```

or

```
--invalid
```

This option is mainly useful for detailed analyses.

For a normal initial examination, such matches are often less relevant.

### Including specific signatures

The following options restrict the output to signatures whose description or label matches the specified expression:

```
-y
```

or

```
--include
```

Basic principle:

```
binwalk -y AUSDRUCK DATEI
```

This allows an extensive analysis to be restricted specifically to certain structures.

### Excluding signatures

The counterpart is:

`-x`

or

`--exclude`

This allows matching signature results to be excluded from the analysis.

Basic principle:

```
binwalk -x AUSDRUCK DATEI
```

This can be helpful with files containing very many known structures that are irrelevant to the specific examination.

### Searching for a byte sequence

With the following options, Binwalk can search for a specified byte sequence:

`-R`

or

`--raw`

Basic principle:

```
binwalk -R ZEICHENFOLGE DATEI
```

This function is useful when the search is not for a file type already defined by Binwalk but for a specific known byte sequence.

### Opcode scan

With the following options, Binwalk searches for known signatures of executable opcodes:

`-A`

or

`--opcodes`

Example:

```
binwalk -A firmware.bin
```

This function can provide indications of executable machine code within a binary image.

However, an opcode match alone does not automatically determine the complete architecture or function of the code found.

It is an analysis pointer.

### Custom magic file

The following options specify a custom magic file:

`-m`

or

`--magic`

Basic principle:

```
binwalk -m DATEI firmware.bin
```

This allows the signature analysis to be extended with custom or specially adapted detection definitions.

This is not required for normal rescue examinations.

It is a function for specialised binary analyses.

### Automatic extraction

One of the most important functions of Binwalk is:

`-e`

or

`--extract`.

Example:

```
binwalk -e firmware.bin
```

Binwalk tries to extract detected known file types automatically.

The function thus goes beyond a mere signature analysis.

The normal signature scan says:

**“There is probably a known structure here.”**

The extraction function additionally tries:

**“Extract this detected structure.”**

### External extraction programs

During automatic extraction, Binwalk can use external auxiliary programs for certain file types.

This means:

A signature type can be detected even though the external program required for the actual extraction is not available.

**Detection and successful extraction are therefore two different capabilities.**

Detected content does not guarantee that Binwalk can automatically unpack or process it on the specific system.

### Extraction directory

The following options specify a separate target directory for extracted content:

`-C`

or

`--directory`

Basic principle:

```
binwalk -e -C ZIELVERZEICHNIS firmware.bin
```

This is particularly useful in rescue and analysis work so that the files created are deliberately stored separately from the source file.

### Recursive extraction

Binwalk supports what is known as Matryoshka analysis.

The option is:

`-M`

or

`--matryoshka`.

It causes extracted files to be examined again with Binwalk.

In combination with extraction, a nested structure can thus be analysed step by step.

Example principle:

```
binwalk -e -M firmware.bin
```

A firmware image can, for example, contain a compressed archive that in turn contains a file system or further compressed files.

Recursive analysis can follow such levels further.

### Recursion depth

The following options limit the maximum recursion depth of the Matryoshka analysis:

`-d`

or

`--depth`

The documented default value of the 2.x command-line interface is:

**8 levels.**

Example:

```
binwalk -e -M -d 4 firmware.bin
```

This limits the recursive examination to four levels.

A limit can be useful to avoid an unexpectedly extensive or cyclical analysis.

### Limiting the number of extracted files

The following options limit the number of extracted files:

`-n`

or

`--count`

With unknown or very complex images, this can help to control the amount of analysis data generated.

### Limiting the size of extracted files

The following options limit the size of individual extracted files:

`-j`

or

`--size`

This function also serves to make automatic extraction more controllable.

### Carving without extraction programs

With the following options, Binwalk can carve data out of files without running the associated external extraction programs:

`-z`

or

`--carve`

This makes it possible to distinguish between:

**carving**

and

**further automatic extraction or processing**

This separation can be security-relevant with unknown objects of examination.

### Recursive analysis with care

Recursive automatic extraction can produce a very large number of files.

It can also apply external extraction programs to detected content.

With unknown or untrustworthy binary data, it should therefore be decided deliberately whether merely a signature analysis or actually automatic recursive extraction is required.

For an initial examination, a simple call such as the following is considerably more cautious than immediate recursive extraction:

```
binwalk DATEI
```

### Entropy analysis

With the following options, Binwalk calculates the entropy of the examined file:

```
-E
```

or

```
--entropy
```

Example:

```
binwalk -E firmware.bin
```

Put simply, entropy analysis examines how strongly the byte values vary in individual areas, or how high their statistical disorder is.

### Significance of high entropy

Areas with high entropy can indicate, for example:

compressed data

or

encrypted data.

The Binwalk project expressly names entropy analysis as a way of detecting unknown compressed or encrypted areas.

However, it is important to note:

**High entropy does not prove encryption.**

Compressed data can also have very high entropy values.

Other data can also appear statistically similar.

Entropy analysis therefore provides indications of data structures, but no automatic classification of content.

### Fast entropy analysis

The following options use a faster but less detailed entropy analysis:

```
-F
```

or

```
--fast
```

With large images, this can be useful for an initial overview.

### Saving the entropy graph

The following options save the generated entropy graph as a PNG file:

```
-J
```

or

```
--save
```

This enables a visual examination of the entropy curve along the analysed binary image.

### Limiting the scan range

Binwalk does not necessarily have to examine an entire file.

The following options specify a start offset:

```
-o
```

or

```
--offset
```

The following options limit the number of bytes to be examined:

```
-l
```

or

`--length`

This allows an analysis to be concentrated specifically on an area already known to be of interest.

### Base address

The following options add a base value to the offsets output:

`-o`

or

`--base`

This can be relevant in analyses where the file represents a section of a larger address space.

This function is not required for normal signature analyses.

### Changing the byte order

With the following options, Binwalk can reverse groups of bytes before the analysis:

`-g`

or

`--swap`

This function is intended for special binary formats or certain byte arrangements.

It should only be used if the structure of the data format being examined requires it.

### Logging results

The following options write analysis results to a file:

`-f`

or

`--log`

Basic principle:

```
binwalk -f ergebnis.txt firmware.bin
```

This allows a scan to be documented for later evaluation.

### CSV output

The following options produce the log output in CSV format:

`-c`

or

`--csv`

This is particularly useful for structured further processing of analysis results.

### Quiet mode

The following options suppress the normal output:

`-q`

or

`--quiet`

Output to `stdout` is then suppressed.

This is useful, for example, for automated processing or targeted logging.

### Verbose mode

The following options enable more detailed output:

`-v`

or

`--verbose`

In detailed analyses, this can provide additional context information.

### Typical first procedure

With an unknown binary file, as little as possible should initially be changed or automatically extracted.

A sensible first step is:

```
binwalk DATEI
```

This first establishes which structures are detected.

Interesting offsets can then be assessed specifically.

Only when extraction is actually required should the following be used:

```
binwalk -e DATEI
```

With nested content, a recursive analysis can then be considered.

### Example of a firmware examination

A typical workflow can look like this:

1. Back up the original file unchanged.
2. Carry out a signature analysis:

```
binwalk firmware.bin
```

3. Check detected structures and offsets.
4. If required, examine the entropy:

```
binwalk -E firmware.bin
```

5. If extraction is required, use a separate target directory.
6. Then examine extracted content separately.

A clear separation between the original and the analysis data created is thus maintained.

### Binwalk and Hexedit

Binwalk and Hexedit complement each other directly.

Binwalk can report, for example, that a known data structure begins at a specific hexadecimal offset.

This offset can then be examined specifically with Hexedit.

The division of tasks is:

**Binwalk → detect structure automatically**

**Hexedit → view bytes directly and edit them if necessary**

A.15 thus forms the automatic structural analysis and A.16 the manual binary analysis.

### Binwalk and Foremost

Binwalk and Foremost can both detect or extract data on the basis of characteristic file features, but they have different focuses.

Foremost is primarily a file carving tool for recovering or extracting known file types from data streams and images.

Binwalk is primarily an analysis tool for binary files and firmware and shows in particular embedded structures and their offsets.

Binwalk is therefore much more specifically geared to firmware analysis.

### Binwalk and PhotoRec

PhotoRec also works with file signatures.

However, PhotoRec is primarily geared to recovering files from storage devices and images.

Binwalk, by contrast, is used to analyse the internal structure of a binary file or firmware image.

The tools can therefore use similar technical principles but perform different rescue tasks.

### Binwalk is not a malware scanner

A detected executable code area or an embedded file does not automatically mean that malware is present.

Binwalk classifies structures on the basis of its analysis mechanisms.

For a malware examination, the security tools intended for this purpose must be used.

### Binwalk is not a disassembler

The opcode scan can provide indications of executable machine code.

However, Binwalk does not replace a complete disassembler or specialised reverse engineering debugger.

Its strength lies in the identification and structural decomposition of binary images.

### Binwalk is not a data recovery program for damaged hardware

Binwalk can analyse files and images.

However, it is not intended for reading physically faulty storage devices as gently as possible.

With a damaged storage device, as complete an image as possible should first be created with a tool intended for this purpose, such as GNU ddrescue.

Binwalk can then be used on this copy or image.

### Do not change original files unnecessarily

A pure Binwalk signature analysis does not change the examined file.

Automatic extraction, by contrast, creates additional files and directories.

These should be stored on a separate writable working medium.

In forensically relevant examinations, the original should be preserved unchanged.

### Untrustworthy extracted content

Extracted files originate from the examined binary image.

They must therefore basically be treated as untrustworthy.

Successful extraction does not mean that the files created can be executed safely.

In particular, executable programs, scripts or unknown binary files should not be started merely because Binwalk has extracted them successfully.

### External auxiliary programs

Automatic extraction of certain formats can depend on additional programs.

If a required auxiliary program is missing, detection can work while extraction fails.

Such an extraction error therefore does not necessarily mean that the detected signature was wrong.

### False positives

Signature-based analyses can produce false positives.

A random byte sequence can match a known file signature.

Although Binwalk tries to assess results on the basis of its signature logic, an interesting finding should nevertheless be confirmed by further analysis.

This can include, for example:

examining the surrounding bytes,

checking the expected file structure,

successful extraction,

comparison with known formats,

or a manual check with Hexedit.

### Limitations

Binwalk can only detect structures for which corresponding detection rules or signatures exist or which become visible through the analysis functions used.

Unknown proprietary formats can remain undetected.

High entropy does not prove encryption.

A signature match does not automatically prove an intact file.

Successful extraction does not prove that the content is trustworthy.

And the absence of a match does not mean that a binary file contains no embedded data.

### Safety notes

Original files and images should be preserved as unchanged as possible for analysis purposes.

Extracted data should be stored on a separate working medium.

Automatically extracted content must basically be treated as untrustworthy.

Recursive extraction can produce large amounts of data and call external auxiliary programs.

With unknown files, a pure signature analysis should therefore be carried out first.

Binwalk must not be understood as a substitute for malware analysis or storage device rescue.

With physically damaged storage devices, an image should first be created with a suitable rescue tool.

### Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Binwalk opens the **analysis** area.

While the previous tools primarily:

rescue data,

manage storage devices,

check hardware states,

search for malware

or control network access,

Binwalk examines the **internal structural composition of binary files and images**.

Its core functions are:

**Signature analysis → identify embedded structures**

**Offset analysis → determine positions within a binary file**

**Extraction → extract detected content**

**Recursive analysis → examine nested structures further**

**Entropy analysis → make areas with different statistical data structures visible**

Together with A.16 Hexedit, this creates a two-stage analysis environment:

**Binwalk automatically finds interesting structures and offsets.**

**Hexedit then enables direct examination of the bytes concerned.**

Within JUMBO BLOCK Rescue, Binwalk is thus the specialised tool for the structural analysis and decomposition of firmware, binary files and comparable images.

# A.16 Hexedit

**Area:** Analysis
**Program:** Hexedit
**Baseline 20261001:** Hexedit 1.6
**Installed package:** `hexedit 1.6-2`
**Operation:** terminal-based interactive interface
**Task:** direct display, examination and editing of files and storage device content at byte level

## Purpose

Hexedit is an interactive hexadecimal editor for the command line or terminal.

The program displays the content of a file directly as a byte sequence. The data is displayed both:

**in hexadecimal**

and

**as characters**

Hexedit is therefore suitable for the direct examination of binary files, images and – with appropriate access rights – device files.

Unlike Binwalk, Hexedit does not primarily interpret the content on the basis of known file signatures.

Hexedit displays the bytes actually present directly.

The program thus answers in particular the question:

**“Which bytes are located at this specific position?”**

## Area of use in the rescue system

Within JUMBO BLOCK Rescue, Hexedit is used for manual low-level analysis.

Typical tasks are:

examining unknown binary files,

checking file headers,

examining specific offsets,

checking embedded strings,

comparison with Binwalk results,

analysing images,

checking file system or partition structures by experienced users,

and selectively changing individual bytes where this is expressly required.

It is precisely the last function that makes Hexedit a powerful but potentially dangerous tool.

## Basic call

The basic syntax is:

```
hexedit [Optionen] [Datei]
```

Example:

```
hexedit firmware.bin
```

Hexedit opens the specified file in its terminal-based interface.

If a file is opened with write access, changes can be made and then saved.

## Screen layout

The display essentially consists of three information areas:

**Offset**

This shows the position of the displayed data within the file.

**Hexadecimal display**

This shows the individual byte values as two-digit hexadecimal numbers.

For example:

```
4A 55 4D 42 4F
```

**ASCII/character display**

This shows – where displayable – the corresponding characters.

This allows both machine-oriented byte values and readable text components within a binary file to be recognised.

## Hexadecimal values

A byte has 256 possible values.

In hexadecimal, they are represented from:

```
00
```

to:

```
FF
```

A byte therefore requires exactly two hexadecimal digits.

For binary analyses, this representation is much clearer than a long sequence of decimal byte values.

### Offset

The offset designates the position of a byte relative to the beginning of the opened file or data object.

The first byte value is located at offset:

`0`

or, in hexadecimal:

`0x0`.

Offsets are particularly important when another analysis tool has found an interesting structure at a specific position.

Binwalk, for example, outputs the offsets of detected structures.

This position can then be examined directly with Hexedit.

### Navigation

Hexedit allows navigation through the opened file with the usual movement keys.

These include in particular the arrow keys.

Navigation keys for larger movements through the file are also available.

This allows navigation both byte by byte and across larger areas.

### Jumping to the beginning of the file

Use the following to jump to the beginning of the file:

`Ctrl+A`

This is particularly useful for examining a file header.

Many file formats store characteristic identifiers and metadata directly at the beginning of the file.

### Jumping to the end of the file

Use the following to jump to the end of the file:

`Ctrl+E`

This makes it possible, for example, to examine trailers, final data structures or appended data.

### Jumping to a specific offset

A particularly important function is:

`Ctrl+G`

This allows a specific position within the file to be jumped to directly.

This is particularly relevant for working together with Binwalk.

If Binwalk reports an interesting structure at a specific offset, for example, there is no need to scroll manually through the entire file.

The offset can be called up directly in Hexedit.

### Decimal and hexadecimal positions

In low-level analyses, it must always be borne in mind in which number system an offset is specified.

A decimal value such as:

`4096`

This corresponds to the following hexadecimal value:

`0x1000`

Analysis tools can output offsets in different representations.

Before a manual change, it must therefore be clearly established which position is actually meant.

### Search

Hexedit has a search function.

Use the following to search for data within the opened file:

`Ctrl+S`

The search can be used for various analysis tasks, for example to find a known string or a known byte sequence.

### Searching for strings

Despite its binary structure, a file can contain readable text.

Examples can be:

version information,

manufacturer identifiers,

configuration values,

paths,

error messages

or other text information.

The search makes it possible to find such strings specifically.

### Searching for hexadecimal values

Hexedit can also be used to search for byte sequences specified in hexadecimal.

This is relevant when the search is not for readable text but for a known binary signature.

A known magic byte sequence can thus be located directly.

### Continuing a search

After a search, the search can be continued to the next matching occurrence.

This allows several identical or similar structures within a file to be examined.

With large images, this is much more efficient than scrolling through manually.

### Direct editing

Hexedit is not just a viewer.

The program allows the displayed data to be changed directly.

Work can be carried out both in the hexadecimal area and in the character area.

Changes affect the corresponding bytes of the opened file.

This is an essential feature of the program and at the same time its greatest risk.

### Hexadecimal editing

When working in the hex area, byte values can be entered or changed directly.

For example, the byte value shown below can be replaced:

```
41
```

The replacement value is:

```
42
```

This changes exactly one byte value.

Such a change can already be enough to:

invalidate a checksum,

damage a file header,

make a structure unreadable,

change a program function,

or damage a file system or image.

### Character-based editing

In the character area, displayable bytes can be edited as characters.

Changing the visible character also changes the underlying byte value.

The character view is therefore not a separate text copy.

The hex and character views represent the same data.

### Switching between the hex and character areas

Use the following to switch between hexadecimal and ASCII/character editing:

`Tab`

Depending on the task, this makes it possible to work either directly with byte values or with readable characters.

### Changes are initially editable

During editing, data is changed within the Hexedit session.

What then matters is whether these changes are actually written to the file.

There is thus an important separation between:

**change in the editor**

and

**saving to the storage device.**

Before saving, it must therefore be checked once more whether the correct data object and the correct offset have actually been edited.

### Saving changes

Hexedit allows changes made to be saved.

A saved change modifies the underlying file or the opened writable data object.

With a normal working copy, this can be intended.

With an original storage device or forensic image, on the other hand, it can be undesirable or destructive.

### Quitting

Hexedit can be exited via its intended quit function.

If there are changes, it must be decided deliberately whether they are to be saved or discarded.

An analysis should therefore not be ended with thoughtless saving.

### Read-only mode

For rescue and analysis work, the following option is particularly important:

```
-r
```

It opens the file in **read-only mode**.

Example:

```
hexedit -r image.dd
```

This allows the file to be examined without changing it via Hexedit.

For pure analysis, this is the preferred mode of operation.

### Read-only as the default for examinations

If the sole purpose is to:

view data,

check offsets,

examine headers,

compare signatures

or check Binwalk results,

the file should be opened with the following option wherever possible:

```
hexedit -r DATEI
```

This considerably reduces the risk of an accidental change.

### Devices and storage devices

Under Unix/Linux, Hexedit can basically open not only normal files but also accessible device files.

A block device can, for example, be represented under:

```
/dev/...
```

Direct examination of the raw data of a storage device is therefore technically possible.

However, this is considerably riskier than examining a normal working copy.

### Direct write access to block devices

If a block device is opened with write access and then changed, saved changes affect the data on the storage device immediately.

A single incorrectly changed byte can damage critical structures.

The following can be affected, for example:

partition information,

boot structures,

file system metadata,

file contents

or other management data.

Direct changes to an original storage device should therefore only be made by users who know the structure concerned and the consequences exactly.

### Image instead of original

For rescue and analysis work, the safer procedure is:

**Back up the original storage device → create an image or working copy → analyse the copy.**

With a physically problematic storage device, GNU ddrescue can be used to create as complete an image as possible first.

This image can then be examined with Hexedit.

Example:

```
hexedit -r rescue.img
```

This avoids working constantly on the damaged original medium.

### Physically damaged storage devices

Hexedit is not a data recovery tool for faulty hardware.

Scrolling, searching and navigating read data from the storage device.

On a physically failing medium, such additional read accesses can be undesirable.

In the event of read errors or recognisable hardware problems, a controlled backup should therefore first be made with a tool intended for this purpose, such as GNU ddrescue.

Hexedit is then applied to the image created.

### Binwalk and Hexedit

Within JUMBO BLOCK Rescue, Binwalk and Hexedit form a directly related pair of analysis tools.

Binwalk can, for example, establish:

“A known data structure begins at offset X.”

Hexedit can then be used to:

view the bytes at offset X,

examine the area before and after the signature,

check header values manually,

or trace the Binwalk detection independently.

The division of work is:

**Binwalk → automatic structure detection**

**Hexedit → manual byte analysis**

### Example of an analysis workflow

An unknown firmware file is first examined with Binwalk:

```
binwalk firmware.bin
```

Binwalk reports interesting structures and offsets.

The file is then opened in Hexedit, write-protected for safety:

```
hexedit -r firmware.bin
```

Use the following to jump to an offset reported by Binwalk:

```
Ctrl+G
```

There, the actual bytes and any readable characters can be examined directly.

In this way, an automatic analysis finding can be traced manually.

### Hexedit and file signatures

Many file formats begin with characteristic byte sequences.

These are often referred to as:

magic bytes,

magic numbers

or file signatures.

Hexedit makes it possible to view such byte sequences directly.

For example, it can be used to check whether a file actually begins with an expected signature or whether the file extension and the actual binary content differ.

### Damaged file headers

With damaged files, Hexedit can be used to examine the header directly.

This makes it possible to establish whether expected signatures or header values are missing or changed.

However, Hexedit does not repair a damaged header automatically.

A manual repair requires precise knowledge of the file format concerned.

### No automatic interpretation

Hexedit does not automatically know what a particular byte means semantically.

Depending on the file format, a value such as the following can have completely different meanings:

FF

The interpretation must be made by the user on the basis of the data structure concerned.

This distinguishes Hexedit from specialised parsers and file system tools.

### No automatic structure check

Hexedit does not automatically check whether a manual change is logically consistent.

Anyone who changes a length value within a header, for example, must know whether any of the following depend on it:

checksums,

offsets,

index structures,

subsequent fields

or other metadata.

The editor does not prevent semantically incorrect changes.

### Overwriting instead of structural editing

A hex editor basically works on existing byte positions.

Editing a byte is therefore not comparable to the convenient editing of a text document.

With binary formats, inserting, removing or moving data can change all subsequent offsets.

Hexedit is primarily intended for direct byte-oriented editing.

### Character encoding

The character view on the right must not be confused with a complete text interpretation.

Not every byte corresponds to a meaningfully displayable character.

Multi-byte character encodings and complex text formats do not necessarily appear in a simple byte/character view as they would be displayed in a text editor.

For binary analysis, the hex display therefore remains authoritative.

### A search is not a file analysis

Finding a string or byte sequence initially only indicates that this data sequence is present at the position concerned.

It does not automatically reveal what semantic function it has.

A match must always be assessed in the context of the surrounding data structure.

### No malware detection

Hexedit is not a malware scanner.

Suspicious bytes or strings can be examined manually, but Hexedit does not automatically classify them as malware.

Specialised tools such as ClamAV are available within JUMBO BLOCK Rescue for malware checks.

### Not a disassembler

Hexedit merely displays machine code as bytes.

It does not automatically translate these bytes into a complete assembler representation and does not analyse program flow.

Reverse engineering at instruction level would require specialised disassemblers or analysis tools.

### No file carving

Hexedit can make raw data visible and search it.

However, it is not an automatic file carving tool like PhotoRec or Foremost.

If file signatures are searched for within an image, specialised carving or signature tools can be much more efficient.

Hexedit is then suitable for manual checking.

### No automatic data recovery

Hexedit does not reconstruct partition tables.

It does not restore deleted files automatically.

It does not correct file systems.

It does not read damaged storage devices in a fault-tolerant way like GNU ddrescue.

Its strength lies exclusively in the direct checking and – if expressly desired – changing of the existing bytes.

### Typical rescue scenarios

Within JUMBO BLOCK Rescue, Hexedit can be used in particular for:

manual checking of a file header,

examining an offset reported by Binwalk,

searching for known strings,

searching for characteristic byte sequences,

examining a storage device image,

checking raw data,

comparing similar binary files at specific positions,

and, in expert special cases, selectively changing individual bytes of a working copy.

### Safe workflow

For a normal analysis, the following is recommended:

1. Do not change the original file or original storage device.
2. If necessary, create a working copy or an image.
3. First open Hexedit in read-only mode:

```
hexedit -r DATEI
```

4. Locate interesting positions via offsets or the search.
5. Examine data and context.
6. If a change is required, open only a separate working copy with write access.
7. Document changes.
8. Then validate the result with suitable tools.

### Forensic considerations

In an examination relevant to evidence, an original should not be opened and changed with a hex editor capable of writing.

A working copy should be clearly separated from the original.

In addition, hash values before and after an analysis or edit can be used to document changes traceably.

However, Hexedit itself does not replace a complete forensic chain of custody.

## Limitations

Hexedit does not interpret complex data structures automatically.

The user must know what the examined bytes mean.

The program does not prevent logically incorrect changes.

A byte change can make an entire file or data structure unusable.

Hexedit replaces neither a file system repair service nor a malware scanner, disassembler, data recovery tool or specialised firmware parser.

## Safety notes

For pure examinations, Hexedit should be used in read-only mode with the following option:

```
-r
```

Original storage devices should not be opened with write access unnecessarily.

Before manual changes, a backup or working copy should be available.

Direct changes to `/dev/...` devices can destroy data immediately.

With physically damaged storage devices, an image should first be created with a specialised rescue tool.

Offsets must be checked unambiguously before a change.

Particular attention must be paid to the distinction between decimal and hexadecimal position information.

A successful change to individual bytes does not mean that the entire data structure is consistent afterwards.

## Technical classification in JUMBO BLOCK Rescue

Together with Binwalk, Hexedit forms the **analysis** area within JUMBO BLOCK Rescue.

The two programs complement each other:

**A.15 Binwalk**

Automatic analysis of binary files and images for known structures, signatures, embedded content and interesting offsets.

**A.16 Hexedit**

Direct manual examination of the actual bytes at these positions.

This results in a clear analysis chain:

**Binwalk finds.**

**Hexedit shows.**

**The user assesses.**

If a change is technically required, Hexedit can also carry it out. Precisely for this reason, however, read-only mode is particularly important for pure analysis.

Within JUMBO BLOCK Rescue, Hexedit thus represents the lowest manual analysis level of the two programs:

**direct access to the bytes of a file or image.**

# A.17 Nmap

**Area:** Network
**Program:** Nmap – Network Mapper
**Baseline 20261001:** Nmap 7.95
**Installed packages:** `nmap 7.95+dfsg-3`, `nmap-common 7.95+dfsg-3`
**Operation:** command line
**Task:** network discovery, port analysis, service and version detection, operating system detection and extended network analysis

## Purpose

Nmap stands for:

**Network Mapper**

Nmap is a tool for examining and inventorying IP networks.

Its original and still central task is port scanning. In addition, Nmap can, among other things:

detect active hosts,

examine TCP and UDP ports,

classify port states,

identify network services,

determine software versions of services,

estimate operating systems on the basis of their network behaviour,

examine network paths,

and carry out additional network checks via the Nmap Scripting Engine.

Nmap is thus one of the most comprehensive network diagnostic tools in JUMBO BLOCK Rescue.

## Area of use in the rescue system

In a rescue environment, Nmap can be used, for example, to:

establish reachable devices in a local network,

inventory an unknown network range,

determine the open ports of a system,

establish which network service responds on a port,

determine version information of a service,

discover unexpectedly reachable services,

examine the effects of firewalls,

or classify a device more precisely on the basis of its network behaviour.

Nmap works actively.

Depending on the selected scan method, the program sends network packets or connection attempts specifically to the systems being examined.

### Legal and operational prerequisite

Nmap may only be used against systems and networks for whose examination appropriate authorisation exists.

A port scan is not a purely passive observation.

Depending on the options used, numerous packets, complete connections or application-specific requests can be generated.

Some NSE scripts go considerably further and can contain intrusive tests, brute-force checks, exploit tests or even denial-of-service functions.

In production, KRITIS, industrial and third-party networks, it must therefore be clarified before a scan which examinations are permitted.

### Basic syntax

The basic syntax is:

```
nmap [Optionen] ZIEL
```

The target can be specified, for example, as a:

host name,

IPv4 address,

IPv6 address,

address range

or network.

Example for a single authorised host:

```
nmap 192.0.2.10
```

The address here serves solely as a documentation example.

### Default scan

A simple call such as the following carries out a port scan against the target:

```
nmap ZIEL
```

By default, Nmap scans the **1.000 most frequently used ports** of the protocol being scanned.

This is not the same as a scan of all 65.535 TCP ports.

### Host discovery

Before Nmap carries out an extensive port scan, it normally first tries to establish whether a target host is reachable or active.

This process is referred to as:

**Host Discovery**

This means that Nmap does not have to scan every possibly inactive host completely.

### Host discovery only

The following option disables the subsequent port scan:

`-sn`

Example:

```
nmap -sn 192.0.2.0/24
```

Nmap then carries out host discovery and reports the hosts detected as reachable without subsequently running a normal port scan.

This function is suitable for an initial inventory of an authorised network segment.

### “Ping scan” does not mean ICMP only

The option `-sn` is traditionally also referred to as a ping scan.

However, this does not mean that Nmap uses only a conventional ICMP echo request.

Nmap can use various methods for host discovery.

The discovery probes actually used depend, among other things, on the target and the available privileges.

### Local Ethernet

With directly connected Ethernet networks, Nmap uses ARP by default for local targets, or the corresponding neighbour discovery mechanisms with IPv6, because these methods are particularly reliable for local hosts.

This is important for rescue deployments:

A host may not respond to ICMP ping and still be detected by Nmap on the local Ethernet.

### Skipping host discovery

The following option skips normal host discovery:

`-Pn`

Nmap then treats the specified target addresses as if they were active and runs the requested further scans.

Example:

```
nmap -Pn 192.0.2.10
```

This option can be useful if a host blocks discovery probes even though its services are reachable.

### Consequence of `-Pn`

`-Pn` does not mean:

“Host is demonstrably online.”

It means:

**“For the further examination, Nmap should assume that the host is online.”**

With large address ranges, this can considerably extend a scan, because Nmap also examines target addresses that do not exist or cannot be reached.

### Listing targets without scanning

The following option carries out a list scan:

`-sL`

Nmap lists the intended targets without examining them with a normal host or port scan.

This is a useful way of checking before a larger scan.

In particular, it can be used to check whether the specified address range actually corresponds to the intended target range.

### Ports

TCP and UDP use numerical port numbers.

The regular port range extends from:

`1`

to:

`65535`.

Port numbers help to distinguish different network services of a host from one another.

However, an open port initially only shows that a service or application appears reachable via the protocol concerned.

### 1.000 ports by default

Without a specific port specification, Nmap normally scans the 1.000 most frequently used ports.

The selection is based on frequency information in the Nmap services database.

This makes a default scan much faster than a complete scan of all ports.

### Selecting specific ports

The following option specifies particular ports:

`-p`

Example:

```
nmap -p 22,80,443 192.0.2.10
```

Only the specified ports are then examined.

Ranges can also be specified:

```
nmap -p 1-1024 192.0.2.10
```

### All ports

The following option scans ports 1 to 65535:

`-p-`

Example:

```
nmap -p- 192.0.2.10
```

A complete port scan generates considerably more network traffic and accordingly takes more time than a default scan.

### Fast scan

The following option reduces the number of ports examined:

`-F`

If frequency information is present in `nmap-services`, the **100 most frequent ports** are examined instead of the default 1.000.

This is suitable for a quick initial overview.

### Top ports

The following option explicitly specifies the number of ports selected on the basis of their frequency:

```
--top-ports N
```

Example:

```
nmap --top-ports 20 192.0.2.10
```

This examines the 20 ports weighted highest according to the Nmap database.

### Distinguishing TCP and UDP

TCP and UDP ports are separate namespaces. The following two are therefore different examination targets:

```
53/tcp
```

and

```
53/udp
```

A complete network assessment must not automatically ignore UDP services.

### The six port states

Nmap distinguishes six port states:

```
open
```

```
closed
```

```
filtered
```

```
unfiltered
```

```
open|filtered
```

```
closed|filtered
```

These states do not necessarily describe an absolute property of the port.

They describe **how Nmap assesses the port from the perspective of the current scan point and on the basis of the responses received.**

### open

`open` means that an application is actively accepting TCP connections, UDP datagrams or corresponding protocol communication on this port.

An open port is not automatically a security vulnerability.

It initially shows that a network service is reachable there.

It must then be assessed:

Which service is it?

Is it needed?

Is it secured correctly?

Should it be reachable from this network?

### `closed`

`closed` means that the port is reachable and responds to Nmap's test packets, but no application is listening for connections there.

A closed port therefore also provides information:

The target host or network stack is basically reachable.

### `filtered`

`filtered` means that Nmap cannot establish whether the port is open, because a packet filter or other network obstacle prevents the examination.

The cause can be, for example:

a host-based firewall,

an external firewall,

a router rule,

or another filter element.

`filtered` therefore does not automatically mean:

“Port closed.”

### `unfiltered`

`unfiltered` means that the port is reachable, but Nmap cannot determine with the scan method used whether it is open or closed.

This state occurs in particular with the ACK scan.

### `open|filtered`

`open|filtered` means that Nmap cannot distinguish whether a port is open or filtered.

This occurs with scan methods in which an open port may not generate a response.

A typical example is the UDP scan.

## `closed|filtered`

`closed|filtered` means that Nmap cannot determine whether a port is closed or filtered.

This state is used in the IP ID idle scan.

### Results depend on perspective

The Nmap port states apply from the perspective of the system from which the scan was carried out.

A port can, for example, appear as follows:

within a local network: `open`

and

from another network: `filtered`

Firewall rules, routing, NAT and other network elements influence the result.

### TCP SYN scan

The following option enables the TCP SYN scan:

`-sS`

Example:

```
sudo nmap -sS 192.0.2.10
```

This scan is often referred to as a **half-open scan**.

Nmap sends a SYN packet, as normally used at the start of establishing a TCP connection.

If the target port responds with SYN/ACK, this indicates an open port.

If it responds with RST, it is considered closed.

If there is no response or a certain ICMP response, the port can be classified as filtered.

### SYN scan and privileges

Under Unix/Linux, the SYN scan requires the ability to send raw packets.

Appropriate privileges are required for this.

If these are not available, Nmap typically uses the TCP connect scan instead for a normal TCP scan.

### TCP connect scan

The following option enables the TCP connect scan:

`-sT`

Example:

```
nmap -sT 192.0.2.10
```

Nmap uses the operating system's `connect()` system call.

With an open port, a complete TCP connection is thus established.

### Difference between `-sS` and `-sT`

With the SYN scan, the complete TCP connection setup is normally not completed.

With the connect scan, the connection is actually established via the operating system.

The connect scan does not require raw packet privileges, but it is typically less efficient and can be logged on the target system more clearly as normal connection attempts.

### UDP scan

The following option enables a UDP port scan:

`-sU`

Example:

```
sudo nmap -sU 192.0.2.10
```

UDP scans are technically more difficult than TCP scans, because UDP has no TCP-like connection setup.

Many open UDP services do not respond at all to an unsuitable or empty packet.

### UDP results

If a UDP service responds appropriately, Nmap can classify a port as open.

An ICMP port unreachable message typically indicates a closed UDP port.

If Nmap does not receive a clear response, the result can be:

```
open|filtered
```

This means:

The port could be open and not respond – or a filter could be suppressing the communication.

### UDP scans can be slow

UDP scans can take considerably longer than TCP scans.

One reason for this is that missing responses and ICMP rate limits cause waiting times and retries.

For rescue diagnostics, it therefore often makes sense to examine relevant UDP ports selectively first.

### Combining TCP and UDP

UDP scanning can be combined with a TCP scan.

Example:

```
sudo nmap -sS -sU -p T:22,80,443,U:53,161 192.0.2.10
```

Here, the following prefix designates TCP ports:

```
T:
```

The following prefix designates UDP ports:

```
U:
```

### ACK scan

With the following option, Nmap carries out a TCP ACK scan:

```
-sA
```

This scan is not intended to establish whether a port is open.

It serves in particular to obtain information about filter or firewall behaviour.

It can classify ports, for example, as:

```
filtered
```

or

```
unfiltered
```

### Other scan methods

Nmap 7.95 supports other specialised scan methods.

These include:

TCP NULL scan,

FIN scan,

Xmas scan,

Window scan,

Maimon scan,

Idle scan,

SCTP scans

and IP protocol scanning.

These methods are intended for special analysis tasks.

They should not be used indiscriminately.

Their validity depends, among other things, on the target operating system, its network stack and existing filters.

### Service detection

An open port does not initially say reliably which specific service is running there.

The following assignment is merely a typical one:

Port 22 = SSH

Port 80 = HTTP

Port 443 = HTTPS

An administrator can basically operate a service on a different port.

### Version and service detection

With the following option, Nmap enables service and version detection:

`-sV`

Example:

```
nmap -sV 192.0.2.10
```

Nmap sends additional probes to detected ports and analyses their responses.

The aim is to establish as precisely as possible which service is actually reachable.

### Possible `-sV` information

Where the responses allow this, Nmap can determine information such as:

service protocol,

application name,

software version,

host name,

device type,

operating system family,

and CPE identifier.

Not every service provides all this information.

## `nmap-service-probes`

Version detection uses the Nmap database:

`nmap-service-probes`.

It contains probes for various services and rules for detecting and evaluating their responses.

With `-sV`, Nmap therefore does not rely merely on the port number.

### Version detection intensity

The following option controls how intensively Nmap uses various service probes:

`--version-intensity`

The range of values extends from:

`0`

to:

`9`.

The documented default value is:

`7`.

A higher intensity increases the probability of precise identification but can take longer and generate more network communication.

### Light version scan

`--version-light`

This option corresponds to:

`--version-intensity 2`

It carries out a faster, less intensive version detection.

### All version probes

`--version-all`

This option corresponds to:

`--version-intensity 9`.

With it, Nmap tries all suitable version detection probes.

This is correspondingly more extensive.

### Operating system detection

With the following option, Nmap enables operating system detection:

`-O`

Example:

```
sudo nmap -O 192.0.2.10
```

Nmap uses TCP/IP stack fingerprinting for this.

It sends various TCP and UDP probes and examines the responses of the target system in detail.

### OS fingerprint

Nmap compares the observed network characteristics with its fingerprint database:

`nmap-os-db`.

Under suitable conditions, Nmap can derive information from this such as:

manufacturer,

operating system family,

operating system generation,

device type

and, in some cases, CPE information.

### OS detection is an estimate

Operating system detection is not a direct query of the installed operating system.

It is based on its observed network behaviour.

The result can therefore be:

unambiguous,

ambiguous

or indeterminable.

According to Nmap, conditions are particularly good if at least one open and one closed TCP port have been found.

### More aggressive OS guessing

With the following option, Nmap can output near matches more aggressively when fingerprints do not match unambiguously:

```
--osscan-guess
```

Such results are marked with corresponding probability or confidence information.

Such an estimate must not be documented as a reliably determined operating system.

### Aggressive detection

The following option enables several advanced functions together:

```
-A
```

These include in particular:

operating system detection,

version detection,

script scanning

and traceroute.

`-A` therefore does not merely mean:

“more thorough port scan”.

It enables additional active analysis methods.

### Use `-A` deliberately

A call such as the following generates considerably more interaction with the target than a simple port scan:

```
nmap -A ZIEL
```

In sensitive production or KRITIS networks, this option should therefore not be used reflexively.

First, it should be clarified which information is actually needed.

### Nmap Scripting Engine

A particularly powerful component is the:

**Nmap Scripting Engine – NSE.**

NSE uses Lua scripts to automate numerous network tasks.

These include:

extended network discovery,

additional service detection,

information queries,

authentication checks,

malware checks,

vulnerability checks

and other specialised tests.

## Default scripts

The following option runs the default set of NSE scripts:

`-sC`

This corresponds to:

`--script=default`

Example:

```
nmap -sC 192.0.2.10
```

## NSE is not inherently passive or harmless

The official Nmap documentation expressly points out that even some scripts in the `default` category can be considered intrusive.

Furthermore, NSE scripts do not run in a sandbox.

Third-party scripts should therefore only be used if their origin and function are trustworthy or have been checked.

## NSE categories

Nmap assigns NSE scripts to categories.

In Nmap 7.x, these include:

`auth`

`broadcast`

`brute`

`default`

`discovery`

`dos`

`exploit`

`external`

`fuzzer`

`intrusive`

`malware`

`safe`

`version`

`vuln`.

These categories describe different purposes and, in part, also the risk profile of the scripts.

### `safe` does not mean risk-free

The following category contains scripts that are not designed to crash services, exploit vulnerabilities or consume large amounts of resources:

`safe`

Nevertheless, this does not mean that such a scan remains free of side effects under all circumstances.

These scripts also generate active network communication.

### Intrusive scripts

The following category contains scripts that carry a higher risk:

`intrusive`

Such scripts may:

use resources of the target system,

impair a service,

or be perceived as an attack by administrators.

In production systems, such scripts should only be used after explicit approval.

### `vuln`

The following category contains scripts for checking for certain known vulnerabilities:

`vuln`

The result of such a script must be assessed technically.

A script does not replace a complete vulnerability analysis.

### `exploit`, `brute` and `dos`

Particular care applies to categories such as:

`exploit`

`brute`

and

`dos`.

They can contain specifically exploiting, authentication-related or service-loading functions.

Such categories are not part of a normal rescue network diagnosis.

Their mere technical availability is no approval for their use.

### Single NSE script

A specific script can be selected with:

```
--script
```

Basic principle:

```
nmap --script SCRIPTNAME ZIEL
```

Its function should be checked before execution.

### Help for an NSE script

The following option displays the documentation of a script:

```
--script-help
```

Example principle:

```
nmap --script-help SCRIPTNAME
```

This is expressly advisable before using an unknown NSE script.

### Script arguments

Some NSE scripts accept additional parameters.

They are passed with:

```
--script-args
```

The required parameters depend on the script and should only be used in accordance with the documentation of the script concerned.

### Output

Nmap can display results directly in the terminal or save them to files.

For documented rescue deployments, saving the scan results is particularly useful.

### Normal output

The following option saves the normal Nmap output:

```
-oN DATEI
```

Example:

```
nmap -oN scan.txt 192.0.2.10
```

This format is well suited to human evaluation.

### XML output

The following option produces XML output:

```
-oX DATEI
```

Example:

```
nmap -oX scan.xml 192.0.2.10
```

XML is particularly suitable for machine processing.

### Grepable output

The following option produces the older grepable output format:

```
-oG DATEI
```

The Nmap documentation now describes this format as **deprecated**.

For new automated processing, XML is the more suitable structured output.

### Several formats at once

With the following option, Nmap saves the results in several formats at once:

```
-oA BASISNAME
```

These are:

normal output,

XML

and grepable format.

Example:

```
nmap -oA netzscan 192.0.2.10
```

Correspondingly, files with the following extensions are created:

```
.nmap
```

```
.xml
```

```
.gnmap.
```

### Verbose output

The following option makes the output more detailed:

```
-v
```

Several v increase the level of detail:

```
-vv
```

Nmap then displays additional information during the scan and, for example, reports open ports found earlier.

### Open or potentially open ports only

The following option restricts the output to ports that are at least potentially open:

```
--open
```

This can make extensive results clearer.

However, it changes the display and should not lead to the existence of other port states being forgotten.

### Network interfaces and routes

With the following option, Nmap displays the network interfaces and routing information it has detected and then exits:

```
--iflist
```

This is useful if it is unclear which interface or route Nmap would use for a scan.

### Timing

Nmap offers various timing mechanisms for controlling scan speed.

A well-known short form are the timing templates:

`-T0`

to:

`-T5`.

They range from very slow to very aggressive timing settings.

### Do not confuse timing templates with scan quality

A higher timing template does not automatically mean:

“better scan”.

More aggressive timings can save time in fast and reliable networks.

In slow, lossy or heavily filtered networks, on the other hand, they can lead to less reliable results or unnecessary load.

### Scope affects run time

The scan time depends, among other things, on:

number of targets,

number of ports,

TCP or UDP,

firewalls,

packet loss,

rate limits,

version detection,

OS detection,

NSE scripts,

and selected timing parameters.

A complete UDP scan, for example, can take considerably longer than a small TCP scan.

### Influence of firewalls

A firewall can influence Nmap results considerably.

A filtered port often provides no direct response.

Nmap may then have to wait for timeouts and repeat probes.

Heavily filtered networks can therefore slow a scan down considerably.

### GUFW and Nmap

The local GUFW/UFW configuration of JUMBO BLOCK Rescue can also influence network communication.

If unexpected scan results occur, it should therefore also be checked:

which interface is used,

which route is active,

and which local firewall configuration applies.

The firewall should not be disabled across the board just to carry out a scan.

### Nmap and ARP Scan

A.20 ARP Scan is geared specifically to detecting devices in the local layer 2 network.

Nmap also has powerful local host discovery mechanisms.

The tools therefore partly overlap but have different focuses.

Nmap additionally offers extensive port, service, operating system and script analysis.

### Nmap and Wireshark

Nmap and Wireshark perform fundamentally different tasks.

**Nmap actively generates network probes and evaluates responses.**

**Wireshark records visible network traffic and analyses packets.**

The two tools can complement each other excellently.

An Nmap scan can, for example, be observed in parallel with Wireshark in order to examine the packets actually sent and received.

### Nmap and TCPDump

TCPDump can also record the network traffic generated by Nmap.

This makes it possible, particularly in a terminal environment, to trace which packets are transmitted during a scan.

### Nmap and MTR

MTR primarily examines the network path and reachability across several hops.

Nmap, by contrast, primarily examines hosts, ports, services and their network characteristics.

The two tools thus complement each other in diagnosing complex network problems.

### Nmap and Dig

Dig is used for targeted DNS diagnostics.

Although Nmap can detect DNS-related services and carry out additional DNS checks via NSE, it does not replace a specialised DNS query tool.

## Typical rescue workflow

A controlled network analysis should be carried out step by step.

**1. Determine the network range unambiguously**

Before every scan, it must be established which network may be examined.

**2. Check the target range**

With larger ranges, a list scan can be useful first:

```
nmap -sL NETZ
```

**3. Detect hosts**

Then:

```
nmap -sn NETZ
```

**4. Examine an interesting host specifically**

For example:

```
nmap HOST
```

**5. Examine required ports specifically**

```
nmap -p PORTLISTE HOST
```

**6. Service detection if required**

```
nmap -sV HOST
```

**7. Further analysis only where justified**

For example, OS detection or selected NSE scripts.

In this way, the maximum possible scan intensity is not automatically used against every target.

## Example: unknown device in the local network

If an unknown host is found, a normal port scan can first show which common TCP ports are reachable.

The following call can then provide additional information about the services responding there:

```
nmap -sV HOST
```

With appropriate authorisation and technical need, the following call can then attempt an operating system estimate:

```
sudo nmap -O HOST
```

The results are then assessed together.

### Example: service not reachable

If a particular service is not reachable, its port can be examined specifically:

```
nmap -p PORT HOST
```

A result of:

```
closed
```

This means something different from a result of:

`filtered`.

`closed` shows that the port is reachable but no service is listening there.

`filtered` means that, because of filtering, Nmap cannot make a clear statement about open or closed.

This distinction is essential for troubleshooting.

### Example: UDP service

For a known UDP service, UDP should be examined specifically:

```
sudo nmap -sU -p PORT HOST
```

A normal TCP scan would not correctly capture a service offered exclusively via UDP.

### Documenting scan results

In technical deployments, relevant scans should be saved.

Example:

```
nmap -sV -oA analyse HOST
```

Both human-readable and structured output are then available for later evaluation.

Time, scan source, target range and options used should also be documented.

### Limitations

Nmap sees a target from the network perspective of the scanning host.

Firewalls, NAT, routing and packet filters can influence results.

An `open` port does not prove that the service is secure or insecure.

A `filtered` port does not prove that no service is running there.

Service detection can be wrong.

OS detection is a fingerprint-based estimate.

NSE results must be assessed in the context of the script concerned.

An inconspicuous Nmap scan does not prove the complete security of a system.

### Safety notes

Only scan authorised target systems and network ranges.

Before larger scans, check the target range and netmask carefully.

Do not put unnecessary load on production and KRITIS systems with aggressive scan options.

UDP scans can take a long time and generate considerable traffic.

`-A` enables several additional analysis methods and should be used deliberately.

Check NSE scripts on the basis of their documentation and category before use.

Do not treat scripts from the `intrusive`, `brute`, `exploit` or `dos` categories as normal diagnostic tools.

Only use third-party NSE scripts from a trustworthy source and after checking.

Scan results potentially contain sensitive information about the infrastructure examined and should be protected accordingly.

### Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Nmap opens the extensive **network** area.

Its core tasks can be divided into five levels:

**1. Host discovery**
Which systems are reachable?

**2. Port scanning**
Which TCP or UDP ports appear open, closed or filtered?

**3. Service detection**
Which network services respond on these ports?

**4. OS detection**
Which operating system or device class matches the observed network behaviour?

**5. NSE**
Which additional authorised network checks are to be carried out specifically?

Nmap thus provides an active view of the network environment.

The other network tools of JUMBO BLOCK Rescue complement this perspective:

**A.18 Wireshark – graphical packet capture and protocol analysis**

**A.19 TCPDump – terminal-based packet capture**

**A.20 ARP Scan – local layer 2 device detection**

**A.21 MTR – path, latency and reachability diagnostics**

**A.22 Dig – DNS diagnostics**

**A.23 Whois – query of registration information**

**A.24 Chromium – web-based access to network and administration interfaces**

Nmap takes on the role of the active network mapper:

**Find hosts → examine ports → identify services → classify systems → enable targeted further analysis.**

# A.18 Wireshark

**Area:** Network
**Program:** Wireshark
**Baseline 20261001:** Wireshark 4.4.18
**Installed packages:** `wireshark 4.4.18-0+deb13u1`, `wireshark-common 4.4.18-0+deb13u1`
**Operation:** graphical user interface
**Task:** capture, decoding and detailed analysis of network packets

## Purpose

Wireshark is a network protocol analyser.

The program can capture network traffic, open saved packet captures and analyse the network protocols they contain in detail.

While Nmap examines a network actively by generating packets and requests specifically, Wireshark works primarily as an observation and analysis tool.

Its core tasks are:

**Capture packets**

**Decode protocols**

**Examine communication sequences**

**Filter packets**

**Analyse connections and data streams**

**Generate network statistics**

**Save capture files and examine them again**

Within JUMBO BLOCK Rescue, Wireshark is thus the central graphical tool for packet and protocol analysis.

## Area of use in the rescue system

Wireshark can be used in particular to:

observe the network communication of a system,

analyse connection problems,

examine DNS queries,

trace TCP connection setup and teardown,

analyse DHCP communication,

examine ICMP traffic,

make ARP communication visible,

analyse HTTP and other application protocols,

detect unexpected network communication,

trace Nmap scans at packet level,

narrow down firewall problems,

and evaluate existing PCAP or PCAPNG files offline.

## Wireshark is not a port scanner

Wireshark and Nmap perform different tasks.

**Nmap**

This actively generates network probes and evaluates the responses.

**Wireshark**

This captures network packets and decodes their content.

Wireshark therefore does not automatically carry out a port scan.

However, it can make the network traffic of an Nmap scan visible.

## Live capture and offline analysis

Wireshark has two fundamentally different modes of operation.

**Live capture**

Network packets are captured directly from a network interface.

**Offline analysis**

An existing capture file is opened and analysed.

No active network connection is required for offline analysis.

This means that previously saved network captures can also be examined later within JUMBO BLOCK Rescue.

## Capture interfaces

For a live capture, it must be specified from which network interface data is to be captured.

Depending on the system, the following may be present, for example:

Ethernet interfaces,

Wi-Fi interfaces,

loopback,

virtual interfaces,

VPN interfaces

and other capture interfaces provided by the operating system.

Before starting a capture, check which interface actually carries the network traffic of interest.

### Several interfaces

Wireshark can capture from several network interfaces at the same time.

This can be useful, for example, if a system communicates simultaneously via Ethernet and another network interface.

However, several interfaces also increase the amount of data captured.

For a targeted diagnosis, only the interface or interface group actually needed should therefore be selected.

### Capture privileges

Capturing network packets under Linux requires appropriate permissions.

For security reasons, Wireshark separates the actual packet capture from the extensive graphical analysis.

The following component in particular is used for capture:

`dumpcap`

The specific permissions configuration depends on the distribution and installation.

It must therefore not be inferred from the mere package list of the baseline which user groups or individual capture permissions have been preconfigured for JUMBO BLOCK Rescue.

### Dumpcap

Dumpcap is part of the Wireshark tool family and is intended for the actual capture of network traffic.

It can capture network packets from live interfaces and write them to capture files.

The default output format of Dumpcap is PCAPNG.

Wireshark uses the capture infrastructure to separate packets more securely from the actual graphical protocol analysis.

### Promiscuous mode

With Ethernet captures, an interface can be operated in promiscuous mode.

The interface is configured in such a way that it does not only accept frames addressed directly to its own MAC address.

However, this does not automatically mean that all traffic of a switched network becomes visible.

### Limitations in switched networks

In a modern Ethernet switch, unicast traffic is normally only forwarded to the port where the destination concerned is located.

A computer running Wireshark therefore typically sees:

its own traffic,

broadcasts,

relevant multicasts

and traffic that the network infrastructure actually forwards to its port.

To analyse other unicast traffic specifically, a correspondingly configured network infrastructure may be required, for example a mirror/SPAN port or another authorised measuring point.

Promiscuous mode alone does not bypass this network architecture.

### Wi-Fi and monitor mode

With supported Wi-Fi hardware and suitable drivers, monitor mode can be relevant for certain analyses.

Wi-Fi traffic can then be captured at a different level from that of a normal connection to an access point.

Whether this works depends on:

Wi-Fi hardware,

driver,

operating system,

interface configuration

and the selected channel.

Installing Wireshark alone therefore does not guarantee a working Wi-Fi monitor mode.

### Starting a capture

A live capture is started in the graphical interface via the desired interface.

During the capture, Wireshark continuously displays the incoming packets and decodes detected protocols.

The capture can then be stopped and saved.

## The three central view areas

The classic Wireshark view essentially consists of:

**Packet List**

List of captured packets.

**Packet Details**

Decoded protocol structure of the selected packet.

**Packet Bytes**

Actual byte values of the packet.

This allows a packet to be examined on three levels at the same time:

overview,

protocol structure,

raw data.

## Packet List

For each packet, the packet list shows typical information such as:

packet number,

time,

source,

destination,

protocol,

length

and a summarised description.

The columns actually displayed can be customised.

## Packet Details

In the Packet Details area, Wireshark breaks down the selected packet according to the detected protocols.

With an Ethernet/IP/TCP packet, for example, various levels can be displayed separately:

Ethernet,

Internet Protocol,

Transmission Control Protocol,

application protocol.

The individual levels can be expanded and examined in detail.

### Packet Bytes

Packet Bytes shows the actual bytes of the selected packet.

This makes it possible to trace the raw data on which the decoding is based.

This view is particularly relevant with:

unknown protocol fields,

faulty packets,

proprietary data

or detailed protocol analysis.

### Protocol dissectors

Wireshark uses what are known as dissectors to detect network protocols and interpret their fields.

A dissector knows the structure of a particular protocol and can display its fields in an understandable form.

Wireshark supports a very large number of such protocol dissectors.

### A port number is not the same as a protocol

Wireshark uses various information for protocol detection.

A typical port number is an important indication, but a service can be operated on an unusual port.

In certain cases, manual assignment may therefore be necessary.

### Decode As

**Decode As** can be used to instruct Wireshark to interpret certain network traffic as a selected protocol.

This is useful, for example, if a known service is operated on an unusual port and is therefore not detected correctly automatically.

Decode As does not change the captured network packets.

It changes their interpretation within Wireshark.

### Capture filters and display filters

One of the most important fundamentals of Wireshark is the clear separation between:

**Capture Filter**

and

**Display Filter**.

They are two different filter systems with different syntax and different effects.

They must not be confused with each other.

## Capture filter

A capture filter is applied **before or during packet capture**.

It decides which packets are included in the capture file at all.

Packets excluded by the capture filter are then not in the capture.

They can therefore not be made visible again later by changing a display filter.

## Capture filter syntax

Capture filters use the filter syntax of the underlying packet capture library.

Typical filter elements refer, for example, to:

hosts,

networks,

ports

and protocols.

Examples:

```
host 192.0.2.10
tcp port 443
udp port 53
net 192.0.2.0/24
```

The addresses used here serve solely as documentation examples.

## Use capture filters sparingly

Capture filters reduce:

amount of data,

storage requirement

and analysis effort.

However, they involve a risk:

A capture filter that is defined too narrowly can discard precisely the packets that are needed later for fault analysis.

If sufficient storage space is available, it can therefore be useful with unclear fault patterns to capture more broadly first and then narrow down with display filters.

### Display filter

Display filters are applied to packets that have already been captured or opened.

They only decide which packets Wireshark currently displays.

Packets that do not match are not deleted from the capture file.

The filter can be changed or removed at any time.

### Example `tcp`

The following display filter shows packets in which Wireshark has decoded TCP accordingly:

```
tcp
```

### Example `dns`

The following display filter shows packets that Wireshark decodes as DNS:

```
dns
```

### Example IP address

A filter can refer to specific protocol fields.

For example:

```
ip.addr == 192.0.2.10
```

This allows IPv4 packets to be filtered in which the specified address occurs according to the field comparison.

### Example TCP port

Example:

```
tcp.port == 443
```

This allows TCP traffic with the port concerned to be displayed.

### Source and destination direction

Display filters can take source and destination into account separately.

For example:

```
ip.src == 192.0.2.10
```

Or:

```
ip.dst == 192.0.2.10
```

This allows the direction of communication to be examined specifically.

### Logical combinations

Display filter conditions can be combined logically.

For example:

```
tcp && ip.addr == 192.0.2.10
```

This displays TCP packets that additionally match the specified IP condition.

Complex display filters enable very precise narrowing down of large packet captures.

### Filters do not change the data

A display filter does not delete packets.

It only changes the current display.

This is crucial for the analysis:

**Captured** describes the packets actually captured.

**Displayed** describes the packets displayed on the basis of the current display filter.

### Finding packets

Wireshark offers functions for finding packets specifically.

Depending on the function selected, the search can be for, for example:

display filter criteria,

hex values,

strings

or other packet characteristics.

This makes it easier to examine large capture files.

### Marking packets

Interesting packets can be marked.

This makes it quicker to find relevant points again within an extensive capture.

Marks do not change the actual network traffic.

### Time display

Wireshark can display packet times in different ways.

The following, for example, can be relevant for the analysis:

absolute time of day,

time since the start of the capture,

time since the previous packet

or other time references.

The right time display depends on the fault pattern.

### Time stamps

The time stamps of a capture come from the capture environment concerned.

Their accuracy depends, among other things, on:

operating system,

capture mechanism,

hardware,

driver

and time configuration.

When comparing several systems, it must therefore be taken into account whether their clocks were sufficiently synchronised.

### TCP connection setup

Wireshark is excellently suited to examining a TCP connection setup.

The classic sequence consists of:

SYN

SYN/ACK

ACK.

If an expected response is missing, the packet capture can provide important indications of the point at which communication fails.

### TCP connection teardown

FIN and RST processes can also be examined.

This makes it possible to establish, for example, whether a connection was:

terminated in an orderly manner,

reset by one side

or aborted unexpectedly.

### TCP retransmissions

Wireshark can display various TCP analysis indications.

These can include, for example, detected retransmissions.

However, such indications must be assessed in context.

A situation classified as a retransmission does not automatically mean that the cause clearly lies in the network.

Capture losses, asymmetric capture or other effects can influence the analysis.

### Checksums

Wireshark can examine checksums of various protocols.

With local captures, however, apparently faulty checksums can occur even though the network communication actually transmitted is correct.

An important cause is **checksum offloading**.

The network card calculates certain checksums only after the point at which Wireshark has captured the packet in the host.

A displayed checksum error is therefore not automatically an error actually transmitted on the network.

### DNS analysis

DNS traffic can be filtered with:

```
dns
```

This makes it possible to examine, for example:

DNS queries,

responses,

queried names,

response addresses,

error codes

and response times.

A.22 Dig is also available for targeted active DNS queries.

### DHCP analysis

Wireshark can display DHCP communication in detail.

This makes it possible, for example, to examine whether a client:

requests an address,

reaches a DHCP server,

receives an offer

and is assigned a configuration.

This is particularly helpful with systems without the expected network configuration.

### ARP analysis

ARP traffic can be displayed with:

```
arp
```

This makes it possible to examine local IPv4 address resolution at Ethernet level.

This complements A.20 ARP Scan:

**ARP Scan generates targeted ARP requests for device detection.**

**Wireshark can make the ARP packets actually transmitted visible.**

### ICMP analysis

IPv4 ICMP traffic can be examined with:

```
icmp
```

This can make visible, for example, echo requests, echo replies and certain error messages.

Corresponding ICMPv6 protocol fields exist for IPv6.

### HTTP

Unencrypted HTTP traffic can be decoded in detail – provided that it is visible in the capture.

Requests, responses, headers and other protocol information, among other things, can be examined.

At the same time, this means:

Unencrypted network protocols can transmit sensitive information openly.

Capture files must therefore be protected accordingly.

### TLS

With TLS-encrypted communication, Wireshark can analyse the visible TLS protocol information.

However, the encrypted application data does not become visible in plain text merely because Wireshark has captured the traffic.

Decryption requires suitable keys or session secrets and appropriate technical prerequisites.

### Encryption remains encryption

Wireshark does not automatically “crack” a TLS connection.

Without the required cryptographic information, the encrypted payload remains protected.

Nevertheless, metadata of the communication remains visible, for example:

IP addresses,

ports,

packet sizes,

times

and certain unencrypted protocol information.

### Follow Stream

Wireshark can reconstruct related communication streams and display them in a coherent view.

Supported streams can be followed via:

**Analyze → Follow**

These include TCP, UDP, TLS, HTTP, HTTP/2, QUIC, WebSocket and other stream types supported by Wireshark.

### Follow TCP Stream

**Follow TCP Stream** displays the payload of a selected TCP connection in its communication order.

This makes it much easier to recognise which data was exchanged within a connection than looking at individual packets in isolation.

At the same time, Wireshark sets a matching display filter for the stream concerned.

### Follow Stream does not change the capture

Follow Stream reconstructs and filters the display.

The original packets of the capture file are preserved.

The stream view can therefore be closed and the analysis then continued at packet level.

### Statistics

Wireshark has extensive statistics functions.

They are located in particular in the menu:

**Statistics**

This allows large capture files to be examined in summary form.

### Protocol Hierarchy

The Protocol Hierarchy shows which protocols were detected within a capture and what proportion of the captured traffic they account for.

This makes it possible to establish quickly which protocols dominate a capture.

### Endpoints

The Endpoints statistics summarise individual communication endpoints.

Depending on the protocol level, the following can be displayed, for example:

MAC addresses,

IPv4 addresses,

IPv6 addresses

and other endpoints.

This is helpful for identifying particularly active systems within a capture.

### Conversations

The Conversations view summarises communication relationships between two endpoints at a time.

It shows statistical values for individual connections or communication pairs.

This makes it possible to recognise, for example:

which systems communicate with each other,

how many packets were transmitted,

which amounts of data were generated,

and when a communication began.

### I/O Graphs

Wireshark can display network activity graphically over time.

I/O graphs are helpful for making visible, for example:

traffic peaks,

periodic communication,

interruptions

or temporal relationships.

## Expert Information

Wireshark provides what is known as Expert Information.

This function summarises certain conspicuous or analysis-relevant protocol events.

These can include indications of:

errors,

warnings,

unusual sequences

and other protocol events.

## Expert Information is not a final verdict

Expert Information makes it easier to find interesting events.

However, the entries are analysis pointers.

They do not automatically prove a network defect or security incident.

Every finding must be assessed in the context of the protocol and capture concerned.

## Capture files

Wireshark can save network captures and open them again later.

The central formats are in particular:

**PCAP**

and

**PCAPNG**.

## PCAPNG

PCAPNG is the modern default format of the Wireshark capture tool family.

Compared with conventional PCAP, it can store additional metadata.

This can include, for example, information about capture interfaces and comments.

PCAPNG is particularly relevant for several interfaces captured at the same time.

## PCAP

PCAP is an older and very widespread capture format.

It is supported by numerous network tools.

It is therefore well suited for exchange with other analysis programs, but does not offer all the metadata options of PCAPNG.

### Capture files contain sensitive data

A capture can contain considerably more information than an ordinary diagnostic log.

Depending on the network traffic, it can contain:

internal IP addresses,

host names,

DNS queries,

communication partners,

protocol data,

unencrypted content,

authentication information with insecure protocols

and other sensitive operating data.

Capture files must therefore be protected accordingly.

### Do not pass on capture files uncontrolled

Before a capture is passed on, it must be checked which information it contains.

This applies in particular to:

corporate networks,

KRITIS environments,

production systems,

personal communication

and security-relevant infrastructures.

A PCAP file should not be regarded as uncritical merely because it contains “only network packets”.

### Capture File Properties

Wireshark can display properties of a capture file.

These include:

file path,

file size,

cryptographic hash values,

file format,

encapsulation,

time of the first and last packet,

capture information,

interfaces

and statistical information.

This is helpful for the technical documentation of a capture.

### Capture comments

PCAPNG can store comments on the entire capture file.

This allows, for example, analysis context or technical notes to be documented.

If a capture file is used as evidence, changes to metadata or comments must be taken into account and documented accordingly.

### Ring buffer

For long-term captures, Wireshark or its capture engine can use several capture files and rotate them.

A ring buffer prevents a single capture file from growing indefinitely.

Depending on the configuration, older capture files can be overwritten when a specified number is reached.

This must be configured deliberately for long-term diagnostics.

### Stopping automatically

A capture can be ended automatically on the basis of various conditions.

These can include:

amount of data captured,

elapsed time,

number of packets

or corresponding capture conditions.

This makes it possible to prevent an unattended capture from running indefinitely.

### Packet loss during capture

Wireshark can only analyse packets that have actually been captured.

With high network load, insufficient system performance or an unsuitable capture configuration, packets can be lost during capture.

Missing packets can then distort the protocol analysis.

In critical examinations, the quality of the capture itself must therefore also be assessed.

### Snap length

The snapshot length, or snaplen, determines how many bytes of a packet are captured.

If only part of each packet is stored, this reduces the storage requirement and amount of data.

At the same time, payload needed for later analysis may be missing.

For an unknown cause of fault, a shortened snaplen should therefore only be used deliberately.

### Name resolution

Wireshark can partly resolve numerical addresses and ports into names.

This improves readability.

However, name resolution can generate additional network communication and influence the interpretation.

In a precise analysis, it should therefore always remain recognisable which actual numerical address or port number lies behind a displayed name.

### Exporting packets

Wireshark can export selected or filtered packets to new capture files.

This allows a smaller relevant excerpt to be created from a large capture.

The original should be preserved unchanged if a later complete analysis might be required.

### Exporting packet bytes

Selected packet bytes can also be exported.

This is useful, for example, if a particular binary content is then to be examined with other analysis tools.

### Export Objects

For certain supported protocols, Wireshark can extract transmitted objects via **Export Objects**.

Such extracted files come directly from the observed network traffic.

They must therefore basically be treated as untrustworthy.

An extracted file should not be executed merely because Wireshark has reconstructed it successfully.

### Capinfos

The Wireshark tool family includes:

`capinfos`.

Capinfos reads capture files and outputs information and statistics about them.

This can be used, for example, for a quick terminal-based inventory of a capture file.

### Editcap

`editcap` can read capture files, process selected packets and create a new capture file.

Its tasks include cutting out or selecting capture ranges and converting capture formats.

In important examinations, the original should be preserved.

### Mergecap

`mergecap` can merge several capture files into one common file.

By default, the packets are merged chronologically on the basis of their time stamps.

This is useful, for example, if several partial captures are to be analysed together.

### Wireshark and TCPDump

A.19 TCPDump performs part of the same basic task as Wireshark:

capturing and examining network packets.

However, the focus differs.

**Wireshark**

This offers extensive graphical protocol analysis.

**TCPDump**

This is geared to fast terminal-based packet capture and filtering.

A PCAP file created with TCPDump can then be examined in detail with Wireshark.

The two programs thus complement each other directly.

### Wireshark and Nmap

Nmap actively generates network traffic.

Wireshark can observe this traffic.

A combined diagnostic workflow can therefore be:

Start a Wireshark capture.

Carry out an Nmap scan.

Stop the capture.

Then examine the Nmap probes and responses in Wireshark.

This makes it possible to trace why Nmap has classified a port as `open`, `closed` or `filtered`, for example.

### Wireshark and GUFW

GUFW/UFW controls network traffic on the basis of firewall rules.

Wireshark analyses packets that are visible at the capture point concerned.

The two tools thus perform different tasks:

**GUFW/UFW → control network access**

**Wireshark → observe and analyse network communication**

With firewall problems, both perspectives together can be helpful.

### Wireshark and Dig

Dig generates targeted DNS queries and displays their DNS responses in a structured way.

At the same time, Wireshark can observe the actual DNS packet traffic.

This makes it possible to examine differences between, for example:

application result

and

actual network traffic.

### Wireshark and MTR

MTR generates diagnostic traffic for examining the network path.

Wireshark can make this traffic visible at packet level.

MTR provides the clear path and latency perspective, Wireshark the detailed packet perspective.

### Typical rescue workflow

A controlled use of Wireshark can be carried out as follows:

**1. Define the problem**

For example:

DNS does not work,

connection is reset,

service does not respond,

DHCP does not provide an address.

**2. Determine the correct interface**

Select the interface over which the communication concerned actually runs.

**3. Capture filter only if required**

With sufficient storage and an unclear cause, do not restrict unnecessarily at first.

**4. Start the capture**

**5. Reproduce the problem**

Only carry out the action required for the diagnosis.

**6. Stop the capture**

This keeps the capture manageable.

**7. Apply display filters**

For example:

```
dns
```

```
tcp
```

```
arp
```

Alternatively, a targeted host/port filter.

**8. Examine relevant packets**

Look at Packet List, Packet Details and Packet Bytes together.

**9. Use streams and statistics**

If required, use Follow Stream, Conversations, Endpoints or Expert Information.

**10. Save the capture**

In deployments relevant to documentation, keep the original capture unchanged and carry out further processing on a copy.

## Example: DNS problem

A capture is started.

The problematic name resolution is then triggered.

The capture is then stopped.

Display filter:

```
dns
```

It can now be checked:

Was a DNS query sent at all?

To which server?

Did a response come back?

Which response code was returned?

How long did the response take?

This makes it possible to distinguish whether the problem lies before the DNS query, on the transport route or within the DNS response.

### Example: TCP service not reachable

Display filter:

```
tcp.port == PORT
```

It can then be examined:

Is a SYN sent?

Does a SYN/ACK come back?

Does an RST come back?

Is there no response at all?

This allows an abstract “connection does not work” to be traced back to specific packet events.

### Example: DHCP problem

The DHCP communication is captured during connection setup.

It can then be checked which steps of the DHCP sequence are actually visible.

This makes it possible to distinguish, for example, between:

client sends no request,

server does not respond,

response does not reach the client,

or configuration is not adopted for another reason.

## Limitations

Wireshark only sees packets that are actually available at the selected capture point.

Promiscuous mode does not automatically make all traffic of a switched network visible.

Encrypted payload remains encrypted without suitable cryptographic information.

Capture losses can distort analyses.

Checksum offloading can produce apparent checksum errors.

A display filter only shows a subset of the packets present.

A capture filter can permanently exclude packets from the capture.

Protocol dissectors cannot always interpret unknown or unusual protocols correctly.

Wireshark is not an automatic intrusion detection system and not a complete malware scanner.

### Safety notes

Only capture network traffic in environments for which appropriate authorisation exists.

Capture files can contain highly sensitive information.

Unencrypted protocols can transmit access credentials or other confidential content openly.

Store capture files in an appropriately protected manner.

Do not confuse capture filters and display filters.

With long-term captures, take account of storage requirements and ring buffer configuration.

With forensically relevant captures, preserve the original unchanged.

Basically treat extracted network objects as untrustworthy.

Do not interpret apparent checksum errors as real network errors without checking for offloading effects.

A packet capture shows the perspective of the selected capture point and not necessarily the complete communication path.

### Technical classification in JUMBO BLOCK Rescue

Within the network area of JUMBO BLOCK Rescue, Wireshark forms the central **graphical packet and protocol analysis**.

The following comparison applies:

**A.17 Nmap**

This actively examines hosts, ports and services.

By contrast:

**A.18 Wireshark**

This observes the network packets actually visible at the capture point and breaks them down to protocol and byte level.

Its essential analysis levels are:

**Capture → capture network traffic**

**Packet List → overview of the communication sequence**

**Packet Details → examine protocol fields**

**Packet Bytes → check actual raw data**

**Display filter → isolate relevant communication**

**Follow Stream → reconstruct related communication**

**Statistics → summarise larger amounts of data**

Together with Nmap, this creates a particularly powerful combination:

**Nmap generates a controlled active examination.**

**Wireshark shows what is actually communicated over the network in the process.**

The tool that follows adds the same basic packet perspective as a lightweight terminal-based solution:

### A.19 TCPDump

In JUMBO BLOCK Rescue, Wireshark thus takes on the role of the detailed graphical network protocol analyser for live diagnostics and saved capture files.

# A.19 TCPDump

**Area:** Network
**Program:** TCPDump
**Baseline 20261001:** TCPDump 4.99.5
**Installed package:** `tcpdump 4.99.5-2`
**Operation:** command line
**Task:** terminal-based capture, filtering and analysis of network packets

## Purpose

TCPDump is a command-line-based tool for capturing and examining network traffic.

It can capture packets directly from network interfaces, select them on the basis of defined filters, output information about the packets in the terminal and save network traffic to capture files.

These capture files can then be examined again with TCPDump or with other analysis programs such as Wireshark.

Within JUMBO BLOCK Rescue, TCPDump thus represents the compact terminal-based complement to Wireshark.

## Area of use in the rescue system

TCPDump is particularly suitable for:

quick network diagnostics in the terminal,

capturing network traffic without a graphical interface,

targeted examination of individual hosts or ports,

DNS, ARP, ICMP, TCP and UDP diagnostics,

creating PCAP files,

long-term or rotating captures,

analysing existing capture files,

checking Nmap, MTR or Dig traffic,

and preparing a capture for later detailed analysis with Wireshark.

## Basic call

A basic call is:

```
tcpdump
```

TCPDump then tries to capture packets on a suitable network interface.

For controlled rescue use, however, the desired interface should be specified explicitly wherever possible.

### Displaying network interfaces

The following option lists the capture interfaces available to TCPDump:

```
tcpdump -D
```

With an unknown interface configuration, this should be one of the first steps.

Depending on the system, Ethernet, Wi-Fi, loopback, VPN and virtual interfaces, for example, may be present.

### Selecting the interface

The following option specifies the desired capture interface:

```
-i
```

Example:

```
sudo tcpdump -i INTERFACE
```

The capture is then carried out specifically on this interface.

The specific interface name must be taken from the actual system configuration.

### any

Under Linux, the following can be used to capture packets from several available interfaces:

```
-i any
```

This uses the special `any` capture interface.

Example:

```
sudo tcpdump -i any
```

This is useful for initial troubleshooting when it is not yet known over which interface the relevant communication runs.

However, the properties of a capture via `any` differ from those of a capture on a single physical interface.

### Capture permissions

Live capture of network packets requires appropriate system permissions.

Which rights are available to a normal user within a specific installation depends on the system configuration.

The manual therefore does not assume that every user can start a live capture without elevated privileges.

Where privileged access is required, examples are shown with `sudo`.

### Standard output

Without `-w`, TCPDump decodes the captured packets and outputs a textual summary in the terminal.

The exact display depends, among other things, on:

protocol,

selected options,

name resolution,

verbose level

and capture content.

### Limiting the number of packets

The following option ends the capture after the specified number of packets has been processed:

```
-c ANZAHL
```

Example:

```
sudo tcpdump -i INTERFACE -c 100
```

This is particularly useful for short diagnostic tasks.

The capture then does not run indefinitely.

### Disabling name resolution

The following option prevents addresses from being converted into names:

```
-n
```

Example:

```
sudo tcpdump -n -i INTERFACE
```

IP addresses, for example, are then displayed numerically.

This has several advantages:

The output remains unambiguous.

Additional name resolutions are avoided.

The analysis is not distorted or delayed by DNS activity.

For technical rescue diagnostics, `-n` is therefore often useful.

### Displaying ports numerically as well

The following option additionally suppresses the conversion of port numbers into service names:

`-nn`

Example:

```
sudo tcpdump -nn -i INTERFACE
```

In this way, a value such as the following remains visible as a port number instead of being displayed as a service name on the basis of a local services database:

`443`

This is often the clearest display for precise network diagnostics.

### Verbose output

TCPDump supports several levels of more detailed output.

`-v`

This option provides additional information.

`-vv`

This option increases the level of detail further.

`-vvv`

This option provides the maximum verbose level intended for this option group.

However, more output does not automatically mean a better diagnosis.

For extensive captures, overly detailed terminal output can quickly become confusing.

### Time stamps

TCPDump displays packets with time information.

The following option suppresses the output of the time stamp:

`-t`

Additional `-t` change the time display further.

Time stamps are often important for fault analyses and should therefore not be removed without reason.

### Absolute time

The time display can be selected so that absolute date and time information is output.

This is particularly helpful when TCPDump results are to be correlated with:

system logs,

firewall logs,

server logs

or other event logs.

### Filter expression

One of the most important features of TCPDump is the ability to specify a capture filter expression directly on the command line.

Basic principle:

```
tcpdump [Optionen] AUSDRUCK
```

The expression determines which packets are selected for output or capture.

### BPF filters

TCPDump uses the filter expression language of the packet capture infrastructure, often referred to as BPF or libpcap filters.

This filter language is closely related to the **capture filters** of Wireshark.

However, it is not identical to the Wireshark **display filters**.

This is an important distinction.

### Filtering a host

Example:

```
sudo tcpdump -nn host 192.0.2.10
```

This selects packets in which the address concerned is involved as a host according to the filter.

### Source host

The following expression filters specifically by source:

```
src host
```

Example:

```
sudo tcpdump -nn src host 192.0.2.10
```

### Destination host

The following expression restricts to a destination:

```
dst host
```

Example:

```
sudo tcpdump -nn dst host 192.0.2.10
```

### Filtering a network

The following expression specifies a network range:

```
net
```

Example:

```
sudo tcpdump -nn net 192.0.2.0/24
```

The address serves solely as a documentation example.

### Filtering a port

The following expression selects a specific port:

```
port
```

Example:

```
sudo tcpdump -nn port 53
```

The expression alone does not yet necessarily distinguish between TCP and UDP.

### Filtering TCP

The following expression selects TCP traffic:

```
tcp
```

Example:

```
sudo tcpdump -nn tcp
```

### Filtering UDP

The following expression selects UDP traffic:

```
udp
```

Example:

```
sudo tcpdump -nn udp
```

### TCP port

Filters can be combined:

```
sudo tcpdump -nn tcp port 443
```

This selects TCP traffic for port 443.

### UDP port

Example:

```
sudo tcpdump -nn udp port 53
```

This selects UDP traffic for port 53.

### DNS traffic

A simple DNS-related capture can be, for example:

```
sudo tcpdump -nn port 53
```

This can capture TCP and UDP packets that match the port filter.

For an exclusively UDP-related examination:

```
sudo tcpdump -nn udp port 53
```

### ARP

ARP traffic can be examined with:

```
sudo tcpdump -nn arp
```

This is particularly useful with local IPv4 network problems.

This makes it possible to check, for example, whether:

ARP requests are sent

and

ARP responses are received.

### ICMP

IPv4 ICMP traffic can be filtered with:

```
sudo tcpdump -nn icmp
```

This makes it possible to examine, for example, echo requests, echo replies and certain ICMP error messages.

### IPv6

TCPDump also supports IPv6 traffic.

A filter can be restricted to IPv6 with, for example:

```
ip6
```

ICMPv6 can be filtered specifically in the same way.

### Logical combinations

Filter conditions can be combined with logical operators.

The essential operators are:

`and`

`or`

`not`.

Example:

```
tcp and port 443
```

### AND

Example:

```
host 192.0.2.10 and tcp port 443
```

Both conditions must be met.

### OR

Example:

```
port 53 or port 123
```

This selects packets that match either of the two conditions.

### NOT

Example:

```
not port 22
```

This allows packets matching the filter concerned to be excluded.

### Brackets

With more complex expressions, brackets may be required.

As the shell can itself interpret brackets and other special characters, a complex filter expression should sensibly be put in quotation marks.

Example principle:

```
sudo tcpdump -nn 'host 192.0.2.10 and (tcp port 80 or tcp port 443)'
```

This also specifies unambiguously which logical grouping is intended.

### Capture filters reduce the amount of data

A filter is applied during capture.

With a capture saved with `-w`, packets that do not match the filter are not included in the corresponding file.

This reduces:

storage requirement,

write load

and the later amount of data.

But it also means:

**Packets that have not been captured cannot be retrieved later.**

With an unknown cause of fault, a filter should therefore not be chosen unnecessarily narrowly.

### Snaplen

The following option specifies the snapshot length:

```
-s
```

It determines how many bytes of a packet are captured.

With TCPDump 4.99.5, the following value means that the default maximum size of **262144 bytes** is used:

```
-s 0
```

### Truncated packets

A smaller snaplen reduces the amount of data and storage requirement.

However, it can result in only the beginning of a packet being captured.

Crucial information may then be missing for later protocol or payload analysis.

With sufficient storage space, deliberate truncation should therefore only be carried out if the use case justifies it.

### Writing a capture to a file

The following option writes the raw packets to a capture file instead of decoding them normally in the terminal:

```
-w DATEI
```

Example:

```
sudo tcpdump -i INTERFACE -w diagnose.pcap
```

This file can then be opened with TCPDump or Wireshark.

### `-w` saves packets, not terminal text

A capture file created with `-w` contains the captured packet data in capture format.

It is not an ordinary text file containing the TCPDump output otherwise visible.

This is crucial:

**Text output → for direct human viewing**

**Capture file → for renewed packet processing and detailed analysis**

### Reading a capture file

The following option reads an existing capture file:

```
-r DATEI
```

Example:

```
tcpdump -nn -r diagnose.pcap
```

No live network connection is required for this.

### Applying filters to saved captures

A filter expression can also be used when reading an existing capture file.

Example:

```
tcpdump -nn -r diagnose.pcap tcp port 443
```

This does not change the file.

TCPDump merely displays the packets that match the filter.

### Offline analysis

The following combination is particularly useful within a rescue system:

```
tcpdump -r
```

A capture can be:

created on another system,

copied to an analysis medium

and then examined completely offline.

This allows the analysis to be separated from the original production environment.

### Handover to Wireshark

A typical workflow is:

```
sudo tcpdump -i INTERFACE -w diagnose.pcap
```

The capture is then ended.

The file created is then opened with Wireshark.

TCPDump handles the compact capture.

Wireshark then handles the detailed graphical protocol analysis.

### Terminal and GUI complement each other

TCPDump is particularly suitable when:

no graphical interface is needed,

a quick capture is to be started,

work is carried out remotely,

only certain packets are of interest,

or a capture is to be created for later analysis.

Wireshark, by contrast, is particularly powerful with:

protocol decoding,

stream analysis,

statistics,

graphical navigation

and detailed examination of individual fields.

### ASCII output

The following option additionally displays the content of suitable packet data as ASCII:

```
-A
```

Example:

```
sudo tcpdump -A -nn tcp port 80
```

This can be helpful with unencrypted text-based protocols.

However, it must be borne in mind that network data can contain sensitive content.

### Hex and ASCII output

The following option outputs packet data in hexadecimal and additionally as ASCII:

```
-X
```

This is useful when both the actual byte values and displayable characters are to be examined.

### Hex output

TCPDump has further output options for hexadecimal or combined displays.

These are particularly useful for low-level protocol analyses.

For large captures, however, a complete hex output quickly becomes very extensive.

### Link-layer header

The following option displays the link-layer header in the output:

```
-e
```

With Ethernet, this can make additional MAC-related information visible, for example.

This is helpful for ARP, VLAN and layer 2 diagnostics.

### Packet content can be sensitive

Options such as the following can make payload visible directly in the terminal:

```
-A
```

and

`-X`

With unencrypted protocols, this can include, for example:

user names,

application data,

URLs,

headers,

internal information

or other confidential content.

Terminal output and saved logs must therefore be handled accordingly.

### Capture direction

TCPDump 4.99.5 supports the following options for selecting the capture direction, provided that the capture platform used supports this function:

`-Q in`

`-Q out`

and:

`-Q inout`

This makes it possible, for example, to look specifically at only incoming or only outgoing traffic.

Support depends on the platform.

### Incoming traffic

Example principle:

```
sudo tcpdump -Q in -i INTERFACE
```

This restricts – on a supported platform – to incoming packets.

### Outgoing traffic

Example:

```
sudo tcpdump -Q out -i INTERFACE
```

This allows outgoing traffic to be examined accordingly.

### Promiscuous mode

On suitable interfaces, TCPDump tries by default to carry out packet capture in promiscuous mode.

With the following option, the interface is expressly **not** put into promiscuous mode:

```
-p
```

### Promiscuous mode does not automatically see the entire LAN

As with Wireshark, the following applies:

Promiscuous mode does not mean that a computer automatically receives all unicast traffic of a switched network.

Which packets are visible depends on what the network infrastructure actually transmits to the capture interface.

To observe other unicast traffic, a correspondingly configured mirror/SPAN port or another authorised measuring point may be required.

### Capture buffer

TCPDump or libpcap uses a capture buffer.

The following option sets its size on supported platforms:

```
-B
```

With very high network traffic, the buffer size can influence whether the capture application processes packets quickly enough.

However, a larger buffer size is no substitute for sufficiently powerful hardware and a suitable capture architecture.

### Packet loss

When a live capture is ended, TCPDump outputs statistics.

These include information on:

packets received

and

packets dropped by the kernel or capture mechanism, where the platform provides this information.

Such drop counters are important for assessing a capture.

### An incomplete capture can be misleading

If packets are lost during capture, for example:

TCP sequences can appear incomplete,

responses can be missing,

protocol sequences can appear wrong

or retransmissions can become difficult to interpret.

In a critical analysis, not only the network traffic but also the quality of the capture must therefore be assessed.

### Rotation by file size

With the following option, TCPDump can automatically switch to a new file when writing a capture file once the current file reaches the configured size:

```
-C DATEIGRÖSSE
```

This is useful for long-term captures.

### Rotation by time

The following option creates a new capture file at time intervals:

```
-G SEKUNDEN
```

This allows regular capture segments to be created, for example.

### Limiting the number of files

In conjunction with rotation options, the following option limits the number of files created:

```
-W ANZAHL
```

Depending on the combination used, files can be reused cyclically or capture can be ended once the limit is reached.

Before a long-term capture, the exact rotation logic required should therefore be specified deliberately.

### Long-term captures

For longer captures, the following should be clarified in advance:

which traffic is needed,

how high the data rate is,

how much storage space is available,

how long the capture is to run,

whether files are rotated,

and whether older data may be overwritten.

Otherwise, an uncontrolled capture can fill a storage device completely.

### Immediate output

TCPDump buffers its output.

Options for controlling output buffering are available for certain diagnostic and further processing scenarios.

This can be relevant in particular for pipelines or live evaluation.

### Capture via pipe

TCPDump can pass on capture data via standard output if `-w -` is used.

This enables Unix pipelines and specialised further processing.

Such constructions should only be used if the data paths and target programs are known precisely.

### No packet injection

TCPDump is primarily a capture and analysis tool.

Its normal task is not to generate or inject arbitrary network packets.

This distinguishes it, for example, from Nmap, which actively generates network probes.

### TCPDump is mostly passive

A normal TCPDump capture does not itself generate the application communication to be examined.

It observes the traffic visible at the capture point.

However, indirect effects can arise, for example if name resolution is used for the output.

This is another reason why `-n` or `-nn` is often useful for controlled diagnostics.

### TCPDump and Nmap

Nmap and TCPDump complement each other directly.

A possible workflow is:

Start a TCPDump capture.

Carry out an Nmap scan.

Stop the TCPDump capture.

Then analyse the capture.

This makes it possible to trace at packet level:

which probes Nmap sent,

which responses came back

and why a port may have been assessed as open, closed or filtered.

### TCPDump and Wireshark

Both tools can process capture files.

The division of tasks within JUMBO BLOCK Rescue can be described in simplified terms as:

**TCPDump → capture and filter quickly**

**Wireshark → analyse graphically in detail**

TCPDump is not merely a “smaller version” of Wireshark.

Its strength lies in particular in direct integration into terminal, remote and automated workflows.

### TCPDump and ARP Scan

ARP Scan actively generates ARP requests to discover devices in the local network.

TCPDump can observe this ARP traffic.

For example:

```
sudo tcpdump -nn arp
```

This makes it possible to trace which ARP requests are sent and which responses are received.

### TCPDump and MTR

MTR generates diagnostic traffic for examining the network path.

TCPDump can capture this traffic in parallel.

This can provide additional information with unusual routing or firewall problems.

### TCPDump and Dig

Dig generates targeted DNS queries.

At the same time, TCPDump can observe the actual DNS traffic.

A typical filter is:

```
sudo tcpdump -nn port 53
```

This makes it possible to check:

whether the query is actually sent,

to which address it goes,

whether a response comes back

and which transport communication takes place.

## TCPDump and GUFW

GUFW/UFW can filter packets according to the firewall rules.

TCPDump observes network traffic at a specific capture point in the network stack.

When interpreting a capture, it must therefore be taken into account where the capture takes place and which firewall rules are active.

The absence of an expected packet in the capture does not prove, without further analysis, at which point in the communication path it was lost.

## Typical rescue workflow

A controlled use of TCPDump can look like this:

**1. Determine interfaces**

```
tcpdump -D
```

**2. Select the relevant interface**

**3. Disable name resolution**

For clear technical output:

```
-nn
```

**4. Filter the problem as specifically as possible**

For example by host, port or protocol.

**5. Limit the capture by time or number of packets**

For example, with:

```
-c
```

Alternatively, stop the capture deliberately by hand.

**6. Create a capture file for complex problems**

```
-w diagnose.pcap
```

**7. Reproduce the problem**

**8. Stop the capture**

**9. Analyse the file first with TCPDump or then with Wireshark**

**10. Preserve the original capture unchanged for examinations relevant to documentation or evidence**

### Example: DNS does not work

Capture:

```
sudo tcpdump -nn -i INTERFACE port 53
```

The faulty DNS query is then triggered.

It can now be checked:

Does a query go out?

To which IP address?

Via UDP or TCP?

Does a response come back?

If no response is visible, it must then be examined further whether the problem lies, for example, with the DNS server, routing or filtering.

### Example: TCP connection

For a specific service:

```
sudo tcpdump -nn -i INTERFACE tcp port PORT
```

The connection is then triggered.

It can now be examined whether:

a SYN is sent,

a SYN/ACK is received,

an RST is received

or no response is visible at all.

This allows a connection problem to be examined at basic TCP level.

### Example: ARP problem

```
sudo tcpdump -nn -i INTERFACE arp
```

It is then observed whether an ARP request for the required IPv4 address is sent and answered.

If ARP resolution in the local Ethernet is already missing, an IP-based connection to the local destination concerned cannot be established normally.

### Example: capture for Wireshark

```
sudo tcpdump -nn -i INTERFACE -w diagnose.pcap
```

Reproduce the problem.

End the capture.

Then:

```
wireshark diagnose.pcap
```

This opens the capture in Wireshark for a detailed graphical analysis.

### Example: limited number of packets only

```
sudo tcpdump -nn -i INTERFACE -c 500 -w diagnose.pcap
```

The capture is ended after 500 packets have been captured.

With reproducible short fault patterns, this can prevent unnecessarily large files from being created.

### Capture files as sensitive data

Like Wireshark PCAPs, TCPDump captures can contain sensitive information.

Depending on the traffic, this includes:

IP addresses,

MAC addresses,

host names,

DNS queries,

communication relationships,

unencrypted payload,

login credentials of insecure protocols

and other internal information.

Capture files must therefore be protected in accordance with their content.

### Data protection and authorisation

Network traffic may only be captured where the necessary technical and organisational authorisation exists.

A capture can contain communication of other users or systems.

This applies in particular to:

corporate networks,

KRITIS,

production networks,

administrative networks

and shared infrastructures.

### Encrypted traffic

TCPDump can capture encrypted network packets.

However, this does not automatically remove the encryption of the payload.

With TLS, for example, the encrypted payload remains encrypted without the required cryptographic information.

Nevertheless, metadata such as the following remains visible:

IP addresses,

ports,

packet sizes

and times.

### Limitations

TCPDump only sees the traffic available at the selected capture point.

Promiscuous mode does not automatically deliver all traffic of a switched network.

A capture filter discards non-matching packets already during capture.

A snaplen that is too small can truncate important packet data.

Packet loss during capture can distort the analysis.

Encrypted payload remains encrypted without suitable keys or session information.

TCPDump does not replace a complete graphical protocol analysis.

It is not a port scanner, not a malware scanner and not an intrusion prevention system.

### Safety notes

Only carry out live captures in authorised networks.

For technical diagnostics, use `-n` or `-nn` where possible if no name resolution is needed.

Do not set capture filters unnecessarily narrowly if the cause of the fault is still unknown.

Check storage space before long-term captures.

With rotation, specify exactly whether old capture files may be overwritten.

Take capture drop counters into account in critical analyses.

Treat PCAP files as potentially sensitive data.

Do not log or pass on unencrypted packet data uncontrolled.

In examinations relevant to evidence, preserve the original capture unchanged and carry out further processing on copies.

## Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, TCPDump forms the **terminal-based level of packet capture and network traffic analysis**.

The first three tools of the network area thus result in a clear technical gradation:

**A.17 Nmap**

Active network discovery, port analysis, service and system classification.

**A.18 Wireshark**

Extensive graphical packet and protocol analysis.

**A.19 TCPDump**

Fast terminal-based packet capture, filtering and creation of capture files.

A particularly important rescue workflow is:

**TCPDump captures → PCAP is saved → Wireshark analyses in detail.**

The following can likewise be used as a combined analysis workflow:

**Nmap generates diagnostic traffic → TCPDump captures it → Wireshark explains it at protocol level**

TCPDump is thus the compact capture tool of JUMBO BLOCK Rescue for situations in which a fast, controlled view of network traffic, available directly in the terminal, is required.

# A.20 ARP Scan

**Area:** Network
**Program:** arp-scan
**Baseline 20261001:** arp-scan 1.10.0
**Installed package:** `arp-scan 1.10.0-2+b1`
**Operation:** command line
**Task:** detection and basic identification of active IPv4 devices in the local layer 2 network using ARP

## Purpose

ARP Scan is a specialised tool for detecting IPv4 systems in the local network.

The program sends ARP requests specifically to the specified IPv4 targets and displays the ARP responses received.

This allows ARP Scan to detect devices in the directly reachable local layer 2 network.

Typical results contain:

IPv4 address,

MAC address

and – where an assignment is possible – manufacturer or vendor information for the MAC address.

ARP Scan is thus considerably more specialised than Nmap.

Its central question is:

**Which IPv4 devices in the local network respond to ARP?**

## Area of use in the rescue system

Within JUMBO BLOCK Rescue, ARP Scan is particularly suitable for:

quick inventory of a local IPv4 network,

detecting unknown devices,

determining IPv4/MAC assignments,

searching for devices whose IP address is unknown,

checking whether an expected device is reachable locally,

detecting systems despite blocked ICMP echo requests,

determining a possible device manufacturer on the basis of the MAC address,

and preparing a subsequent more detailed examination with Nmap, Wireshark or TCPDump.

## ARP

ARP stands for:

**Address Resolution Protocol**

In IPv4 Ethernet networks, ARP is used to determine the associated link-layer address – typically a MAC address with Ethernet – for a local IPv4 address.

A system can, for example, ask in essence:

**Who has this IPv4 address?**

The system addressed can respond with its MAC address.

ARP Scan automates such ARP requests for a defined set of target addresses.

## Basic principle

ARP Scan sends an ARP request for each target.

If the target responds, the response is evaluated and displayed.

This produces a list of the responding devices in the local network.

## ARP only works locally

One of the most important limitations of ARP Scan is:

**ARP is not routed.**

ARP Scan therefore cannot simply examine devices behind a router in the same way.

The tool is intended for the local network directly reachable via the layer 2 network concerned.

## Consequence for rescue use

If, for example, there is a router between JUMBO BLOCK Rescue and a target network, the ARP requests are not forwarded by the router into this remote network.

Other tools and methods are required for remote IP networks.

Nmap, for example, can carry out IP-based examinations across routing boundaries, provided that the target network is reachable and the examination is authorised.

## IPv4

ARP Scan is used to detect **IPv4 hosts**.

ARP itself uses IPv4 addresses for the assignment.

For the corresponding neighbour discovery, IPv6 does not use ARP but mechanisms of the Neighbor Discovery Protocol.

ARP Scan is therefore not a general IPv6 host scanner.

### Basic syntax

The basic syntax is:

```
arp-scan [Optionen] [Ziele]
```

Targets can be specified, among other things, as:

IPv4 address,

host name,

CIDR network,

address range

or network with netmask.

### Single target

Example:

```
sudo arp-scan 192.0.2.10
```

The address serves solely as a documentation example.

ARP Scan sends an ARP request for the specified target.

### Several targets

Several target addresses can be specified.

This allows a defined group of local systems to be examined.

### CIDR network

A network can be specified in CIDR notation.

Example:

```
sudo arp-scan 192.0.2.0/24
```

ARP Scan generates the corresponding target addresses from this.

With the CIDR target definition, the network and broadcast addresses are also included in the target list.

### Address range

ARP Scan also supports explicit IPv4 address ranges.

Example principle:

```
192.0.2.1-192.0.2.20
```

This allows a small part of a local network to be examined specifically.

### Network and netmask

Alternatively, ARP Scan supports the notation:

`Netzwerk:Netzmaske`

Example principle:

`192.0.2.0:255.255.255.0`

This also allows a set of targets to be defined.

### Determining the local network automatically

One of the most important options is:

`--localnet`

Short form:

`-l`

Example:

```
sudo arp-scan --localnet
```

ARP Scan generates the target list on the basis of the IPv4 address and netmask of the interface used.

### Meaning of `--localnet`

`--localnet` does not mean:

“Scan all networks that this computer can reach.”

It means:

**Generate the target addresses from the IPv4 configuration of the local interface used.**

For rescue use, this is an important distinction.

### Network and broadcast address

With `--localnet`, the addresses are generated from the interface address and netmask.

The list generated from this also contains the network and broadcast address.

ARP Scan thus works through the generated set of targets systematically.

### Selecting the interface

The following options explicitly specify the network interface to be used:

`--interface=INTERFACE`

or:

`-I INTERFACE`

Example:

```
sudo arp-scan --interface=INTERFACE --localnet
```

This is particularly important if a rescue system has several active network interfaces.

### Automatic interface selection

If no interface is specified, ARP Scan looks for the lowest-numbered configured and active interface, excluding loopback.

For reproducible technical examinations, it nevertheless makes sense to specify the desired interface explicitly if several interfaces are present.

### Interface without an IPv4 address

ARP Scan can basically also be used on an interface that does not have its own IPv4 address.

Although ARP uses IPv4 addresses within its messages, it is not itself an IP protocol transported by routers.

For `--localnet`, by contrast, an interface configuration is required from which the local set of targets can be generated.

### Permissions

ARP Scan uses raw socket functions.

Appropriate permissions are required for this.

Under Linux, version 1.10.0 supports POSIX.1e capabilities and requires `CAP_NET_RAW` for the corresponding functions.

Alternatively, it can be run with sufficient system privileges.

Which specific permissions configuration JUMBO BLOCK Rescue uses is not inferred from the mere package baseline.

Where privileged access is required, the examples therefore use:

```
sudo
```

### Standard output

For responding hosts, ARP Scan typically displays:

IPv4 address,

MAC address

and vendor information.

Example structure:

```
IPv4-Adresse MAC-Adresse Hersteller
```

The specific output depends on the vendor data available and the selected options.

### MAC address

The MAC address identifies the network interface at the local link-layer level.

It is particularly helpful for local device analysis, because an IPv4 address can change, while the MAC address of a physical interface often remains more stable.

Nevertheless, it must not be regarded as an unchangeable device identity.

### MAC addresses can be changed

MAC addresses can be:

changed by software,

randomised,

generated virtually,

provided by virtualisation

or deliberately imitated.

A MAC address alone therefore does not prove the physical identity of a device.

### Vendor detection

ARP Scan can try to assign a manufacturer or registered organisation on the basis of the MAC address.

Version 1.10.0 uses in particular the following for this:

`ieee-oui.txt`

and:

`mac-vendor.txt`.

### IEEE assignments

In version 1.10.0, the following file contains assignments from various IEEE MAC address registries:

`ieee-oui.txt`

These include:

MA-L or OUI,

MA-M,

MA-S or OUI36

and IAB.

### Additional assignments

`mac-vendor.txt`

This file contains additional MAC/vendor assignments and can also be used for local or user-defined entries.

### Interpreting the manufacturer information correctly

The vendor information is an **indication of the registered MAC address assignment**.

It is not necessarily the manufacturer of the complete device.

For example, a network interface or a built-in network chip can come from a different manufacturer from the overall system.

### Virtualisation

With virtual machines, the MAC address can be assigned to a virtualisation provider or an address range used for this purpose.

A vendor display can therefore, for example, indicate a virtual network interface without determining the actual manufacturer of the physical host.

### No vendor information

If a MAC address cannot be matched to any known assignment, this does not mean that the device is unknown or faulty.

Possible causes include:

locally administered MAC address,

randomised MAC address,

missing entry in the data used,

or a non-matching vendor assignment.

### Updating the OUI file

The arp-scan project includes the following utility, which can be used to update the IEEE-based vendor information:

`get-oui`

Such an update requires a corresponding data source or network connection.

For the frozen rescue baseline, it is not inferred from this that the vendor data is updated automatically.

### Minimalist output

The following options use reduced output:

`--quiet`

or:

`-q`

For responding hosts, only the following is then displayed:

IP address

and

MAC address.

In this mode, the vendor data is not needed for the normal output.

### Host lines only

The following options suppress header and footer information:

`--plain`

or:

`-x`

Only the information of the responding hosts is output.

This is particularly useful for later processing by scripts.

### User-defined output format

Version 1.10.0 introduced the following option:

`--format`

This allows the output format to be defined flexibly.

This makes structured further processing of the scan results easier.

### Resolving host names

The following options resolve the IP addresses of responding systems into host names:

`--resolve`

or:

`-d`

However, such name resolution generates or requires additional name resolution activity and can influence the output.

For a pure layer 2 inventory, numerical IPv4 and MAC addresses are often more unambiguous.

### Numerical target addresses only

The following options specify that targets must be IPv4 addresses and not host names:

`--numeric`

or:

`-N`

With large target lists, this can reduce the effort for name resolution at startup.

### Targets from a file

The following options read target addresses or target patterns from a file:

`--file=DATEI`

or:

`-f DATEI`

One target is specified per line.

The following allows standard input to be used:

`-`

## `--file` and `--localnet`

When the following option is used:

`--localnet`

Targets cannot then also be specified via `--file` or additionally via the command line.

`--localnet` itself generates the set of targets to be scanned from the interface configuration.

### Retries

By default, ARP Scan does not send requests indefinitely.

The following options specify the total number of attempts per host:

`--retry=ANZAHL`

or:

`-r ANZAHL`

The documented default value is:

**2 attempts per host.**

### Why retries?

ARP works locally and is normally very reliable.

Nevertheless, packets can be lost, for example because of:

network load,

packet loss,

overloaded devices

or other temporary conditions.

A retry reduces the probability of missing an active host merely because of a single lost packet.

### Fewer retries

Fewer retries shorten the scan.

At the same time, the risk of not detecting a device that is actually present increases in the event of packet loss.

### Timeout

The following options specify the initial timeout per host:

`--timeout=MILLISEKUNDEN`

or:

`-t`

The documented default value is:

**500 ms.**

### Backoff

The following options specify the backoff factor for subsequent passes:

`--backoff=FAKTOR`

or:

`-b`

The documented default value is:

**1,5.**

This allows later attempts to take longer response times into account.

### Bandwidth

The following options specify the desired outgoing scan bandwidth:

`--bandwidth=WERT`

or:

`-B`

The documented default value is:

**256.000 Bit/s.**

### Why limit bandwidth?

ARP Scan actively generates ARP requests.

With large target ranges, this can result in a considerable number of broadcast requests.

A controlled send rate prevents unnecessarily high load.

### Higher bandwidth

A higher bandwidth can speed up a scan.

However, values that are too high can cause a high broadcast packet rate.

In sensitive production, industrial or KRITIS networks, the scan rate should therefore not be increased without a technical reason.

### Packet interval

Alternatively, the following options specify a minimum interval between outgoing packets:

`--interval`

or:

`-i`

This allows the send rate to be influenced directly.

`--interval` and `--bandwidth` are not used at the same time.

### Duplicates

A host can respond more than once.

ARP Scan can detect such duplicate responses and display them accordingly.

The following options suppress the display of duplicate packets:

`--ignoredups`

or:

`-g`

### Duplicate responses can be relevant

Duplicates are not inherently meaningless.

If, for example, several responses to one target address occur with different MAC addresses, this can indicate a special network configuration.

Such an observation should be examined and not hastily interpreted as an attack.

Possible causes can include legitimate high-availability, virtualisation or network configurations.

### ARP and firewalls

ARP works below the normal IP port level.

A system that filters ICMP echo or TCP connection attempts can still respond to ARP.

This makes ARP Scan particularly useful for local host detection.

### A device that does not respond to ping can be visible

A device can, for example:

block ICMP echo requests

and

still respond to ARP requests.

Therefore:

“Ping does not respond”

This is not the same as:

“Device is not present in the local network.”

### An ARP finding does not prove a service

An ARP response shows that a system at the local network level is responding to the IPv4 address concerned.

However, it says nothing about:

which TCP ports are open,

which UDP services are present,

which operating system is running,

or whether a particular application service is working.

Other tools are required for this.

### ARP Scan and Nmap

Nmap also has ARP-based host discovery functions for local Ethernet targets.

However, ARP Scan is specifically geared to this task.

A sensible division of work is:

**ARP Scan → detect local IPv4 devices quickly**

**Nmap → then examine systems found specifically for ports, services and other network characteristics**

### ARP Scan and Wireshark

Wireshark can observe the ARP packets generated by ARP Scan.

The following display filter shows the corresponding ARP traffic:

```
arp
```

This makes it possible to examine directly:

which ARP requests were sent,

which devices responded

and which address information is contained in the packets.

### ARP Scan and TCPDump

TCPDump can observe the same process in the terminal.

Example:

```
sudo tcpdump -nn arp
```

ARP Scan can be run in parallel.

This makes it possible to trace sent requests and received replies at packet level.

### ARP Scan and MTR

MTR examines IP-based network paths across several routers or hops.

ARP Scan, by contrast, works exclusively in the local layer 2 network.

The two tools thus lie at different analysis levels:

**ARP Scan → local neighbour**

**MTR → routed network path**

### Typical rescue workflow

A controlled deployment can be carried out as follows:

**1. Establish the physical or logical network connection**

**2. Determine the desired interface**

If there are several interfaces, select the interface that is actually relevant.

**3. Scan the local network**

Example:

```
sudo arp-scan --interface=INTERFACE --localnet
```

**4. Document the results**

In particular:

IPv4 address,

MAC address,

vendor information.

**5. Compare expected and unexpected systems**

**6. Examine interesting systems further specifically**

For example with Nmap.

**7. Observe packet traffic if required**

With Wireshark or TCPDump.

### Example: IP address of a device unknown

A device is physically located in the same local Ethernet network, but its IPv4 address is unknown.

ARP Scan can examine the local network:

```
sudo arp-scan --localnet
```

The responding devices are output with IPv4 and MAC address.

The system being sought can possibly be narrowed down on the basis of:

a known MAC address,

vendor information

or comparison with other devices.

### Example: device does not respond to ping

An expected device does not respond to ICMP echo requests.

This does not yet prove that it is offline.

A local ARP scan can check whether the system still responds to ARP.

If it responds to ARP, there is at least local layer 2 communication.

The cause of the missing ping must then be sought at a different level.

### Example: unknown device

An ARP scan shows an unexpected IPv4/MAC combination.

The further procedure should be controlled:

Document the MAC/vendor information.

Compare the network documentation.

If authorised, then examine the system specifically with Nmap.

If necessary, observe ARP and IP communication with Wireshark or TCPDump.

An unknown ARP entry alone does not prove a security incident.

### Example: wrong interface

If JUMBO BLOCK Rescue has several interfaces, a scan via the wrong interface can deliver a completely different or empty result.

If in doubt, the interface should therefore be specified explicitly:

```
sudo arp-scan -I INTERFACE --localnet
```

This specifies unambiguously which local network is examined.

### Limitations

ARP Scan works with ARP and thus with IPv4-related local neighbour discovery.

ARP is not routed.

Remote networks behind routers therefore cannot be examined directly in the same way.

ARP Scan does not carry out a TCP/UDP port scan.

It does not detect network services.

It does not carry out operating system detection like Nmap.

Vendor information is not a reliable device identification.

MAC addresses can be changed, randomised or virtualised.

A device that does not respond is not switched off with absolute certainty; packet loss, network segmentation or other technical causes can prevent a response.

## Safety notes

Only examine authorised local networks.

Before the scan, check the correct interface and the intended network range.

Do not increase the scan rate unnecessarily in production and KRITIS networks.

Only use vendor information as an indication.

Do not treat MAC addresses as a cryptographically reliable device identity.

Do not classify unknown devices as compromised solely on the basis of an ARP result.

In the case of unusual or contradictory results, verify at packet level with Wireshark or TCPDump.

## Technical classification in JUMBO BLOCK Rescue

Within the network area of JUMBO BLOCK Rescue, ARP Scan performs a clearly defined task:

**fast active detection of local IPv4 devices at layer 2.**

The network tools so far thus form four different analysis perspectives:

**A.17 Nmap**

Active host, port, service and system analysis.

**A.18 Wireshark**

Graphical packet and protocol analysis.

**A.19 TCPDump**

Terminal-based packet capture and filtering.

**A.20 ARP Scan**

Specialised local IPv4 device detection via ARP.

A particularly sensible rescue workflow is:

**ARP Scan → find a local device**

**Nmap → examine ports and services of the device found**

**TCPDump or Wireshark → analyse actual communication at packet level if required**

ARP Scan thus answers very quickly one of the most fundamental questions of a local network diagnosis:

**Which IPv4 devices are actually visible via ARP on my directly connected local network?**

# A.21 MTR

**Area:** Network
**Program:** MTR
**Baseline 20261001:** MTR 0.95
**Installed packages:** `mtr 0.95-1.1+b1`, `mtr-tiny 0.95-1.1+b1`
**Operation:** command line / interactive terminal interface
**Task:** continuous analysis of the network path with hop, latency and packet loss statistics

## Purpose

MTR is a network diagnostic tool that combines essential functions of the following two tools in a single continuous measurement:

**traceroute**

and:

**ping**

The program examines the network path between the local system and a target and determines the routers or hops in between.

Unlike a one-off conventional traceroute, MTR then carries out further measurements continuously.

This produces statistical values for each visible hop on:

response behaviour,

packet loss,

round-trip time,

minimum and maximum values,

average latency

and fluctuations.

## Area of use in the rescue system

Within JUMBO BLOCK Rescue, MTR is particularly suitable for examining:

slow network connections,

sporadic connection problems,

packet loss,

latency problems,

unstable WAN connections,

routing paths,

internet connections,

VPN paths,

connections to remote servers,

differences between IPv4 and IPv6,

and problems that only occur after several network hops.

## Basic principle

MTR sends network probes with deliberately limited TTL values.

With IPv4, TTL stands for:

**Time To Live**

At each router an IP packet passes, the TTL value is reduced.

If the value reaches zero, the packet is not forwarded.

The router concerned can respond to this with an ICMP message.

MTR uses this mechanism to determine the individual hops along a network path.

## Step-by-step TTL

In simplified terms, the examination begins with:

TTL 1

then:

TTL 2

then:

TTL 3

and so on.

This makes the routers along the path visible one after another, provided that they deliver corresponding responses.

## Difference from traceroute

A conventional traceroute examines the path at a specific point in time.

MTR repeats its measurements continuously.

This can make problems visible that only occur intermittently.

For example:

sporadic packet loss,

fluctuating latency,

temporary congestion,

or intermittent reachability problems.

### Difference from ping

Ping primarily examines the reachability of and round-trip time to a target.

MTR, by contrast, simultaneously examines the visible intermediate stations of the path.

In simplified terms:

**Ping → How does the target behave?**

**Traceroute → Via which hops does the path run?**

**MTR → How do the target and visible hops behave over time?**

### Basic call

A basic call is:

```
mtr ZIEL
```

For example:

```
mtr example.net
```

MTR then normally starts its interactive terminal display.

### Interactive display

In the interactive view, MTR updates the measured values continuously.

This allows changes to be observed immediately.

Each hop is displayed as a separate line.

Typical columns are:

```
Loss%
Snt
Last
Avg
Best
Wrst
StDev
```

### Loss%

`Loss%` shows the proportion of probes directed at this hop for which no corresponding response was registered.

This value must be interpreted very carefully.

A high value at an intermediate hop does not automatically prove that the router is discarding a corresponding number of transit packets.

### Snt

`Snt` shows the number of probes sent so far.

A statistic with only a few probes is naturally less meaningful than a longer series of measurements.

### Last

`Last` designates the round-trip time of the most recently answered probe.

The unit is typically milliseconds.

### Avg

`Avg` is the average measured round-trip time.

It helps to estimate the typical latency of the hop concerned over the measurement period.

### Best

`Best` is the lowest measured round-trip time.

This value can indicate which latency was achievable under favourable conditions.

### Wrst

`Wrst` designates the highest measured round-trip time.

A very large difference between `Best` and `Wrst` indicates strong fluctuations in the response times.

### StDev

`StDev` designates the standard deviation of the measured round-trip times.

It describes the spread of the measured values.

A larger value means stronger fluctuations.

### Latency

The time displayed by MTR is a **round-trip time**.

It describes the outward and return path of a probe.

It is therefore not directly identical to the pure one-way transit time.

### Jitter and fluctuations

Strongly fluctuating round-trip times can be problematic for certain applications.

These include, for example:

VoIP,

video conferences,

interactive remote sessions

and other time-critical communication.

In addition to the conventional latency values, MTR 0.95 can also display various jitter-related statistics fields.

### The most important interpretation error

A typical MTR result can look like this, for example:

Hop 1: 0 % loss
Hop 2: 0 % loss
Hop 3: 60 % loss
Hop 4: 0 % loss
Hop 5: 0 % loss
Target: 0 % loss

From this, it must **not** be concluded:

“Hop 3 loses 60 % of the network traffic.”

### Why?

Routers have to forward transit traffic and, at the same time, respond to diagnostic packets themselves where applicable.

A router can do the following with its own ICMP responses:

limit them,

give them lower priority

or suppress them completely,

while it continues to forward normal transit traffic correctly.

The official MTR project expressly points out this behaviour of modern routers.

### Decisive interpretation principle

If an intermediate hop shows high apparent loss but later hops and the target do not show corresponding loss, this does **not indicate a proven transit loss at this hop**.

The router may then merely not be responding reliably to the diagnostic probes directed at itself.

### Continuing loss

A pattern such as the following is different:

Hop 1: 0 %
Hop 2: 0 %
Hop 3: 20 %
Hop 4: 20 %
Hop 5: 20 %
Target: 20 %

Here, a loss pattern begins at hop 3 and continues to the target.

This is considerably more relevant diagnostically.

Nevertheless, this pattern must also be checked in the overall context.

### The target is particularly important

The target hop is of central importance for assessing an end-to-end connection.

If the target shows no packet loss, losses at individual intermediate hops must not automatically be interpreted as end-to-end packet loss.

### Hops that do not respond

A router does not have to respond to the diagnostic probes used by MTR.

A hop can therefore appear as follows or in a comparable form:

???

This does not automatically mean that the path is interrupted there.

If later hops respond, the packets are evidently being transported onwards via or beyond this point.

### Firewalls and filters

Firewalls and routers can treat diagnostic traffic differently.

Possible effects are:

ICMP is discarded,

ICMP is limited,

UDP is treated differently from ICMP,

TCP probes are treated differently,

or responses are only partially generated.

MTR results must therefore always be interpreted taking the probe protocol used into account.

### Name resolution

By default, MTR can try to resolve IP addresses into host names.

This improves readability but can:

generate additional DNS communication,

delay the display,

or be disruptive if DNS is faulty.

### Disabling DNS

The following options disable name resolution:

```
-n
```

or:

```
--no-dns
```

Example:

```
mtr -n ZIEL
```

Numerical IP addresses are then displayed.

For a technical rescue diagnosis, this is often particularly useful.

### Host name and IP address together

With the following options, MTR can display both host names and numerical IP addresses:

```
-b
```

or:

```
--show-ips
```

This can be helpful when documenting a network path.

### Forcing IPv4

The following option uses IPv4 exclusively:

```
-4
```

Example:

```
mtr -4 ZIEL
```

### Forcing IPv6

The following option uses IPv6 exclusively:

```
-6
```

Example:

```
mtr -6 ZIEL
```

This allows IPv4 and IPv6 paths to be examined separately.

### Why test IPv4 and IPv6 separately?

A host name can have both IPv4 and IPv6 addresses.

The two protocol families can use:

different routers,

different provider paths,

different firewalls

and have different fault patterns.

A working IPv4 connection therefore does not automatically prove that the IPv6 path also works – and vice versa.

### Report mode

Report mode is particularly important for documentation and reproducible diagnosis.

It is enabled with:

```
-r
```

or:

```
--report
```

Example:

```
mtr -r ZIEL
```

MTR carries out a defined number of measurement cycles, then outputs a summary report and exits.

### Number of measurement cycles

The following options specify the number of measurement cycles:

```
-c ANZAHL
```

or:

```
--report-cycles ANZAHL
```

Example:

```
mtr -r -c 100 ZIEL
```

This produces a report after 100 measurement cycles.

### Saving the report to a file

The text output can be written to a file via the shell.

Example:

```
mtr -n -r -c 100 ZIEL > mtr-report.txt
```

This produces a measurement report that can be documented.

### Why several measurement cycles?

A single measured value only shows a brief moment.

A larger number of measurements can make the following more visible:

sporadic loss,

latency peaks,

fluctuations over time

and unstable connections.

However, the measurement duration should match the actual diagnostic purpose.

### Wide report

The following options use wide report mode:

```
-w
```

or:

```
--report-wide
```

Host names are not truncated in the report in the same way.

This is particularly useful with long DNS names.

### Example for rescue documentation

A practical call is:

```
mtr -n -r -c 100 ZIEL
```

This:

avoids DNS resolution,

carries out 100 measurement cycles,

and then outputs the statistics.

### Interval

The following options set the interval between ICMP echo requests:

```
-i SEKUNDEN
```

or:

```
--interval SEKUNDEN
```

The documented default value is:

**1 second.**

### Measurement load

MTR actively generates network traffic.

The documentation expressly points out that each running MTR instance generates a relevant amount of network traffic and that measurements can influence network performance.

In sensitive environments, the measurement duration and interval should therefore be chosen deliberately.

### Packet size

The following options specify the size of the probe packets:

```
-s PACKETSIZE
```

or:

```
--psize PACKETSIZE
```

With ICMP, the value includes the IP and ICMP headers.

Different packet sizes can deliver different results with certain problems.

### Why packet size can be relevant

Problems can depend, for example, on:

MTU,

tunnel connections,

VPNs,

fragmentation

or certain network paths.

However, the packet size should be changed selectively and not regarded as a general standard diagnosis.

### Selecting the interface

The following options select a specific network interface for the probes:

```
-I NAME
```

or:

```
--interface NAME
```

This is particularly useful if JUMBO BLOCK Rescue has several possible network paths, for example:

Ethernet

and

Wi-Fi.

### Specifying the source address

The following options bind the outgoing connection to a specific source address:

```
-a ADDRESS
```

or:

```
--address ADDRESS
```

This is helpful with systems with several IP addresses or more complex routing configurations.

### Customising report fields

The following options specify the displayed statistics fields and their order:

```
-o FIELDS
```

or:

```
--order FIELDS
```

MTR supports, among others, fields for:

loss ratio,

packets sent,

packets received,

current RTT,

best RTT,

average RTT,

worst RTT,

standard deviation

and various jitter values.

### JSON

For JSON output, MTR 0.95 supports the following options, provided that the library required for this was available at build time:

```
-j
```

or:

```
--json
```

JSON is suitable for automated further processing.

However, the baseline package list alone does not prove with which optional build features the specific binary was created.

Availability should therefore be checked on the running rescue system before it is assumed for an automated workflow.

### XML

The following options produce XML output:

```
-x
```

or:

```
--xml
```

This is also intended for machine processing.

### CSV

MTR also supports CSV output.

This allows measurement data to be transferred, for example, into spreadsheet and analysis tools.

For later automated processing, the field structure actually produced by the installed version should be taken into account.

### ICMP

Conventional MTR operation uses ICMP echo probes.

However, this is not the only supported method.

### UDP

MTR also supports UDP probes.

This can be useful if ICMP is treated differently from UDP in the network.

### TCP

MTR also supports TCP probes.

This can be particularly helpful when diagnosing a specific TCP-based service path.

### Destination port

The following options specify a destination port for TCP, UDP or supported transport-protocol-based measurements:

```
-P PORT
```

or:

```
--port PORT
```

This allows the examination to be adapted more closely to the service actually of interest.

### Why different probe protocols are important

Networks do not necessarily treat the following protocols in the same way:

ICMP,

UDP

and

TCP

Firewalls, ACLs, routers and providers can have different rules for these protocols.

An ICMP-based MTR test therefore does not necessarily reflect the behaviour of a TCP application exactly.

### Example

If an HTTPS connection is causing problems, a TCP-based path examination to the relevant port can provide additional information.

The result must nevertheless be interpreted carefully, because MTR is a diagnostic measurement and not a complete simulation of the application.

### MPLS

With the following options, MTR can display information from ICMP extensions for MPLS, provided that corresponding information is present in the response packets:

`-e`

or:

`--mpls`

This can provide additional path information in provider or larger corporate networks.

### Autonomous system information

MTR can also display information about autonomous systems for hops.

The following options display the AS number:

`-z`

or:

`--aslookup`

Such information requires corresponding external assignment data or queries.

It is therefore not to be equated with a path analysis that works completely offline.

### Interactive control

Keyboard commands are available during interactive MTR execution.

These include:

p – pause measurement

`Space` – resume

n – toggle DNS resolution

r – reset statistics

j – switch between latency and jitter statistics

q – quit

Other settings can also be adjusted at run time.

### Resetting statistics

Resetting the statistics is helpful if a connection is to be prepared first and only a specific period is then to be considered.

After the reset, the statistical values start again.

### The route can change

Internet and corporate routes are not necessarily static.

Hops can change between two measurements.

Different paths can also occur during an ongoing examination.

Possible causes are, for example:

routing changes,

load balancing,

ECMP,

provider changes

or failover.

### Load balancing

Several network paths can result in diagnostic packets passing different routers.

MTR results can therefore look more complex than a simple linear path.

A changing hop therefore does not automatically mean a malfunction.

### Asymmetric routing

MTR measures round-trip times.

The outward and return paths do not have to be identical.

A high RTT value at a hop therefore does not readily reveal whether the delay arose on the outward path, the return path or in the router's handling of the response.

### ICMP prioritisation

A router can forward transit packets with high priority while throttling responses from its own control plane.

Precisely for this reason, intermediate hop loss must be assessed carefully.

### Example: apparent loss at an intermediate hop

Measurement:

Hop 1: 0 %
Hop 2: 40 %
Hop 3: 0 %
Hop 4: 0 %
Target: 0 %

Interpretation:

The 40 % value at hop 2 does not prove 40 % loss of the actual end-to-end traffic.

If hop 2 actually discarded 40 % of all transit packets, corresponding effects would normally also have to be visible at the subsequent hops.

A plausible explanation is therefore a limited readiness of the router to respond to the diagnostic probes.

### Example: loss continues

Hop 1: 0 %
Hop 2: 0 %
Hop 3: 15 %
Hop 4: 15 %
Hop 5: 15 %
Target: 15 %

This pattern is considerably more suspicious.

From hop 3 onwards, a comparable loss is visible up to the target.

Hop 3, or the connection in this area, thus becomes an important point of examination.

Nevertheless, this is a diagnostic indication and not automatically final proof of the exact physical location of the fault.

### Example: latency jump

Hop 1: 1 ms
Hop 2: 2 ms
Hop 3: 3 ms
Hop 4: 35 ms
Hop 5: 36 ms
Target: 36 ms

A permanent increase in latency from hop 4 onwards can indicate a longer network section or a different path characteristic.

Such a jump is not automatically a fault.

With WAN and internet paths, for example, a large geographical distance can be added.

## Geographical distance

Round-trip time is physically limited, among other things, by the length of the transmission path.

A connection over large geographical distances therefore inevitably has a higher minimum latency than a local connection.

MTR can make this latency visible, but cannot conclude a defect from it alone.

## Example: strong fluctuations

Best: 10 ms
Avg: 25 ms
Wrst: 180 ms
StDev: high

This shows a strongly fluctuating response time.

Possible causes must then be examined further.

MTR alone does not automatically identify the cause.

## Typical rescue workflow

A controlled use of MTR can be carried out as follows:

**1. Specify the target**

Use the target that is actually problematic.

**2. Take the protocol family into account**

If necessary, test IPv4 and IPv6 separately.

**3. Disable DNS if appropriate**

Example:

```
mtr -n ZIEL
```

**4. Let the measurement run long enough**

Do not draw conclusions immediately from individual early measured values.

**5. Assess the target hop first**

Is there packet loss or unusual latency there?

**6. Then compare intermediate hops**

Where does a pattern begin that continues to the target?

**7. Do not over-interpret individual isolated loss values**

**8. Create a report if required**

Example:

```
mtr -n -r -c 100 ZIEL
```

**9. Test an alternative probe method if a protocol is suspected**

ICMP, UDP or TCP according to the diagnostic case.

**10. Supplement with TCPDump or Wireshark if required**

This allows the actual probe and response traffic to be examined at packet level.

## MTR and Nmap

Nmap examines:

hosts,

ports,

services

and other system properties.

MTR examines:

the network path

and its continuous response behaviour.

A server can be reachable with Nmap while MTR at the same time reveals unusual latency or packet loss along the path.

## MTR and Wireshark

Wireshark can capture the network packets generated by MTR.

This makes it possible to examine precisely:

which probes are sent,

which ICMP responses come back,

and which network packets are actually visible at the local capture point.

## MTR and TCPDump

TCPDump can capture MTR traffic in the terminal.

This is helpful if:

MTR delivers unusual results,

response packets are to be examined more closely,

or it must be documented which packets were actually transmitted.

## MTR and ARP Scan

ARP Scan examines the directly local IPv4 layer 2 network.

MTR, by contrast, examines a routed IP path across several hops.

The two tools thus complement each other:

**ARP Scan → Who is located locally on the network?**

**MTR → Via which IP path do I reach a remote target, and how does this path behave?**

## MTR and Dig

Dig examines DNS specifically.

MTR examines the network path.

If, for example, a DNS server can only be reached slowly:

Dig can examine the DNS response itself

and:

MTR can analyse the network path to the DNS server.

## Limitations

MTR only shows routers that respond accordingly to the diagnostic probes used.

Intermediate hops that do not respond do not automatically mean an interrupted path.

Packet loss at an intermediate hop is not automatically transit packet loss.

Round-trip time does not show the latency of the outward and return paths separately.

Routing can change during or between measurements.

Load balancing can produce different paths.

ICMP, UDP and TCP probes can be treated differently by network components.

MTR does not automatically identify the physical cause of a problem.

An error-free MTR measurement does not prove that every application and every protocol works without errors.

## Safety notes

Only use MTR against systems and networks whose diagnosis is permitted.

Measurements actively generate network traffic.

Choose the measurement duration and probe interval deliberately in sensitive networks.

Never interpret intermediate hop loss in isolation as proof of a network fault.

Include the target hop and subsequent hops in the assessment.

With dual-stack problems, examine IPv4 and IPv6 separately if appropriate.

With service-related problems, bear in mind that ICMP can be treated differently from TCP or UDP.

Reports can contain internal IP addresses, host names and routing information and must be protected accordingly.

## Technical classification in JUMBO BLOCK Rescue

Within the network area of JUMBO BLOCK Rescue, MTR takes on the role of **continuous path, latency and packet loss analysis**.

This results so far in:

**A.17 Nmap**

Active host, port and service analysis.

**A.18 Wireshark**

Graphical packet and protocol analysis.

**A.19 TCPDump**

Terminal-based packet capture and filtering.

**A.20 ARP Scan**

Local IPv4 device detection at layer 2.

**A.21 MTR**

Continuous analysis of the routed network path.

A particularly important diagnostic workflow is:

**ARP Scan checks the local environment.**

**MTR follows the path to the remote target.**

**Nmap examines the target and its services.**

**TCPDump or Wireshark examine the packets actually transmitted if required.**

MTR thus closes the diagnostic gap between a simple local reachability check and a detailed packet capture.

Its greatest strength lies not only in displaying a network path but in making its behaviour visible **across a series of measurements**.

The central rule for correct use is:

**A conspicuous intermediate hop only becomes particularly meaningful if the conspicuous behaviour continues in the subsequent hops up to the target.**

# A.22 Dig

**Area:** Network
**Program:** Dig
**Technical basis:** BIND 9 DNS Utilities
**Baseline 20261001:** BIND 9.20.29
**Installed package:** `bind9-dnsutils 1:9.20.29-1~deb13u1`
**Operation:** Command line
**Purpose:** Direct DNS queries, resolver diagnostics, analysis of resource records and DNS delegations

## Purpose

Dig stands for:

**Domain Information Groper**

and is a flexible command-line tool for querying DNS name servers.

Dig sends targeted DNS queries and displays detailed DNS responses returned by the queried name server.

This makes it possible to examine, independently of browsers or other applications:

which DNS server responds,

which resource records are present,

which IPv4 or IPv6 addresses are returned,

which name servers are authoritative for a domain,

which mail servers are published,

which TTL values apply,

which DNS status code is returned,

whether one resolver responds differently from another,

and how a DNS delegation proceeds from the root zone to the target.

## Use in the rescue system

Within JUMBO BLOCK Rescue, Dig is particularly suitable for:

DNS troubleshooting,

checking the configured DNS resolvers,

querying specific DNS servers directly,

comparing multiple resolvers,

A and AAAA queries,

MX, NS, TXT, CNAME and SOA queries,

reverse DNS,

analysing DNS delegations,

examining DNSSEC-related data,

comparing DNS over UDP and TCP,

and distinguishing a DNS problem from a general network problem.

### DNS is a diagnostic layer of its own

A common diagnostic mistake is to equate failed name resolution directly with a lack of network connectivity.

Example:

`server.example` cannot be resolved.

This does not automatically mean:

“The network is not working.”

Possible causes include:

DNS server unreachable,

incorrect DNS server configured,

name does not exist,

record missing,

delegation error,

resolver returns an error,

or DNS traffic is being filtered.

Dig makes it possible to examine this layer directly.

### Basic syntax

The typical syntax is:

```
dig [@server] name [type]
```

Example:

```
dig example.com
```

If no record type is specified, Dig queries an:

A

record by default.

### Default resolver

If no specific DNS server is specified with `@server`, Dig uses the name servers listed in:

`/etc/resolv.conf`

.

This initially allows you to examine exactly the DNS path used by the rescue system according to its resolver configuration.

### No usable resolver configured

If Dig finds no usable server address there, the BIND documentation states that it attempts to send the query to the local system.

For diagnostic purposes, it is therefore important not simply to assume which DNS server is being used, but to check the output and system configuration directly.

### Querying a specific DNS server

A DNS server can be specified explicitly:

```
dig @SERVER example.com
```

SERVER can be an IPv4 or IPv6 address, or a resolvable host name.

Example:

```
dig @192.0.2.53 example.com
```

The IP address used is for documentation purposes only.

### Why `@server` matters

This allows different resolvers to be compared directly.

For example:

local DNS server,

corporate resolver,

VPN DNS server,

authoritative name server,

or another explicitly designated resolver.

If two servers return different results, this is an important diagnostic clue.

### Distinguishing direct server queries from the system resolver

A:

`dig example.com`

and a:

`dig @SERVER example.com`

must be considered separately for diagnostic purposes.

The first command uses the system's resolver configuration.

The second sends the DNS query explicitly to the specified server.

This can help determine whether the problem lies with the local resolver selection.

### Structure of the standard output

Standard Dig output contains several sections.

These typically include:

header,

QUESTION SECTION,

ANSWER SECTION,

AUTHORITY SECTION,

ADDITIONAL SECTION,

and final query information.

Not every response contains all sections.

### Header

The header contains key information about the DNS response.

This includes in particular:

opcode,

status,

flags,

query ID,

and the number of records in the various sections.

### Status

Of particular importance is:

`status:`

Successful DNS transport does not automatically mean that the requested name exists.

For example, Dig may receive a valid DNS response whose status is:

`NXDOMAIN`

.

## NOERROR

`NOERROR`

means that the DNS query was processed without a DNS protocol error.

However, this does not necessarily mean that the Answer Section contains a result for the requested record type.

## NXDOMAIN

`NXDOMAIN`

means that the requested DNS name does not exist.

This is a valid DNS response.

One very important technical detail in the BIND documentation is therefore:

Dig exits with **exit code 0 even when an NXDOMAIN response is received**.

Scripts must therefore not equate:

**Exit code 0**

with:

**Name exists**

.

## No response from the server

If Dig receives no response from the DNS server, this is fundamentally different from NXDOMAIN.

Possible causes then include problems with:

reachability,

firewall,

routing,

DNS service,

transport,

or server response.

## QUESTION SECTION

The Question Section shows what was actually queried.

It typically contains:

name,

DNS class,

and record type.

The usual DNS class for Internet DNS is:

`IN`.

## ANSWER SECTION

The Answer Section contains the actual resource records returned in response to the query.

An entry typically contains:

name,

TTL,

class,

record type,

and record data.

## TTL

TTL stands for:

**Time To Live**

and indicates how long a DNS record may be cached in accordance with DNS rules.

Dig displays TTL values numerically by default.

With:

`+ttlunits`

TTL values can be displayed in more readable time units such as seconds, minutes, hours, days or weeks.

## AUTHORITY SECTION

The Authority Section may contain information about the name servers responsible for the relevant DNS namespace or other authority information.

Its contents depend on the specific response.

## ADDITIONAL SECTION

The Additional Section may contain additional resource records that are useful for the response.

Modern DNS responses may also contain EDNS-related information here or elsewhere in the output.

### Query Time

Dig normally displays the measured query time.

This time represents the DNS query from the perspective of the Dig process.

It must not automatically be interpreted as pure network latency.

With a recursive resolver, for example:

cache state,

recursive resolution,

DNSSEC processing,

and server load

can also affect response time.

## SERVER

At the end of the standard output, Dig shows which DNS server returned the response.

This is particularly important during rescue diagnostics.

It allows you to verify that the expected resolver actually responded.

## MSG SIZE

Dig also displays the size of the DNS message received.

This can be relevant, for example, when investigating large DNS responses, EDNS or transport problems.

### A record

An:

A

record maps a name to an IPv4 address.

Example:

```
dig example.com A
```

Since A is the default type, the following is also sufficient:

```
dig example.com
```

### AAAA record

An:

`AAAA`

record contains an IPv6 address.

Example:

```
dig example.com AAAA
```

This makes it possible to check independently whether IPv6 data is published for a name.

### Examining A and AAAA separately

On dual-stack systems, both record types should be checked separately where appropriate.

Successful A resolution does not prove that an AAAA record is present and correct – and vice versa.

This complements the separate IPv4/IPv6 path analysis provided by MTR.

### CNAME

A:

`CNAME`

record defines a canonical name for an alias.

Query:

```
dig example.com CNAME
```

CNAME chains can be relevant during troubleshooting because the name originally used may point to another DNS name.

### MX

`MX`

stands for Mail Exchange.

Query:

```
dig example.com MX
```

MX records specify the mail servers published for a domain and include preference values.

However, an MX record does not prove that the relevant mail server is currently reachable or operational.

It initially documents the DNS configuration.

### NS

With:

`NS`

name server records are queried.

Example:

```
dig example.com NS
```

This can be used to determine which name servers are published for the relevant DNS zone.

## SOA

`SOA`

stands for:

**Start of Authority**

Query:

```
dig example.com SOA
```

The SOA record contains key administrative information about a DNS zone.

This includes, among other things:

primary name server,

responsible contact information in DNS notation,

serial number,

and various timing parameters.

## Serial number

The SOA serial number is particularly useful when examining multiple authoritative name servers.

Different serial numbers may indicate that the servers are not serving the same version of the zone.

The precise interpretation, however, depends on the DNS infrastructure in use.

## TXT

TXT records contain text data.

Query:

```
dig example.com TXT
```

TXT records are used for a variety of purposes.

These can include domain verification and various mail-related policies.

Dig does not automatically interpret the business purpose of a TXT entry.

It displays the DNS data that has been published.

### CAA

CAA records can also be queried directly:

```
dig example.com CAA
```

They contain DNS-based information relating to the issuance of certificates for a domain.

However, Dig does not use this to verify an existing TLS certificate.

It simply queries the relevant DNS record.

### SRV

SRV records containing service information can be queried with:

```
dig NAME SRV
```

.

They can include:

priority,

weight,

port,

and target host.

### PTR and reverse DNS

With:

```
-x
```

Dig performs a simplified reverse DNS query.

Example:

```
dig -x 192.0.2.1
```

Dig automatically constructs the corresponding reverse DNS query and uses:

```
PTR
```

as the record type.

### IPv4 reverse DNS

For IPv4, reverse DNS data is stored under:

```
in-addr.arpa
```

.

When using `-x`, the user does not need to construct the corresponding reverse DNS notation manually.

### IPv6 reverse DNS

`dig -x`

also supports IPv6 addresses.

Dig constructs the corresponding nibble notation under:

`IP6.ARPA`.

### Reverse DNS is not forward DNS

An IPv4 address may have a PTR record that points to a name.

This does not automatically mean that the name in turn resolves via A or AAAA to exactly the same address.

Forward and reverse DNS are separate DNS data sets and should be checked independently where necessary.

### Short output

With:

`+short`

Dig produces greatly reduced output.

Example:

```
dig +short example.com
```

The official documentation describes `+short` as switching from verbose to terse output.

### When `+short` is useful

`+short` is particularly useful for:

quick checks,

shell scripts,

further processing,

and compact output.

For a complete DNS troubleshooting session, however, it is often better to start with the standard output because it shows status, flags and server information.

### Answer Section only

Another useful form is:

```
dig +noall +answer example.com
```

This restricts the output specifically to the answer records.

It is more compact than the standard output, while retaining more DNS structure than a pure `+short` output.

### Forcing IPv4 transport

With:

```
-4
```

Dig uses IPv4 exclusively for transport to the DNS server.

Example:

```
dig -4 example.com
```

This determines the **transport to the DNS server**.

It does not automatically determine the record type being queried.

### Forcing IPv6 transport

With:

```
-6
```

Dig uses IPv6 exclusively to transport the DNS query.

Example:

```
dig -6 example.com
```

Here too, a distinction must be made between:

the transport protocol to the resolver,

and:

the DNS record type being queried.

### Important distinction

```
dig -6 example.com A
```

means:

contact the DNS server over IPv6,

but query an IPv4 A record.

```
dig -4 example.com AAAA
```

means:

contact the DNS server over IPv4,

but query an IPv6 AAAA record.

This distinction is very useful for dual-stack diagnostics.

### UDP and TCP

Dig uses UDP for standard DNS queries by default, unless the query type or an option requires a different transport.

With:

`+tcp`

TCP can be forced explicitly.

Example:

```
dig +tcp example.com
```

The BIND 9.20 documentation also confirms that, after receiving a response marked as truncated over UDP, Dig can normally switch to TCP.

### Why test TCP?

If:

```
dig example.com
```

fails, but:

```
dig +tcp example.com
```

works,

this is an important diagnostic clue.

In that case, differences involving the following should be investigated in particular:

firewall,

UDP transport,

fragmentation,

network path,

or DNS infrastructure.

### DNS does not use UDP only

The blanket statement:

“DNS runs over UDP port 53”

is incomplete.

Traditional DNS uses both UDP and TCP.

In addition, the version of Dig documented here supports further DNS transports.

### DNS over TLS

BIND 9.20 supports in Dig:

`+tls`

for DNS over TLS.

By default, it uses port:

`853`

.

Additional TLS-related options are available for certificate verification.

### Non-standard port

With:

`-p PORT`

a different destination port can be used for the DNS query.

Example:

```
dig @SERVER -p PORT example.com
```

This is useful for DNS servers that listen on a non-standard port for testing or diagnostic purposes.

### Timeout

With:

`+timeout=T`

the query timeout is set in seconds.

The documented default in BIND 9.20 is:

**5 seconds.**

### Retries

With:

`+tries=T`

the number of query attempts can be adjusted.

The documented default is:

**3 attempts.**

Timeout and retry values should be changed deliberately during diagnostics, because very short values can make slow but functioning DNS services appear to have failed.

### Recursion

Standard Dig queries set the:

**RD – Recursion Desired**

bit by default.

This asks a recursive resolver to perform the complete name resolution on behalf of the client.

With:

`+norecurse`

this behaviour can be disabled.

### Recursive resolver

A recursive resolver performs the necessary subsequent DNS queries on behalf of the client.

This is the typical operating mode for DNS servers provided to end devices as resolvers.

### Authoritative name server

An authoritative name server, by contrast, is responsible for the DNS data of a specific zone.

Dig can query such a server directly:

```
dig @AUTHORITATIVE-SERVER example.com A
```

This allows you to compare:

What does the authoritative server return?

What does the recursive resolver in use return?

### Cache differences

A recursive resolver may return data from its cache.

An authoritative server, by contrast, returns the authoritative data for its zone.

As a result, different answers may temporarily be visible while changes propagate.

TTL values play an important role here.

### `+trace`

One of Dig's most powerful diagnostic functions is:

`+trace`

Example:

```
dig +trace example.com
```

Dig follows the delegation path starting from the root name servers.

It performs iterative queries and follows the referrals it receives until the requested name is resolved.

### What `+trace` shows

In simplified form, this can reveal:

Root zone `.`

→ responsible top-level domain

→ name servers of the target zone

→ response from the target zone.

This can identify where a problem occurs within a DNS delegation chain.

### `+trace` is not a standard resolver query

A:

```
dig example.com
```

normally queries the configured recursive resolver.

A:

```
dig +trace example.com
```

instead performs iterative queries itself along the delegation chain.

The two tests therefore examine different DNS paths.

### `@server` together with `+trace`

If `@server` is specified in addition to `+trace`, the BIND documentation states that this server is used only for the initial query for the root name servers.

Subsequent delegation tracing proceeds according to the referrals received.

### DNSSEC data

With:

```
+dnssec
```

Dig requests DNSSEC-related data.

In particular, this sets the DO bit for the DNS query.

This can, for example, make the RRSIG data associated with signed records visible.

### `+dnssec` does not automatically mean full validation

Dig is primarily a DNS query and diagnostic tool.

The mere presence of an RRSIG in the output does not by itself prove that a complete DNSSEC chain of trust has been successfully validated.

For DNSSEC-specific validation tasks, the BIND tool family includes additional tools such as `delv`.

### `+trace` and DNSSEC

With `+trace`, BIND 9.20 also enables `+dnssec` to make the queries more closely resemble the behaviour of a name server.

### DNS flags

Various DNS flags may appear in Dig output.

Important examples include:

`qr` – Query Response

`aa` – Authoritative Answer

`rd` – Recursion Desired

`ra` – Recursion Available

`ad` – Authentic Data

The meaning of a flag must always be considered in the context of the specific response.

### `aa`

The flag:

`aa`

indicates that the response is marked as authoritative.

This is particularly relevant when querying an authoritative server directly.

### `ra`

`ra`

stands for:

**Recursion Available**

and indicates that the responding server offers recursion.

### `ad`

`ad`

stands for:

**Authentic Data**.

In responses from a validating resolver, this flag can indicate that the resolver has validated the relevant data using DNSSEC.

The flag represents a statement made by the responding resolver.

### Multiple queries

The BIND implementation of Dig can perform multiple DNS queries in a single invocation.

Dig also supports a batch mode.

### Batch file

With:

```
-f DATEI
```

queries can be read from a file.

This is useful for repeatable diagnostic procedures.

### User configuration `.digrc`

Dig can read user-specific default options from:

```
${HOME}/.digrc
```

.

This can change the behaviour of a Dig invocation even though the relevant option is not visible directly on the command line.

### Reproducible diagnostics

With:

```
-r
```

reading:

```
${HOME}/.digrc
```

is suppressed.

The BIND documentation recommends this option particularly for scripts that require predictable behaviour.

It can also be useful for documented rescue procedures when user-specific Dig options need to be excluded.

### Exit codes

Dig uses defined exit codes.

Important values are:

`0` – DNS response received, explicitly including NXDOMAIN

`1` – usage/syntax error

`8` – batch file could not be opened

`9` – no response from the server

`10` – internal error.

### Implications for scripts

A script must therefore not merely check whether:

`dig` returns exit code 0.

It must also evaluate the actual DNS status or the expected records.

Otherwise, an NXDOMAIN response could incorrectly be interpreted as successful name resolution.

### Zone transfer

Dig also supports:

AXFR

for full zone transfers.

Example:

```
dig @SERVER example.com AXFR
```

AXFR uses TCP.

### AXFR is not a general discovery method

An authoritative DNS server may restrict zone transfers to explicitly authorised systems.

A refused AXFR attempt therefore does not mean that DNS is malfunctioning.

Restricting access to zone transfers is common and security-relevant in production DNS infrastructures.

Zone transfers must only be performed in authorised environments.

### Typical rescue procedure

A structured DNS diagnostic procedure can be carried out as follows:

**1. Check normal resolution**

```
dig example.com
```

**2. Verify the resolver actually being used**

Check the SERVER line in the output.

**3. Examine A and AAAA separately**

```
dig example.com A
```

```
dig example.com AAAA
```

**4. If necessary, query a specific resolver directly**

```
dig @SERVER example.com A
```

**5. Compare the results from multiple resolvers**

**6. Identify the authoritative name servers**

```
dig example.com NS
```

**7. Query the authoritative server directly**

**8. For delegation problems, `+trace` should be used**

```
dig +trace example.com
```

**9. If a transport problem is suspected, test TCP**

```
dig +tcp example.com
```

**10. If necessary, monitor network traffic in parallel with TCPDump or Wireshark**

### Example: browser reports “Server not found”

First:

```
dig example.com
```

If Dig returns a valid address, basic DNS resolution is working from Dig's perspective.

Further checks are then required to determine whether the problem lies, for example:

in the browser,

in a proxy configuration,

with a different resolver library,

with IPv4/IPv6,

or at the application layer.

### Example: Dig receives no response

If:

```
dig example.com
```

receives no DNS response, another designated DNS server can be tested directly:

```
dig @SERVER example.com
```

If the direct query works, this is an important indication of a problem with the normal resolver configuration or the DNS server normally in use.

### Example: A works, AAAA does not

```
dig example.com A
```

returns an IPv4 address.

```
dig example.com AAAA
```

returns no corresponding IPv6 record.

This does not automatically indicate a general DNS failure.

The two record types must be evaluated separately.

### Example: resolver returns stale data

A recursive resolver returns a different address from the authoritative name server.

In that case, the following should be examined in particular:

TTL,

cache,

authoritative data,

and the timing of any DNS change.

### Example: delegation problem

A name does not resolve reliably through certain resolvers.

With:

```
dig +trace example.com
```

the delegation chain can be examined.

This makes it possible to see which name servers are referenced at each level and where iterative resolution fails.

### Example: UDP/TCP difference

First:

```
dig example.com
```

then:

```
dig +tcp example.com
```

If only the TCP query works, the DNS transport path should be investigated more closely.

TCPDump or Wireshark can be used in parallel to check which DNS packets are actually sent and received.

### Dig and Wireshark

Wireshark can analyse the DNS packets generated by Dig in detail.

A display filter:

```
dns
```

shows the corresponding DNS traffic.

This can be used to verify:

which query Dig actually sent,

to which server,

which record type was requested,

which response was returned,

and which DNS flags or records it contains.

### Dig and TCPDump

TCPDump provides terminal-based monitoring of DNS traffic.

A typical filter is:

```
sudo tcpdump -nn port 53
```

This can be used to observe traditional DNS queries and responses over TCP or UDP port 53.

DNS over TLS on port 853 requires a correspondingly different capture filter.

### Dig and MTR

Dig answers the question:

**What does DNS return?**

MTR answers the question:

**How does the network path to a target behave?**

If a DNS server does not respond, MTR can provide additional clues about the IP path.

However, MTR is not a substitute for DNS analysis.

### Dig and Nmap

Nmap can check whether a DNS service is visible on the network at a target system.

Dig can then send an actual DNS query to that service.

The two therefore complement each other:

**Nmap → Is a service reachable or detectable?**

**Dig → What DNS response does the name server return?**

### Dig and ARP Scan

For a local DNS server, ARP Scan can first help determine whether its IPv4 system is visible on the local Layer 2 network.

Dig then examines the DNS service itself.

This separates the different layers:

**ARP → local neighbourhood**

**IP/transport → reachability**

**DNS → name resolution**

### Limitations

Dig tests DNS.

It does not prove that a web server, mail server or other service is actually working.

The presence of an A or AAAA record does not prove that the published address is reachable.

An MX record does not prove that the mail server is reachable.

A PTR record does not prove symmetrical forward resolution.

A DNS response may come from a resolver cache.

Different resolvers may temporarily return different data.

`+dnssec` alone is not equivalent to complete independent DNSSEC validation.

Exit code 0 explicitly does not mean that the requested name exists; NXDOMAIN is also a received DNS response.

### Safety notes

Only examine DNS servers directly when authorised to do so.

Perform zone transfers only in environments where this is explicitly permitted.

When comparing results, always document which resolver was queried.

Check A and AAAA records separately when troubleshooting dual-stack issues.

Do not equate DNS reachability with service reachability.

Do not confuse `+trace` with a normal recursive resolver query.

In scripts, evaluate NXDOMAIN and other DNS status values in addition to the process exit code.

DNS output may contain internal host names, IP addresses and infrastructure information and should be protected accordingly.

### Technical role within JUMBO BLOCK Rescue

Within the network section of JUMBO BLOCK Rescue, Dig provides **targeted DNS and resolver analysis**.

The network tools therefore now cover clearly separated layers:

**A.17 Nmap**

active host, port and service analysis.

**A.18 Wireshark**

graphical packet and protocol analysis.

**A.19 TCPDump**

terminal-based packet capture and filtering.

**A.20 ARP Scan**

local IPv4 device discovery on Layer 2.

**A.21 MTR**

continuous path, latency and packet-loss analysis.

**A.22 Dig**

DNS, resolver, resource-record and delegation analysis.

A particularly effective rescue procedure for a name-resolution problem is therefore:

**ARP Scan → Is the local DNS server visible on Layer 2?**

**MTR → Is its IP path reachable, and does it show any anomalies?**

**Dig → What DNS response does it actually return?**

**TCPDump → Are the query and response visible on the network?**

**Wireshark → What do the DNS packets contain in detail?**

Dig therefore establishes a crucial distinction:

**The network is working**

does not automatically mean:

**DNS is working.**

And:

**DNS returns an address**

does not automatically mean:

**The addressed service is working.**

This clear separation of diagnostic layers is precisely what makes Dig an important component of JUMBO BLOCK Rescue.

# A.23 Whois

**Bereich:** Netzwerk
**Programm:** Whois
**Baseline 20261001:** Whois 5.6.3
**Installiertes Paket:** `whois 5.6.3`
**Bedienung:** Kommandozeile
**Aufgabe:** Abfrage öffentlicher WHOIS-Verzeichnisdienste für Domains, IP-Netzbereiche, AS-Nummern und weitere Registry-Objekte

## Zweck

Whois ist ein Kommandozeilenclient für WHOIS-Verzeichnisdienste.

Die installierte Version sucht nach Objekten in Datenbanken, die über das WHOIS-Protokoll gemäß RFC 3912 erreichbar sind.

WHOIS-Datenbanken können – abhängig von Registry, Objektart und deren Veröffentlichungsregeln – Informationen über:

Domains,

IP-Adressbereiche,

IPv4- und IPv6-Netze,

Autonomous-System-Nummern,

Netzwerkorganisationen,

Registrare,

technische beziehungsweise administrative Registry-Objekte

und Kontakt- oder Abuse-Informationen

bereitstellen.

Welche Informationen tatsächlich zurückgegeben werden, bestimmt der jeweils angesprochene WHOIS-Dienst.

## Einsatzgebiet im Rescue-System

Innerhalb von JUMBO BLOCK Rescue eignet sich Whois insbesondere zur:

Zuordnung öffentlicher IP-Adressen zu registrierten Netzbereichen,

Ermittlung der für einen IP-Block zuständigen Registry beziehungsweise Organisation,

Untersuchung von Autonomous-System-Nummern,

Abfrage von Domain-Registrierungsinformationen,

Ermittlung zuständiger Registrar- oder Registry-Systeme,

Suche nach veröffentlichten Abuse-Kontakten,

Einordnung unbekannter öffentlicher Netzwerkziele,

und Ergänzung von Nmap-, MTR-, Dig-, TCPDump- und Wireshark-Ergebnissen um Registryinformationen.

## WHOIS ist kein DNS

WHOIS und DNS erfüllen unterschiedliche Aufgaben.

DNS beantwortet beispielsweise:

**Welche IP-Adresse gehört zu einem Namen?**

WHOIS kann dagegen beispielsweise Hinweise darauf liefern:

**Welchem registrierten Netzbereich ist diese öffentliche IP-Adresse zugeordnet?**

Dig und Whois ergänzen sich deshalb, ersetzen sich aber nicht.

## WHOIS ist keine Geo-IP-Datenbank

WHOIS-Daten können Adress- oder Länderangaben enthalten.

Daraus darf jedoch nicht automatisch der tatsächliche physische Standort eines Servers oder Endgeräts abgeleitet werden.

Eine registrierte Organisation kann ihre Infrastruktur an anderen Standorten betreiben.

Netze können:

international genutzt,

weiterdelegiert,

an Kunden vergeben,

über Cloud-Infrastrukturen verwendet

oder an anderen Orten geroutet

werden.

## WHOIS ist keine Personensuche

WHOIS ist technisch ein Registry- beziehungsweise Verzeichnisabfragesystem.

Es sollte nicht als Werkzeug verstanden werden, mit dem sich zuverlässig die natürliche Person hinter:

einer Domain,

einer IP-Adresse

oder einem Server

bestimmen lässt.

Insbesondere moderne Domainregistrierungen enthalten häufig datenschutzbedingt redigierte oder durch Proxy-/Privacy-Dienste ersetzte Kontaktinformationen.

## Grundsyntax

Der grundlegende Aufruf lautet:

```
whois OBJECT
```

Das Objekt kann je nach Abfrage beispielsweise sein:

Domainname,

IPv4-Adresse,

IPv6-Adresse,

AS-Nummer,

NIC-Handle

oder ein anderer vom betreffenden WHOIS-Dienst unterstützter Objekttyp.

## Automatische Serverauswahl

Whois 5.6.3 versucht selbst zu bestimmen, welcher WHOIS-Server für das angegebene Objekt geeignet ist.

Dadurch genügt für viele Standardabfragen:

```
whois OBJECT
```

ohne dass der Benutzer den Registryserver vorher kennen muss.

## Domain abfragen

Beispiel:

```
whois example.com
```

Der Client versucht, den für diese Domain geeigneten WHOIS-Dienst zu bestimmen und die Anfrage dorthin zu richten.

Welche Daten zurückgegeben werden, hängt von:

Top-Level-Domain,

Registry,

Registrar,

WHOIS-Modell,

Datenschutzregeln

und Serverimplementierung

ab.

## IP-Adresse abfragen

Auch eine öffentliche IPv4-Adresse kann unmittelbar abgefragt werden:

```
whois 192.0.2.1
```

Die hier verwendete Adresse dient ausschließlich als Dokumentationsbeispiel.

Bei einer realen öffentlichen Adresse kann die Antwort beispielsweise Informationen zum registrierten Netzbereich und zur zuständigen Organisation enthalten.

## IPv6-Adresse

Whois kann ebenfalls IPv6-Objekte abfragen.

Beispielprinzip:

```
whois IPv6-ADRESSE
```

Der Client bestimmt dabei nach seinen internen Regeln den geeigneten WHOIS-Dienst.

## AS-Nummer

Autonomous-System-Nummern können beispielsweise in der Form:

```
AS64496
```

abgefragt werden.

Eine WHOIS-Antwort kann abhängig von der Registry Informationen zum betreffenden AS-Objekt enthalten.

## Autonomous System

Ein Autonomous System ist im Internet-Routing ein administrativ kontrollierter Routingbereich, dem eine Autonomous-System-Nummer – ASN – zugeordnet ist.

WHOIS-Daten können helfen, eine in Netzwerkdiagnosen auftretende ASN einer registrierten Organisation beziehungsweise einem Registryobjekt zuzuordnen.

## MTR und Whois

MTR kann den Netzwerkpfad zu einem entfernten Ziel sichtbar machen.

Whois kann anschließend zur Einordnung öffentlicher IP-Adressen beziehungsweise AS-Informationen einzelner Netzbereiche verwendet werden.

Damit entsteht die Kombination:

**MTR → Wo verläuft der sichtbare IP-Pfad?**

**Whois → Welchen registrierten Netzobjekten sind öffentliche Adressen beziehungsweise AS-Objekte zugeordnet?**

### Bestimmten WHOIS-Server auswählen

Mit:

`-h HOST`

beziehungsweise:

`--host=HOST`

wird ein bestimmter WHOIS-Server ausdrücklich ausgewählt.

Grundprinzip:

`whois -h SERVER OBJECT`

Damit kann die automatische Serverauswahl umgangen werden.

### Warum einen Server ausdrücklich auswählen?

Dies ist nützlich, wenn:

eine bestimmte Registry direkt abgefragt werden soll,

mehrere WHOIS-Quellen verglichen werden sollen,

ein Referral überprüft werden soll,

oder bekannt ist, dass ein bestimmtes Objekt in einer bestimmten Datenbank untersucht werden muss.

### Port festlegen

Mit:

`-p PORT`

beziehungsweise:

`--port=PORT`

kann der Zielport ausdrücklich angegeben werden.

Klassisches WHOIS nach RFC 3912 verwendet TCP Port:

**43**.

Ein abweichender Port kann für besondere beziehungsweise nichtstandardisierte Dienste angegeben werden.

### WHOIS verwendet TCP

Klassisches WHOIS arbeitet über eine TCP-Verbindung.

Der Client verbindet sich mit dem WHOIS-Server, sendet die Anfrage und empfängt die textuelle Antwort.

Das unterscheidet WHOIS beispielsweise von klassischen DNS-Standardabfragen, die häufig zunächst UDP verwenden.

### IANA zuerst abfragen

Whois 5.6.3 besitzt die Option:

`-I`

Damit wird zunächst:

`whois.iana.org`

abgefragt.

Anschließend folgt der Client dem von IANA gelieferten Referral zum für die Anfrage autoritativen WHOIS-Server.

Diese Funktion unterstützt:

IP-Adressen,

AS-Nummern

und Domains.

### Datenschutz bei `-I`

Die Dokumentation weist ausdrücklich darauf hin:

Bei Verwendung von `-I` erhält der IANA-WHOIS-Server die vollständige Anfrage.

Dies sollte berücksichtigt werden, wenn Abfragen sensible interne Untersuchungszusammenhänge erkennen lassen könnten.

### Referral

WHOIS ist historisch auf verschiedene Registries und Registrare verteilt.

Ein Server kann deshalb auf einen anderen WHOIS-Dienst verweisen.

Ein solcher Verweis wird als Referral bezeichnet.

Der Client kann entsprechenden Referral-Informationen folgen.

### Registry und Registrar

Bei Domainabfragen muss zwischen Registry und Registrar unterschieden werden.

Die **Registry** verwaltet die zentrale Registrierungsdatenbank einer Top-Level-Domain beziehungsweise des entsprechenden Namensraums.

Der **Registrar** ist der Anbieter, über den eine konkrete Domain registriert beziehungsweise verwaltet wird.

Je nach WHOIS-Modell können Registry- und Registrar-Daten auf unterschiedlichen Servern liegen.

### Thin Registry

Bei bestimmten gTLD-WHOIS-Systemen können Registryserver nur einen Teil der Registrierungsinformationen liefern und auf den zuständigen Registrar verweisen.

Whois 5.6.3 besitzt dafür spezielle Referral-Logik.

### Referral-Rekursion deaktivieren

Mit:

```
--no-recursion
```

kann die Rekursion von Registry- zu Registrarservern deaktiviert werden.

Dies ist hilfreich, wenn ausdrücklich nur die Antwort der zunächst angesprochenen Registry betrachtet werden soll.

### Warum Referral-Kontrolle wichtig ist

Ohne diese Unterscheidung kann eine WHOIS-Ausgabe so wirken, als stamme sie vollständig aus einer einzigen Datenquelle.

Tatsächlich kann der Client jedoch einem Referral gefolgt sein.

Für eine dokumentierte Untersuchung sollte deshalb berücksichtigt werden, welche Quelle die jeweilige Information geliefert hat.

### Ausführliche Programmausgabe

Mit:

```
--verbose
```

arbeitet der Client ausführlicher.

Dies kann bei der technischen Diagnose der WHOIS-Abfrage hilfreich sein.

### Rechtliche Hinweise ausblenden

Mit:

```
-H
```

werden bestimmte rechtliche Hinweise beziehungsweise Disclaimer unterdrückt, die einige Registries in ihren Antworten ausgeben.

Das verändert nicht die Registrydaten selbst, sondern deren Darstellung durch den Client.

### Registry-Hinweise können wichtig sein

Bei dokumentations- oder compliancebezogenen Untersuchungen sollte nicht automatisch jeder Disclaimer entfernt werden.

Registryhinweise können Informationen enthalten über:

Nutzungsbedingungen,

Weiterverwendung,

Abfragelimits

oder Interpretation der Daten.

### RIPE-artige WHOIS-Dienste

Whois 5.6.3 unterstützt zusätzlich eine Reihe von Optionen, die von:

`whois.ripe.net`

und einigen ähnlich arbeitenden WHOIS-Diensten verstanden werden.

Diese Optionen sind **nicht universell auf alle WHOIS-Server übertragbar**.

### Wichtige Konsequenz

Eine Option, die bei RIPE funktioniert, muss bei einer anderen Registry nicht dieselbe Bedeutung besitzen oder überhaupt unterstützt werden.

WHOIS ist hinsichtlich der Abfragesprache nicht weltweit vollständig vereinheitlicht.

### Kurze IP-/Abuse-Ausgabe

Bei RIPE-artigen Servern kann:

`-b`

eine kurze Darstellung von IP-Adressbereichen mit Abuse-Kontakt anfordern.

Dies kann für eine schnelle Einordnung eines öffentlichen Netzbereichs nützlich sein.

Die Option ist jedoch serverabhängig.

### Objektfilterung deaktivieren

Bei unterstützenden RIPE-artigen Servern kann:

`-B`

die serverseitige Objektfilterung deaktivieren.

Die Manpage beschreibt dies unter anderem als Möglichkeit, E-Mail-Adressen anzuzeigen.

Ob und welche Daten tatsächlich zurückgegeben werden, bestimmt der Server.

Die Option hebt keine serverseitigen Datenschutz- oder Zugriffsregeln auf.

### Exakte Netzpräfixsuche

Mit:

`-x`

kann bei unterstützenden RIPE-artigen WHOIS-Diensten nach einer exakten Übereinstimmung eines Netzwerkpräfixes gesucht werden.

Dies ist insbesondere bei verschachtelten Netzzuweisungen nützlich.

### Weniger spezifische Objekte

Für unterstützende Server existieren Optionen zur Suche nach weniger spezifischen Netzobjekten.

`-l`

liefert eine Ebene weniger spezifisch.

`-L`

liefert alle Ebenen weniger spezifischer Objekte.

Damit kann die Hierarchie einer Netzzuweisung untersucht werden.

### Spezifischere Objekte

Entsprechend stehen:

`-m`

für eine Ebene spezifischere Objekte

und:

`-M`

für alle Ebenen spezifischerer Objekte

zur Verfügung.

Dies kann bei delegierten beziehungsweise unterteilten Netzbereichen hilfreich sein.

### Warum Netz-Hierarchien relevant sind

Eine öffentliche IP-Adresse kann Teil eines großen Providerblocks sein, innerhalb dessen ein kleinerer Bereich einem Kunden oder Dienst zugeordnet wurde.

Nur die Betrachtung des größten Netzblocks kann deshalb zu einer zu groben Zuordnung führen.

Weniger und stärker spezifische Objekte helfen, diese Hierarchie zu untersuchen.

### Quellen auswählen

Mit:

`-s SOURCE[,SOURCE]...`

kann bei RIPE-artigen Servern festgelegt werden, welche gespiegelten Datenquellen durchsucht werden sollen.

Die Reihenfolge der Quellen ist relevant.

Welche Quellen verfügbar sind, kann über eine Serverabfrage ermittelt werden.

### Serverinformationen

Mit:

`-q KEYWORD`

können unterstützende RIPE-artige Server nach Informationen über sich selbst gefragt werden.

Dokumentierte Schlüsselwörter sind unter anderem:

`version`

`sources`

`types`.

### Objekttypen

Mit:

`-T TYPE[,TYPE]...`

kann die Suche bei unterstützenden Servern auf bestimmte Objekttypen eingeschränkt werden.

Dies kann große oder mehrdeutige Ergebnisse präzisieren.

### Objektvorlagen

Mit:

`-t TYPE`

kann die Vorlage eines Objekttyps abgefragt werden.

Mit:

`-v TYPE`

kann eine ausführlichere Vorlage angefordert werden.

Diese Funktionen beziehen sich auf Server, die die entsprechende RIPE-artige Abfragesprache unterstützen.

### Inverse Suche

Mit:

```
-i ATTR[,ATTR]...
```

können bei unterstützenden Servern Objekte gesucht werden, deren angegebene Attribute einen bestimmten Wert besitzen.

Dies ist eine weiterführende Registrydatenbankfunktion und keine allgemeine Suchfunktion über das Internet.

### Kontakt-Rekursion deaktivieren

Bei RIPE-artigen Abfragen kann:

```
-r
```

rekursive Suchvorgänge nach Kontaktinformationen deaktivieren.

Dies unterscheidet sich von:

```
--no-recursion
```

welches die Rekursion von Registry- zu Registrarservern betrifft.

Die beiden Optionen dürfen daher nicht verwechselt werden.

### Referral auf RIPE-artigen Servern

Mit:

```
-R
```

kann bei entsprechenden Servern das Folgen bestimmter Referrals deaktiviert und stattdessen die lokale Kopie des Objekts angezeigt werden.

Auch diese Option ist serverspezifisch.

### Konfigurationsdatei

Whois 5.6.3 berücksichtigt:

```
/etc/whois.conf
```

sofern diese Datei vorhanden ist.

Damit können alternative WHOIS-Server definiert werden, die die eingebauten Regeln ergänzen oder überschreiben.

### Aufbau von `whois.conf`

Die Datei ordnet Suchmuster bestimmten WHOIS-Servern zu.

Vereinfacht:

```
MUSTER WHOIS-SERVER
```

Das Muster wird gegen das abzufragende WHOIS-Objekt geprüft.

Bei einem Treffer wird der zugeordnete Server verwendet.

### Bedeutung für Rescue-Diagnosen

Eine lokale `whois.conf` kann beeinflussen, welcher WHOIS-Server tatsächlich abgefragt wird.

Bei unerwarteten Ergebnissen sollte daher berücksichtigt werden, dass die automatische Serverwahl lokal überschrieben sein kann.

Aus der Paketbaseline allein wird nicht abgeleitet, ob JUMBO BLOCK Rescue eine angepasste `/etc/whois.conf` besitzt.

### Umgebungsvariable `WHOIS_SERVER`

Die Variable:

`WHOIS_SERVER`

kann einen Server vorgeben, wenn Whois selbst nicht bestimmen kann, wo das betreffende Objekt zu finden ist.

Ist diese Variable nicht gesetzt und kann der Client keinen passenden Server bestimmen, sieht die dokumentierte Fallback-Logik für bestimmte Objekte weitere Standardserver vor.

### Umgebungsvariable `WHOIS_OPTIONS`

Mit:

`WHOIS_OPTIONS`

können Optionen definiert werden, die vor den explizit auf der Kommandozeile angegebenen Optionen ausgewertet werden.

Dies kann das Verhalten eines Aufrufs verändern.

Für reproduzierbare Untersuchungen sollten solche Umgebungseinflüsse berücksichtigt werden.

### Internationalisierte Domains

Whois 5.6.3 unterstützt die Verarbeitung internationalisierter Domainnamen.

Kommandozeilenargumente werden entsprechend der aktuellen Locale interpretiert und für IDN-Domains in eine ASCII-kompatible Darstellung umgesetzt.

### ACE / Punycode

WHOIS-Konfigurationen verwenden für internationalisierte Domainnamen die ASCII Compatible Encoding – ACE.

Solche Namen sind häufig an der:

`xn--`

Darstellung erkennbar.

## Zeichenkodierung

Das klassische WHOIS-Protokoll definiert keine einheitliche Zeichenkodierung für Zeichen außerhalb von ASCII.

Die Manpage weist deshalb darauf hin, dass Implementierungen in diesem Bereich stark variieren.

Whois versucht bei bekannten Servern gegebenenfalls, die Ausgabe passend zur aktuellen System-Locale umzuwandeln.

## WHOIS-Daten sind nicht einheitlich

Anders als beispielsweise DNS Resource Records besitzen WHOIS-Antworten weltweit kein einheitliches, überall identisches Ausgabeformat.

Unterschiedliche Registries können:

andere Feldnamen,

andere Objektmodelle,

andere Referralmechanismen,

andere Datenschutzregeln

und andere Textformate

verwenden.

## Automatische Auswertung ist daher schwierig

Ein Shell-Skript sollte WHOIS-Ausgaben nicht ohne Weiteres so behandeln, als hätten alle Registries dieselbe Struktur.

Ein Parser, der für eine Registry funktioniert, kann bei einer anderen fehlschlagen.

## Domain-WHOIS und Datenschutz

Bei Domainregistrierungen können personenbezogene Felder:

redigiert,

durch Privacy-Dienste ersetzt,

nicht veröffentlicht

oder nur über andere geregelte Verfahren verfügbar

sein.

Das Fehlen eines Namens oder einer Adresse ist daher nicht automatisch ein technischer Fehler.

### Registrant ist nicht zwingend Betreiber

Selbst wenn ein WHOIS-Dienst Registrierungsinformationen anzeigt, darf daraus nicht automatisch geschlossen werden, dass die genannte Organisation:

den Server physisch betreibt,

den Inhalt einer Website erstellt,

den Netzwerkverkehr verursacht,

oder die betreffende Infrastruktur technisch administriert.

Domaininhaber, Registrar, Hostinganbieter, CDN, Cloudanbieter und tatsächlicher Dienstbetreiber können unterschiedliche Stellen sein.

### IP-WHOIS und Netzzuordnung

Bei öffentlichen IP-Adressen beschreibt WHOIS typischerweise die Registrierung beziehungsweise Zuweisung eines Netzbereichs.

Dies ist besonders hilfreich, um festzustellen, ob eine Adresse beispielsweise einem:

Internetprovider,

Hostinganbieter,

Cloudnetz,

Unternehmen

oder anderen registrierten Netzbetreiber

zugeordnet ist.

Die genaue Aussagekraft hängt vom jeweiligen Registryobjekt ab.

### IP-Zuordnung beweist keine Benutzeridentität

Aus der registrierten Organisation eines IP-Netzblocks lässt sich nicht automatisch bestimmen, welcher konkrete Benutzer eine einzelne IP-Adresse zu einem bestimmten Zeitpunkt verwendet hat.

Das gilt insbesondere bei:

dynamischen IP-Adressen,

NAT,

Carrier-Grade NAT,

VPN,

Proxy,

Cloudsystemen

und gemeinsam genutzter Infrastruktur.

### ASN-Zuordnung

WHOIS kann Informationen zu registrierten Autonomous-System-Objekten liefern.

Diese Angaben helfen bei der administrativen Einordnung eines AS.

Sie ersetzen jedoch keine Echtzeit-BGP-Analyse.

### Registrierung und tatsächliches Routing

Ein Registryeintrag und die aktuell im Internet sichtbare Routingrealität sind unterschiedliche Informationsquellen.

Ein Netz kann beispielsweise über:

andere Upstream-Provider,

mehrere Autonomous Systems,

Anycast

oder komplexe Routingbeziehungen

erreichbar sein.

WHOIS beschreibt Registryinformationen, nicht automatisch den vollständigen aktuellen Routingzustand.

### Whois und Dig

Eine sinnvolle Kombination lautet:

```
dig example.com A
```

ermittelt die veröffentlichte IPv4-Adresse.

Anschließend:

```
whois IP-ADRESSE
```

liefert Registryinformationen zum betreffenden öffentlichen Netzbereich.

Damit wird getrennt:

**DNS → Name zu Adresse**

**WHOIS → Adresse zu registriertem Netzobjekt**

### Whois und MTR

MTR kann öffentliche Routeradressen entlang eines Pfads sichtbar machen.

Whois kann einzelne dieser Adressen beziehungsweise zugehörige Netzobjekte administrativ einordnen.

Dabei muss berücksichtigt werden:

Ein MTR-Hop kann eine Interface-Adresse anzeigen, deren WHOIS-Zuordnung nicht unmittelbar den tatsächlichen Betreiber des gesamten Pfadsegments beschreibt.

### Whois und Nmap

Nmap kann einen öffentlichen Host technisch untersuchen.

Whois kann die öffentliche IP-Adresse registryseitig einordnen.

Vereinfacht:

**Nmap → Was ist netzwerktechnisch am Ziel sichtbar?**

**Whois → Welcher Registryzuordnung gehört der öffentliche Adressbereich an?**

### Whois und Wireshark/TCPDump

Wireshark oder TCPDump können eine unbekannte öffentliche Gegenstelle im Netzwerkverkehr sichtbar machen.

Whois kann anschließend zur administrativen Einordnung der betreffenden öffentlichen IP-Adresse verwendet werden.

Ein sinnvoller Analyseablauf lautet:

**Capture → unbekannte IP feststellen → Whois-Abfrage → Registryzuordnung dokumentieren**

### Whois und ARP Scan

ARP Scan arbeitet im unmittelbar lokalen IPv4-Layer-2-Netz und zeigt MAC-Adressen.

Whois arbeitet mit öffentlichen Registrydaten.

Private lokale Adressen wie RFC-1918-Adressen besitzen keine individuelle öffentliche WHOIS-Zuordnung zu einem lokalen Endgerät.

Für ein lokales Gerät sind daher beispielsweise:

ARP,

DHCP,

Switchinformationen

oder lokale Inventardaten

relevanter als eine öffentliche WHOIS-Abfrage.

### Typischer Rescue-Ablauf: unbekannte öffentliche IP-Adresse

**1. IP-Adresse dokumentieren**

Beispielsweise aus:

TCPDump,

Wireshark,

Nmap

oder einem Log.

**2. WHOIS-Abfrage durchführen**

```
whois IP-ADRESSE
```

**3. Netzbereich identifizieren**

Prüfen, welchem registrierten Präfix beziehungsweise Netzobjekt die Adresse zugeordnet ist.

**4. Organisation und Registry prüfen**

**5. Falls vorhanden Referral beziehungsweise spezifischeres Objekt untersuchen**

**6. Ergebnisse nicht mit einer persönlichen Identität gleichsetzen**

**7. Bei Bedarf technische Analyse mit Nmap, MTR oder Dig fortsetzen**

## Typischer Rescue-Ablauf: unbekannte Domain

**1. DNS prüfen**

```
dig DOMAIN
```

**2. Domain-WHOIS prüfen**

```
whois DOMAIN
```

**3. Registrar beziehungsweise Registryinformationen dokumentieren**

**4. Aufgelöste öffentliche IP-Adresse separat untersuchen**

```
whois IP-ADRESSE
```

**5. Domainregistrierung und IP-Netzzuordnung getrennt bewerten**

Eine Domain kann bei Organisation A registriert sein und auf Infrastruktur von Organisation B zeigen.

## Typischer Rescue-Ablauf: MTR-Hop einordnen

MTR zeigt eine unbekannte öffentliche Routeradresse.

Diese Adresse kann mit:

```
whois IP-ADRESSE
```

abgefragt werden.

Dadurch kann ein Hinweis auf den registrierten Netzbetreiber entstehen.

Bei mehreren aufeinanderfolgenden Hops können dadurch administrative Netzgrenzen sichtbarer werden.

Die WHOIS-Zuordnung ersetzt jedoch keine vollständige Routinganalyse.

### Typischer Rescue-Ablauf: AS-Nummer

Ist eine AS-Nummer bekannt:

```
whois ASNUMMER
```

kann das entsprechende Registryobjekt abgefragt werden.

Dies kann Informationen über die registrierte Organisation und weitere technische Registrydaten liefern.

### Private IPv4-Adressen

Private IPv4-Adressbereiche sind nicht wie öffentliche Internetadressen einzelnen lokalen Organisationen über das globale WHOIS-System zugeordnet.

Eine Abfrage einer privaten Adresse kann daher höchstens Informationen über den reservierten Adressbereich liefern, nicht darüber, welches lokale Gerät im untersuchten Netzwerk diese Adresse verwendet.

### Loopback und reservierte Adressen

Dasselbe Grundprinzip gilt für andere reservierte oder spezielle Adressbereiche.

WHOIS ist keine lokale Inventardatenbank.

### Offline-Betrieb

Whois benötigt für eine aktuelle externe WHOIS-Abfrage eine Netzwerkverbindung zum betreffenden WHOIS-Server.

Ohne Netzwerkzugang kann der Client keine aktuelle Antwort einer externen Registry abrufen.

Damit unterscheidet er sich beispielsweise von Werkzeugen, die ausschließlich lokale Daten untersuchen.

### WHOIS-Server kann nicht erreichbar sein

Eine fehlgeschlagene WHOIS-Abfrage beweist nicht:

dass das Objekt nicht existiert.

Mögliche Ursachen sind beispielsweise:

keine Netzwerkverbindung,

WHOIS-Port blockiert,

Server nicht erreichbar,

Rate Limiting,

falscher Server,

oder serverseitige Einschränkungen.

### Rate Limits

WHOIS-Dienste können Abfragen begrenzen.

Zu viele automatisierte Anfragen können:

gedrosselt,

temporär blockiert

oder abgewiesen

werden.

Whois sollte deshalb nicht für unkontrollierte Massensuchen verwendet werden.

### Datenschutz und Nutzungsbedingungen

WHOIS-Antworten können personenbezogene oder kontaktbezogene Informationen enthalten.

Die Tatsache, dass Daten öffentlich abrufbar sind, bedeutet nicht automatisch, dass sie ohne Einschränkung für jeden Zweck weiterverarbeitet oder veröffentlicht werden dürfen.

Registryhinweise und geltende Datenschutz- beziehungsweise Nutzungsregeln sind zu beachten.

### Daten können veraltet sein

WHOIS-Daten sind Registrydaten.

Sie können:

veraltet,

unvollständig,

redigiert

oder aufgrund von Delegationen nur indirekt aussagekräftig

sein.

Ein WHOIS-Ergebnis sollte daher als Informationsquelle dokumentiert, aber nicht überinterpretiert werden.

### WHOIS und RDAP

Viele Registries stellen inzwischen zusätzlich beziehungsweise zunehmend RDAP – Registration Data Access Protocol – bereit.

Der hier dokumentierte Befehl `whois` ist jedoch ein Client für den klassischen WHOIS-Verzeichnisdienst.

Die Existenz moderner RDAP-Dienste ändert nicht die Funktion des in der Baseline installierten Whois-Clients.

## Grenzen

Whois liefert Registry- beziehungsweise Verzeichnisdaten.

Es führt keinen Portscan durch.

Es analysiert keine Netzwerkpakete.

Es löst keine DNS-Probleme.

Es bestimmt nicht zuverlässig den physischen Standort eines Servers.

Es identifiziert nicht zuverlässig den tatsächlichen Benutzer einer IP-Adresse.

Es beweist nicht den Betreiber einer Website.

Es beweist nicht das Eigentum an einem physischen System.

Es liefert keine vollständige Echtzeitansicht des Internet-Routings.

WHOIS-Ausgabeformate und verfügbare Felder unterscheiden sich zwischen Registries.

Datenschutzregeln können wesentliche Felder ausblenden.

## Sicherheitshinweise

WHOIS-Ergebnisse immer als Registryinformationen und nicht als Identitätsbeweis behandeln.

Registrierte Organisation nicht automatisch mit tatsächlichem Benutzer oder technischem Betreiber gleichsetzen.

Domainregistrierung und Hostinginfrastruktur getrennt untersuchen.

Bei IP-Adressen Netzblockzuordnung und konkrete Endgeräteidentität nicht verwechseln.

Bei MTR-Hops WHOIS-Zuordnung nicht als vollständigen Beweis der Routingverantwortung interpretieren.

Referral-Quellen bei wichtigen Untersuchungen dokumentieren.

WHOIS-Antworten können personenbezogene Daten enthalten.

Rate Limits und Nutzungsbedingungen der Registry beachten.

Bei automatisierter Verarbeitung unterschiedliche Serverformate berücksichtigen.

## Technische Einordnung in JUMBO BLOCK Rescue

Whois übernimmt innerhalb des Netzwerkbereichs von JUMBO BLOCK Rescue die Rolle der **administrativen und registrybasierten Einordnung öffentlicher Netzwerkobjekte**.

Damit ergibt sich inzwischen eine sehr klare Arbeitsteilung:

### A.17 Nmap

aktive Host-, Port- und Dienstanalyse.

### A.18 Wireshark

grafische Paket- und Protokollanalyse.

### A.19 TCPDump

terminalbasierte Paketaufzeichnung und Filterung.

### A.20 ARP Scan

lokale IPv4-Geräteerkennung auf Layer 2.

### A.21 MTR

fortlaufende Pfad-, Latenz- und Paketverlustanalyse.

### A.22 Dig

DNS-, Resolver- und Delegationsanalyse.

### A.23 Whois

Registry-, Netzblock-, Domain- und AS-Einordnung.

Ein typischer kombinierter Rescue-Workflow kann damit lauten:

**Dig → Welche Adresse liefert DNS?**

**Whois → Welchem öffentlichen Registry-Netzbereich ist diese Adresse zugeordnet?**

**MTR → Über welchen sichtbaren Netzwerkpfad wird das Ziel erreicht?**

**Nmap → Welche Ports und Dienste sind am autorisierten Ziel sichtbar?**

**TCPDump/Wireshark → Welche Kommunikation findet tatsächlich statt?**

Whois liefert dabei bewusst eine andere Informationsklasse als die übrigen Werkzeuge:

Es untersucht nicht primär das Verhalten des Netzwerks.

Es fragt die **registrierten Verwaltungs- und Zuordnungsdaten** zu Netzwerkobjekten ab.

Für die korrekte Interpretation gilt deshalb als zentrale Regel:

**WHOIS sagt, was eine Registry über ein Objekt veröffentlicht – nicht automatisch, wer ein konkretes System tatsächlich betreibt, benutzt oder kontrolliert.**

# A.24 Chromium

**Bereich:** Netzwerk
**Programm:** Chromium
**Baseline 20261001:** Chromium 154.0.8037.57
**Installierte Pakete:** `chromium 154.0.8037.57-1~deb13u1`, `chromium-common 154.0.8037.57-1~deb13u1`, `chromium-l10n 154.0.8037.57-1~deb13u1`, `chromium-sandbox 154.0.8037.57-1~deb13u1`
**Bedienung:** Grafische Benutzeroberfläche
**Aufgabe:** Webzugriff, Zugriff auf lokale webbasierte Geräteoberflächen sowie HTTP-/HTTPS- und Webanwendungsdiagnose

## Zweck

Chromium ist ein Webbrowser und zugleich die Open-Source-Basis der Chromium-Browserfamilie.

Innerhalb von JUMBO BLOCK Rescue besitzt Chromium eine weitergehende diagnostische Funktion.

Viele moderne Netzwerkgeräte und Infrastruktursysteme werden über webbasierte Administrationsoberflächen bedient.

Dazu gehören beispielsweise:

Router,

Firewalls,

Managed Switches,

Access Points,

NAS-Systeme,

Servermanagementsysteme,

Drucker,

USV-Systeme,

Storage-Systeme,

Virtualisierungsplattformen

und andere Appliances.

Chromium ermöglicht den direkten Zugriff auf solche HTTP- beziehungsweise HTTPS-basierten Verwaltungsoberflächen.

Darüber hinaus können die integrierten Developer Tools verwendet werden, um den Webverkehr einer Anwendung aus Sicht des Browsers zu untersuchen.

## Einsatzgebiet im Rescue-System

Chromium eignet sich innerhalb von JUMBO BLOCK Rescue insbesondere für:

Zugriff auf öffentliche Webseiten,

Zugriff auf lokale Router- und Appliance-Oberflächen,

HTTP-/HTTPS-Funktionstests,

Prüfung webbasierter Administrationsoberflächen,

Untersuchung von Zertifikatsfehlern,

Analyse von HTTP-Statuscodes,

Analyse von Webrequests,

Kontrolle von Request- und Response-Headern,

Untersuchung von Cookies,

Analyse von Weiterleitungen,

Untersuchung von Ladeproblemen,

Abruf technischer Dokumentation,

Download benötigter Dateien

und ergänzende Diagnose zusammen mit Dig, MTR, Nmap, TCPDump und Wireshark.

## Browser und Netzwerkdiagnose

Ein Browser testet mehr als die reine IP-Erreichbarkeit.

Beim Aufruf einer HTTPS-Webseite können unter anderem folgende Komponenten beteiligt sein:

DNS,

IP-Routing,

TCP beziehungsweise andere unterstützte Webtransporte,

TLS,

Zertifikatsprüfung,

HTTP,

Redirects,

Cookies,

JavaScript,

Web APIs

und die eigentliche Webanwendung.

Dadurch kann Chromium Fehler sichtbar machen, die mit einem einfachen Ping oder Portscan nicht erkennbar wären.

## Typische Diagnosekette

Der Aufruf:

`https://example.com`

kann vereinfacht mehrere Schritte erfordern:

**1. Name auflösen**

DNS ermittelt eine Adresse.

**2. Ziel erreichen**

Das Netzwerk transportiert die Verbindung zum Ziel.

**3. Sichere Verbindung aufbauen**

Bei HTTPS wird eine verschlüsselte Verbindung aufgebaut und die Identität des Servers anhand der Zertifikatsmechanismen geprüft.

**4. HTTP-Kommunikation**

Browser und Server tauschen HTTP-Anfragen und Antworten aus.

**5. Ressourcen laden**

HTML kann weitere Ressourcen anfordern, beispielsweise:

CSS,

JavaScript,

Bilder,

Fonts,

API-Daten.

**6. Anwendung ausführen**

JavaScript und andere Browserfunktionen erzeugen die eigentliche Webanwendung.

Ein Fehler auf einer dieser Ebenen kann dazu führen, dass die Seite nicht oder nicht vollständig funktioniert.

## Direkte IP-Adresse

Lokale Geräte können häufig direkt über ihre IP-Adresse angesprochen werden.

Beispielprinzip:

`http://192.0.2.1`

oder:

`https://192.0.2.1`

Die hier verwendete Adresse dient ausschließlich als Dokumentationsbeispiel.

Bei einer direkten IP-Verbindung wird die normale DNS-Auflösung des Zielnamens umgangen.

### Bedeutung für die Diagnose

Funktioniert:

`https://IP-ADRESSE`

aber:

`https://hostname.example`

nicht,

ist dies ein wichtiger Hinweis.

Mögliche Untersuchungsrichtungen sind dann beispielsweise:

DNS,

Namensauflösung,

Proxykonfiguration,

Host-basierte Serverkonfiguration

oder TLS-Zertifikatszuordnung.

Die direkte IP-Verbindung beweist jedoch nicht automatisch, dass alle Funktionen der Webanwendung korrekt arbeiten.

### Hostname kann technisch erforderlich sein

Moderne Webserver können mehrere virtuelle Hosts auf derselben IP-Adresse betreiben.

Außerdem können TLS-Zertifikate für bestimmte DNS-Namen ausgestellt sein.

Eine Webseite, die über ihren korrekten Hostnamen funktioniert, muss daher nicht zwingend über die reine IP-Adresse identisch funktionieren.

### HTTP

HTTP überträgt Webinhalte ohne die durch HTTPS bereitgestellte Transportverschlüsselung und Serverauthentifizierung.

Bei unverschlüsselten Verbindungen können Daten auf dem Netzwerkpfad grundsätzlich beobachtet oder verändert werden.

Anmeldedaten und andere sensible Informationen sollten deshalb nicht unbedacht über eine ungesicherte HTTP-Verbindung übertragen werden.

## HTTPS

HTTPS verbindet HTTP mit einer verschlüsselten Transportverbindung.

Bei einer ordnungsgemäß authentifizierten HTTPS-Verbindung werden die übertragenen Daten zwischen Browser und Gegenstelle gegen einfaches Mitlesen und Verändern auf dem Transportweg geschützt.

HTTPS allein beweist jedoch nicht, dass der Inhalt einer Webseite vertrauenswürdig ist.

Auch eine schädliche Webseite kann grundsätzlich HTTPS verwenden.

## Zertifikate

Für die Authentifizierung einer HTTPS-Gegenstelle verwendet der Browser digitale Zertifikate.

Bei der Prüfung spielen unter anderem eine Rolle:

Zertifikatskette,

vertrauenswürdige Zertifizierungsstelle,

Gültigkeitszeitraum,

angeforderter Hostname

und weitere Sicherheitsanforderungen.

## Zertifikatsfehler

Chromium kann bei Problemen mit einer sicheren Verbindung Warnungen anzeigen.

Typische Fehlerklassen umfassen beispielsweise:

nicht vertrauenswürdige Zertifizierungsstelle,

Hostname passt nicht zum Zertifikat,

ungültiges beziehungsweise abgelaufenes Zertifikat,

oder andere TLS-/Zertifikatsprobleme.

Solche Warnungen dürfen in einer Rescue-Situation nicht reflexartig ignoriert werden.

## Lokale Appliances und selbstsignierte Zertifikate

Lokale Netzwerkgeräte verwenden teilweise:

selbstsignierte Zertifikate,

interne Zertifizierungsstellen,

veraltete Zertifikate

oder Zertifikate, deren Name nicht zur verwendeten IP-Adresse passt.

Eine Warnung kann deshalb bei einer lokalen Appliance einen technisch erklärbaren Grund besitzen.

Sie bleibt dennoch ein Sicherheitsindikator.

### Vor dem Umgehen einer Zertifikatswarnung

Vor einem bewussten Fortfahren sollte geprüft werden:

Ist dies wirklich das erwartete Gerät?

Stimmt die IP-Adresse?

Stimmt der Hostname?

Ist das Gerät über einen vertrauenswürdigen lokalen Pfad erreichbar?

Ist die Zertifikatsabweichung für dieses Gerät bekannt und dokumentiert?

Hat sich das Zertifikat unerwartet geändert?

Könnte eine falsche Gegenstelle oder ein Man-in-the-Middle-Szenario vorliegen?

### Systemzeit

Die Systemzeit ist bei Zertifikatsprüfungen relevant.

Ist Datum oder Uhrzeit des Rescue-Systems erheblich falsch, können eigentlich gültige Zertifikate als noch nicht gültig oder bereits abgelaufen erscheinen.

Bei unerwarteten Zertifikatsfehlern gehört die Kontrolle der Systemzeit daher zu den sinnvollen Diagnosemaßnahmen.

### Sicherheitsanzeige

Chromium stellt Informationen zum Sicherheitszustand einer Verbindung in der Benutzeroberfläche bereit.

Eine sichere Verbindung bedeutet dabei primär, dass die Verbindung zum angegebenen Ziel entsprechend geschützt wurde.

Sie ist kein Gütesiegel für:

den Inhalt,

den Betreiber,

heruntergeladene Dateien

oder geschäftliche Aussagen der Webseite.

### Weiterleitungen

Webserver können den Browser auf eine andere URL weiterleiten.

Typische HTTP-Statuscodes für Redirects sind beispielsweise:

`301`

`302`

`303`

`307`

`308`.

Weiterleitungen können legitim sein, beispielsweise:

HTTP → HTTPS,

alter Hostname → neuer Hostname,

Loginseite → Anwendung.

Sie können bei fehlerhafter Konfiguration aber auch zu:

Redirect-Schleifen,

falschen Zielseiten

oder unerwarteten Hosts

führen.

## HTTP-Statuscodes

Chromium beziehungsweise die Developer Tools können HTTP-Statuscodes sichtbar machen.

Wichtige Gruppen sind:

`1xx` – Information

`2xx` – erfolgreiche Verarbeitung

`3xx` – Weiterleitung

`4xx` – clientbezogene Fehler beziehungsweise Anforderungen

`5xx` – serverseitige Fehler.

## Beispiele

`200 OK`

zeigt eine erfolgreiche HTTP-Antwort.

`301` oder `302`

zeigen eine Weiterleitung.

`401`

weist auf eine erforderliche beziehungsweise fehlgeschlagene Authentifizierung hin.

`403`

zeigt, dass der Server die Anfrage ablehnt.

`404`

bedeutet, dass die angeforderte Ressource nicht gefunden wurde.

`500`

weist auf einen internen Serverfehler hin.

`502`

kann auf ein Problem zwischen Gateway/Proxy und Backend hinweisen.

`503`

zeigt, dass der Dienst aktuell nicht verfügbar ist.

Ein HTTP-Statuscode muss immer im Kontext der konkreten Anwendung interpretiert werden.

### Developer Tools

Chromium enthält integrierte Developer Tools – DevTools.

Diese Werkzeuge sind nicht ausschließlich für Webentwickler interessant.

Im Rescue-Einsatz können sie bei der Untersuchung webbasierter Systeme sehr hilfreich sein.

### DevTools öffnen

Unter Linux kann DevTools typischerweise mit:

`Ctrl` + `Shift` + `I`

geöffnet werden.

Alternativ kann die Funktion über das Browsermenü aufgerufen werden.

### Network Panel

Für die Rescue-Diagnose ist insbesondere das:

**Network Panel**

wichtig.

Solange die Developer Tools geöffnet sind und die Aufzeichnung aktiv ist, protokolliert das Network Panel die Netzwerkrequests der Seite.

### Was das Network Panel sichtbar macht

Zu einzelnen Requests können unter anderem untersucht werden:

URL,

HTTP-Methode,

Status,

Ressourcentyp,

Initiator,

übertragene Datenmenge,

Zeitverhalten,

Request Headers,

Response Headers,

Request Payload,

Response,

Cookies

und Timing.

### Diagnosebeispiel

Eine Appliance-Oberfläche wird angezeigt, aber bestimmte Informationen fehlen.

Im Network Panel kann sichtbar werden:

HTML-Seite: `200`

JavaScript: `200`

API-Aufruf: `500`

Damit wird erkennbar:

Die Weboberfläche selbst wurde geladen.

Das eigentliche Problem liegt möglicherweise bei einer Backend-API.

### Request Headers

Request Headers zeigen Informationen, die Chromium an den Server sendet.

Dazu können beispielsweise gehören:

Hostinformationen,

Accept-Header,

Content-Type,

Cookies,

Origin,

Referer

und weitere HTTP-Header.

Nicht jeder Request enthält dieselben Header.

### Response Headers

Response Headers stammen vom Server.

Sie können beispielsweise Informationen enthalten über:

Content-Type,

Caching,

Weiterleitungen,

Cookies,

Sicherheitsrichtlinien

und weitere HTTP-Eigenschaften.

### Payload

Bei Requests, die Daten übertragen, kann DevTools die Request-Nutzlast anzeigen.

Dies ist diagnostisch hilfreich, kann aber gleichzeitig sensible Daten offenlegen.

Dazu können gehören:

Formulardaten,

API-Parameter,

Benutzernamen,

Tokens

oder andere Anwendungsdaten.

### Response

Der Bereich:

**Response**

zeigt den vom Server zurückgelieferten Inhalt einer ausgewählten Anfrage, soweit DevTools ihn darstellen kann.

Dies ist beispielsweise bei API-Fehlern hilfreich.

Eine grafische Oberfläche kann lediglich:

„Fehler“

anzeigen, während die zugrunde liegende Serverantwort detailliertere technische Informationen enthält.

## Initiator

DevTools kann anzeigen, wodurch ein Request ausgelöst wurde.

Dies hilft beispielsweise festzustellen, ob eine Ressource:

durch das HTML-Dokument,

ein Script,

eine Weiterleitung

oder einen anderen Mechanismus

angefordert wurde.

## Timing

Der Timing-Bereich zerlegt die zeitlichen Bestandteile eines Requests.

Damit lässt sich genauer untersuchen, wo Zeit innerhalb eines Webrequests verbraucht wird.

Ein langsam wirkender Webdienst muss nicht zwingend durch reine Netzwerklatenz langsam sein.

Auch Serververarbeitung oder andere Bestandteile können erheblichen Anteil haben.

## Preserve Log

Mit:

**Preserve log**

kann das Network Panel Requests über Seitenwechsel beziehungsweise Reloads hinweg erhalten.

Dies ist besonders nützlich bei:

Loginvorgängen,

Redirect-Ketten

und Fehlern, die unmittelbar nach einer Navigation auftreten.

## Cache deaktivieren

Während DevTools geöffnet ist, kann für Diagnosezwecke der Browsercache deaktiviert werden.

Damit lässt sich prüfen, ob ein Problem durch zwischengespeicherte Ressourcen beeinflusst wird.

## Hard Reload

Für Webdiagnosen kann eine Seite ohne die übliche Cache-Nutzung neu geladen werden.

Dies ist hilfreich, wenn beispielsweise:

alte JavaScript-Dateien,

alte CSS-Dateien

oder andere gecachte Ressourcen

als Fehlerursache vermutet werden.

### Filter

Das Network Panel kann Requests filtern.

Damit lassen sich beispielsweise gezielt betrachten:

Fetch/XHR,

JavaScript,

CSS,

Bilder,

Dokumente,

WebSockets

oder bestimmte Domains beziehungsweise Request-Eigenschaften.

### Fetch/XHR

Moderne Webanwendungen laden viele Daten nach dem initialen Seitenaufbau über:

Fetch

oder:

XMLHttpRequest.

Eine Seite kann deshalb optisch geladen sein, obwohl die eigentliche Datenkommunikation fehlschlägt.

Für Appliance-Diagnosen sind Fetch-/XHR-Requests daher häufig besonders interessant.

### WebSocket

Webanwendungen können dauerhafte bidirektionale WebSocket-Verbindungen verwenden.

DevTools kann entsprechende Verbindungen und Nachrichten untersuchen.

Dies kann bei:

Live-Monitoring,

Statusanzeigen,

Dashboards

und administrativen Webanwendungen

hilfreich sein.

## HAR

Netzwerkrequests können aus DevTools als:

**HAR – HTTP Archive**

exportiert werden.

Eine HAR-Datei kann zahlreiche Details über Webrequests enthalten.

## HAR-Dateien können hochsensibel sein

Je nach Anwendung können HAR-Daten enthalten:

URLs,

Query-Parameter,

Header,

Cookies,

Request-Inhalte,

Antwortinformationen

und andere Sitzungsdaten.

Eine HAR-Datei darf deshalb nicht unkontrolliert weitergegeben werden.

Vor einer Weitergabe muss geprüft werden, ob sie:

Authentifizierungsinformationen,

Session-IDs,

Tokens,

personenbezogene Daten

oder interne Infrastrukturinformationen

enthält.

## DevTools ist kein vollständiger Paketmitschnitt

Das Network Panel arbeitet auf Browser- beziehungsweise Webanwendungsebene.

Es ersetzt keinen vollständigen Netzwerk-Capture.

Für eine paketgenaue Analyse dienen innerhalb von JUMBO BLOCK Rescue:

TCPDump

oder:

Wireshark.

## Wichtige Trennung

**Chromium DevTools**

zeigt die Webkommunikation aus Sicht des Browsers.

**TCPDump/Wireshark**

zeigen den am Capture-Punkt sichtbaren Netzwerkverkehr.

Diese Perspektiven ergänzen sich.

## Inkognito-Modus

Chromium besitzt einen Inkognito-Modus.

Inkognito ist ein eigener Browserfenster-Modus.

Die darin besuchten Seiten werden nicht in der normalen Browserhistorie gespeichert.

Die Sitzung verwendet einen temporären Cookie-/Site-Data-Kontext.

Nach Ende der Inkognito-Sitzung werden diese temporären Sitzungsdaten verworfen.

## Sinn im Rescue-System

Inkognito kann nützlich sein, um eine Weboberfläche ohne den normalen bestehenden Browserzustand zu testen.

Dies kann helfen, Einflüsse von:

bestehenden Cookies,

Anmeldesitzungen

und bestimmten gespeicherten Seitendaten

zu reduzieren.

## Inkognito bedeutet nicht Anonymität

Der Inkognito-Modus macht den Benutzer im Netzwerk nicht unsichtbar.

Webseiten können weiterhin Netzwerkkommunikation sehen.

Auch beispielsweise:

Netzwerkadministratoren,

Internetprovider,

Firewalls,

Proxies

und andere Infrastruktur auf dem Netzwerkpfad

können abhängig von ihrer Position und technischen Möglichkeiten Netzwerkaktivität erkennen.

### Downloads im Inkognito-Modus

Heruntergeladene Dateien werden durch das Schließen der Inkognito-Sitzung nicht automatisch gelöscht.

Sie bleiben im Dateisystem vorhanden, bis sie ausdrücklich entfernt werden.

Das ist für ein Rescue-System besonders wichtig.

### Lesezeichen im Inkognito-Modus

Auch ausdrücklich angelegte Lesezeichen werden durch das Beenden einer Inkognito-Sitzung nicht automatisch entfernt.

### Inkognito und forensische Anforderungen

Inkognito ist kein forensischer Spurenvernichtungsmodus.

Es darf nicht als Garantie interpretiert werden, dass auf dem Gesamtsystem oder außerhalb des Browsers keinerlei Spuren einer Aktivität existieren.

### Downloads

Chromium kann Dateien aus dem Internet oder von lokalen Websystemen herunterladen.

Dies ist im Rescue-Betrieb beispielsweise nützlich für:

Dokumentationen,

Konfigurationsdateien,

Logs,

Firmwaredateien,

Treiber,

Prüfsummen

oder andere benötigte Ressourcen.

### Download bedeutet nicht Vertrauenswürdigkeit

Eine erfolgreich über HTTPS heruntergeladene Datei ist nicht automatisch sicher.

HTTPS schützt die Übertragung zur authentifizierten Gegenstelle.

Es beweist nicht, dass die Datei:

schadcodefrei,

unverändert gegenüber einer Herstellerreferenz,

für das Zielsystem geeignet

oder tatsächlich vom erwarteten Hersteller erstellt

wurde.

### Downloads verifizieren

Bei sicherheitsrelevanten Downloads sollten – sofern vom Hersteller bereitgestellt – zusätzlich geprüft werden:

Herkunft,

Dateiname,

Version,

kryptografische Prüfsumme

und/oder:

digitale Signatur.

Innerhalb von JUMBO BLOCK Rescue können hierfür je nach Dateityp andere Werkzeuge ergänzend eingesetzt werden.

### Download aus unbekannter Quelle

Dateien unbekannter Herkunft sollten nicht unmittelbar ausgeführt oder geöffnet werden.

Das gilt insbesondere für:

Programme,

Scripts,

Archive,

Office-Dokumente,

PDF-Dateien

und andere aktive beziehungsweise komplexe Dateiformate.

### ClamAV als Ergänzung

Eine heruntergeladene Datei kann vor weiterer Verwendung mit ClamAV untersucht werden.

Dies erhöht die diagnostische Sicherheit, ersetzt aber keine:

Herkunftsprüfung,

Signaturprüfung

oder manuelle Bewertung.

Ein fehlender ClamAV-Fund beweist nicht, dass eine Datei ungefährlich ist.

### GnuPG als Ergänzung

Stellt ein Hersteller eine OpenPGP-Signatur bereit, kann später im Verschlüsselungsblock dokumentiertes GnuPG zur Signaturprüfung eingesetzt werden.

Damit lässt sich – bei korrekt geprüfter Schlüsselkette – die Authentizität beziehungsweise Integrität deutlich stärker bewerten als allein anhand eines Browserdownloads.

### Sandbox

Chromium verwendet eine Mehrprozessarchitektur und Sandboxmechanismen.

Webinhalte werden nicht einfach mit uneingeschränkten Systemrechten in einem einzigen Browserprozess ausgeführt.

Rendererprozesse werden eingeschränkt, um die Auswirkungen einer kompromittierten Rendering-Komponente zu begrenzen.

### Sandbox ist keine absolute Sicherheitsgrenze

Sandboxing reduziert Risiken.

Es bedeutet nicht:

„Eine beliebige schädliche Webseite kann gefahrlos geöffnet werden.“

Browser und Sandbox können Sicherheitslücken besitzen.

Der sichere Grundsatz bleibt deshalb:

Unbekannte oder offensichtlich verdächtige Inhalte nur öffnen, wenn dies für die Diagnose notwendig und vertretbar ist.

### `chromium-sandbox`

Die Baseline enthält ausdrücklich das Paket:

`chromium-sandbox 154.0.8037.57-1~deb13u1`.

Damit ist die zu dieser Chromium-Paketversion gehörende Sandbox-Komponente als installiert belegt.

Aus der Paketliste allein wird jedoch nicht abgeleitet, welcher konkrete Sandboxmechanismus für jeden einzelnen Chromium-Prozess zur Laufzeit aktiv ist.

### Site Isolation

Chromium verwendet auf Desktop-Systemen Site Isolation als zusätzliche Sicherheitsarchitektur.

Webseiten verschiedener Sites werden in getrennten Rendererprozessen isoliert.

Damit soll insbesondere verhindert beziehungsweise erschwert werden, dass eine kompromittierte Webseite Daten anderer Sites aus einem gemeinsamen Rendererprozess auslesen kann.

### Defense in Depth

Sandbox und Site Isolation sind Beispiele für:

**Defense in Depth.**

Die Sicherheitsarchitektur verlässt sich nicht ausschließlich darauf, dass Webinhalte fehlerfrei verarbeitet werden.

Zusätzliche technische Grenzen sollen die Auswirkungen eines Fehlers reduzieren.

### Kein Grund, Sicherheitsmechanismen abzuschalten

Für einen Rescue-Browser wäre es sicherheitstechnisch besonders problematisch, Schutzmechanismen wie die Sandbox ohne zwingenden technischen Grund zu deaktivieren.

Ein Start mit Optionen, die Browser-Sicherheitsgrenzen abschalten, sollte daher nicht Bestandteil eines normalen Rescue-Workflows sein.

### Lokale Geräte mit alten Weboberflächen

Ältere Appliances können Browserkompatibilitätsprobleme verursachen.

Beispiele sind:

veraltete TLS-Konfigurationen,

nicht mehr unterstützte Kryptografie,

alte JavaScript-Implementierungen,

unsichere Zertifikate

oder proprietäre Browseranforderungen.

### Sicherheitsgrenzen nicht für Kompatibilität opfern

Wenn eine alte Appliance mit aktuellem Chromium nicht mehr funktioniert, bedeutet dies nicht automatisch, dass Chromium fehlerhaft ist.

Der Grund kann eine veraltete Sicherheitskonfiguration des Geräts sein.

Ein Rescue-System sollte nicht leichtfertig seine Browser-Sicherheitsmechanismen reduzieren, nur um eine technisch überholte Weboberfläche erreichbar zu machen.

### Captive Portals

Bei WLAN- oder anderen Netzwerken kann vor dem eigentlichen Internetzugriff ein Captive Portal erforderlich sein.

Ein Browser ist für diese Anmeldung häufig notwendig.

Ein Zertifikats- oder Redirectproblem direkt nach einer Netzwerkverbindung kann deshalb auch mit einem Captive Portal zusammenhängen.

### Proxy

Webzugriffe können über einen Proxy geleitet werden.

Ein Browserproblem kann deshalb entstehen, obwohl:

DNS,

IP-Routing

und direkter Netzwerkzugriff

grundsätzlich funktionieren.

Bei unerwarteten Browserfehlern gehört eine mögliche Proxykonfiguration daher zur Diagnose.

### Cookies

Cookies speichern kleine Datenobjekte im Browserkontext.

Sie werden häufig für:

Sitzungen,

Anmeldestatus,

Präferenzen

und Webanwendungsfunktionen

verwendet.

Fehlerhafte oder veraltete Cookies können eine Webanwendung beeinflussen.

### Cookies können sicherheitsrelevant sein

Session-Cookies können Authentifizierungsinformationen repräsentieren.

Sie dürfen nicht unkontrolliert:

exportiert,

protokolliert,

weitergegeben

oder in Screenshots beziehungsweise HAR-Dateien veröffentlicht

werden.

### Local Storage und andere Site-Daten

Moderne Webanwendungen verwenden neben Cookies weitere lokale Browsermechanismen.

Dadurch kann ein Fehler auch von gespeichertem Browserzustand abhängen.

Ein Test in einer frischen Inkognito-Sitzung kann helfen festzustellen, ob der bestehende Browserzustand Einfluss hat.

### Cache

Chromium speichert Webressourcen zwischen, um wiederholte Zugriffe zu beschleunigen.

Bei administrativen Weboberflächen kann ein veralteter Cache jedoch gelegentlich zu einer Mischung aus alten und neuen Webressourcen führen.

DevTools erlaubt deshalb gezielte Tests ohne Cache.

### JavaScript

Viele moderne Verwaltungsoberflächen sind stark von JavaScript abhängig.

Wenn JavaScript-Code nicht geladen wird oder während der Ausführung fehlschlägt, kann eine Seite:

leer bleiben,

nur teilweise erscheinen,

keine Daten anzeigen

oder Bedienelemente nicht ausführen.

### Console

Die DevTools-Konsole kann JavaScript-Fehler und andere Meldungen der Webseite anzeigen.

Sie ist deshalb eine wichtige Ergänzung zum Network Panel.

### Vorsicht mit Console-Befehlen

Code aus unbekannten Webseiten, Foren oder Nachrichten sollte nicht unkritisch in die DevTools-Konsole kopiert werden.

JavaScript, das dort manuell ausgeführt wird, arbeitet im Kontext der betreffenden Webseite und kann sicherheitsrelevante Aktionen auslösen.

### Passwort- und Zugangsdaten

Auf einem Rescue-System sollte bewusst entschieden werden, ob Zugangsdaten im Browser gespeichert werden.

Bei einem temporären oder gemeinsam verwendeten Rescue-System ist dauerhafte Speicherung von:

Passwörtern,

Tokens,

Formulardaten

oder Sitzungsinformationen

besonders kritisch.

## Rescue-System als temporäre Arbeitsumgebung

Wird JUMBO BLOCK Rescue als Live-/Rescue-System eingesetzt, sollte nicht automatisch davon ausgegangen werden, dass jeder Browserzustand nach dem Neustart verschwindet.

Dies hängt von:

Bootmedium,

Persistenz,

Mounts,

Downloadzielen

und konkreter Systemkonfiguration

ab.

Aus der Chromium-Paketbaseline allein lässt sich keine Aussage über das Persistenzverhalten von JUMBO BLOCK Rescue ableiten.

## Typischer Rescue-Ablauf: lokale Routeroberfläche nicht erreichbar

**1. Lokale Netzkonfiguration prüfen**

Ist das Rescue-System im richtigen Netz?

**2. ARP Scan einsetzen**

Ist das Gerät auf dem lokalen IPv4-Layer-2-Netz sichtbar?

**3. Nmap einsetzen**

Ist der erwartete HTTP-/HTTPS-Port erreichbar?

**4. Chromium öffnen**

Direkte Geräteoberfläche aufrufen.

**5. Browserfehler genau dokumentieren**

DNS?

Timeout?

Connection refused?

TLS?

Zertifikat?

HTTP-Fehler?

**6. DevTools Network öffnen**

HTTP-Kommunikation untersuchen.

**7. Bei Bedarf TCPDump oder Wireshark ergänzen**

Paketebene untersuchen.

### Typischer Rescue-Ablauf: Hostname funktioniert nicht

**1. Chromium-Fehler dokumentieren**

**2. Dig verwenden**

A-/AAAA-Auflösung prüfen.

**3. Gegebenenfalls direkte IP-Verbindung testen**

**4. Zertifikats- und Hostname-Unterschied berücksichtigen**

**5. MTR für den Netzwerkpfad einsetzen**

**6. Nmap für den Zielport einsetzen**

Damit wird die Browsermeldung in einzelne Diagnoseebenen zerlegt.

### Typischer Rescue-Ablauf: Webseite lädt teilweise

**1. DevTools öffnen**

**2. Network Panel aktivieren**

**3. Seite neu laden**

**4. Fehlgeschlagene Requests identifizieren**

Besonders auf:

4xx,

5xx,

CORS-Fehler,

fehlgeschlagene Requests

und ungewöhnlich lange Requests

achten.

**5. Request auswählen**

Headers, Response und Timing untersuchen.

**6. Console prüfen**

JavaScript- beziehungsweise Webanwendungsfehler kontrollieren.

**7. Bei Bedarf ohne Cache wiederholen**

### Typischer Rescue-Ablauf: HTTPS-Warnung bei lokaler Appliance

**1. IP-Adresse und Hostname kontrollieren**

**2. Sicherstellen, dass das erwartete Gerät angesprochen wird**

**3. Systemzeit prüfen**

**4. Zertifikatsfehler genau lesen**

**5. Zertifikatsdaten untersuchen**

**6. Mit bekannter Gerätedokumentation vergleichen**

**7. Nur nach bewusster Risikobewertung fortfahren**

Eine Zertifikatswarnung ist kein lästiger Dialog, sondern ein Hinweis darauf, dass die erwartete Authentifizierung der Gegenstelle nicht vollständig bestätigt werden konnte.

## Typischer Rescue-Ablauf: Download einer Reparaturdatei

**1. Offizielle Quelle bestimmen**

**2. HTTPS-Verbindung und Zielhostname prüfen**

**3. Datei herunterladen**

**4. Hersteller-Prüfsumme beziehungsweise Signatur beschaffen**

**5. Datei kryptografisch verifizieren**

**6. Gegebenenfalls zusätzlich mit ClamAV untersuchen**

**7. Erst danach verwenden**

## Chromium und Dig

Dig untersucht DNS direkt.

Chromium verwendet Namensauflösung als Teil des Webzugriffs.

Wenn Chromium einen Hostnamen nicht erreichen kann, hilft Dig bei der Trennung:

**DNS-Problem**

oder:

**Browser-/Webproblem.**

## Chromium und MTR

MTR untersucht den Netzwerkpfad.

Chromium untersucht die eigentliche Webverbindung.

Eine langsame Webseite kann durch:

Netzwerklatenz,

Paketverlust,

Serververarbeitung,

viele Webrequests,

JavaScript

oder andere Faktoren

verursacht werden.

MTR und Chromium DevTools liefern deshalb unterschiedliche Perspektiven.

### Chromium und Nmap

Nmap kann feststellen, ob beispielsweise TCP-Port 80 oder 443 an einem autorisierten Ziel erreichbar erscheint.

Chromium prüft anschließend, ob dort tatsächlich eine für den Browser nutzbare Webanwendung bereitsteht.

Vereinfacht:

**Nmap → Ist der Port/Dienst netzwerkseitig sichtbar?**

**Chromium → Funktioniert die Webanwendung über diesen Dienst?**

### Chromium und TCPDump

TCPDump kann die durch Chromium erzeugte Netzwerkkommunikation terminalbasiert aufzeichnen.

Dies ist hilfreich, wenn Browsermeldungen allein nicht erklären, was auf dem Netzwerk passiert.

### Chromium und Wireshark

Wireshark bietet die detaillierteste grafische Paketperspektive.

Chromium DevTools kennt dagegen die Webanwendungsstruktur und kann beispielsweise:

URLs,

HTTP-Requests,

Header,

Statuscodes,

Initiatoren

und Webanwendungs-Timing

direkt zuordnen.

Beide Werkzeuge ergänzen sich.

### Chromium und Whois

Eine über Chromium aufgerufene öffentliche Domain kann mit Dig aufgelöst werden.

Die resultierende öffentliche IP-Adresse kann anschließend mit Whois registryseitig eingeordnet werden.

Damit ergibt sich:

**Chromium → Webanwendung**

**Dig → DNS**

**Whois → Registryzuordnung**

## Grenzen

Chromium ist kein Paket-Sniffer.

Chromium ersetzt Wireshark oder TCPDump nicht.

Chromium ist kein Portscanner.

Chromium ersetzt Nmap nicht.

Chromium ist kein DNS-Diagnosewerkzeug auf Protokollebene.

Chromium ersetzt Dig nicht.

Eine funktionierende Webseite beweist nicht, dass alle Netzwerkdienste des Systems funktionieren.

Eine HTTPS-Verbindung beweist nicht, dass der Inhalt einer Webseite vertrauenswürdig ist.

Eine Zertifikatswarnung darf nicht automatisch ignoriert werden.

Inkognito macht den Benutzer nicht anonym.

Inkognito löscht heruntergeladene Dateien nicht automatisch.

Die Sandbox reduziert Risiken, macht das Öffnen unbekannter Inhalte aber nicht risikofrei.

DevTools zeigt die Browserperspektive und keinen vollständigen Paketmitschnitt.

## Sicherheitshinweise

Chromium in einer Rescue-Umgebung grundsätzlich als aktiven Netzwerkclient betrachten.

Unbekannte und kompromittierungsverdächtige Webseiten nur öffnen, wenn dies für die Diagnose erforderlich ist.

Zertifikatswarnungen nicht routinemäßig umgehen.

Vor der Eingabe von Zugangsdaten Hostname und Verbindungszustand prüfen.

Downloads unbekannter Herkunft nicht unmittelbar ausführen.

Herstellerdateien nach Möglichkeit über kryptografische Prüfsummen oder digitale Signaturen verifizieren.

Sensible Downloads gegebenenfalls zusätzlich mit ClamAV untersuchen.

Keine unbekannten Scripts in die DevTools-Konsole kopieren.

HAR-Dateien vor Weitergabe auf Zugangsdaten, Cookies und Tokens prüfen.

Session-Cookies und Authentifizierungsinformationen vertraulich behandeln.

Bei temporären Rescue-Systemen nicht ohne Prüfung annehmen, dass alle Browserdaten beim Neustart verschwinden.

Sandbox- und andere Browser-Sicherheitsmechanismen nicht leichtfertig deaktivieren.

## Technische Einordnung in JUMBO BLOCK Rescue

Chromium übernimmt innerhalb des Netzwerkbereichs von JUMBO BLOCK Rescue die Rolle des **grafischen Webclients und der Webanwendungsdiagnose**.

Damit ist Block A.17 bis A.24 funktional vollständig gegliedert:

**A.17 Nmap**

aktive Host-, Port- und Dienstanalyse.

**A.18 Wireshark**

grafische Paket- und Protokollanalyse.

**A.19 TCPDump**

terminalbasierte Paketaufzeichnung und Filterung.

**A.20 ARP Scan**

lokale IPv4-Geräteerkennung auf Layer 2.

**A.21 MTR**

fortlaufende Pfad-, Latenz- und Paketverlustanalyse.

**A.22 Dig**

DNS-, Resolver-, Resource-Record- und Delegationsanalyse.

**A.23 Whois**

Registry-, Netzblock-, Domain- und AS-Einordnung.

**A.24 Chromium**

grafischer Webzugriff sowie HTTP-/HTTPS- und Webanwendungsdiagnose.

Die Werkzeuge bilden damit eine Diagnosekette über mehrere Ebenen:

**ARP Scan**

lokales Gerät erkennen.

↓

**Dig**

Namensauflösung untersuchen.

↓

**MTR**

Netzwerkpfad analysieren.

↓

**Nmap**

Zielports und Dienste untersuchen.

↓

**Chromium**

Webdienst und Benutzeroberfläche prüfen.

↓

**DevTools**

HTTP-/Webanwendungskommunikation analysieren.

↓

**TCPDump / Wireshark**

bei Bedarf auf die Paketebene wechseln.

↓

**Whois**

öffentliche Netzwerkobjekte administrativ einordnen.

Chromium nimmt dabei eine besondere Stellung ein:

Es ist nicht nur ein Werkzeug für den Zugriff auf das Internet.

In einem modernen Rescue-System ist der Browser zugleich die **Bedien- und Diagnoseoberfläche für eine große Zahl webbasierter Infrastrukturkomponenten**.

Seine wichtigste diagnostische Stärke liegt deshalb in der Verbindung von:

**realem Webzugriff,**

**HTTP-/HTTPS-Verhalten,**

**TLS-/Zertifikatsprüfung**

und:

**integrierter Request-Analyse über DevTools.**

Für den Rescue-Einsatz gilt dabei als zentrale Sicherheitsregel:

**Eine Browserwarnung sollte nicht beseitigt werden, bevor verstanden wurde, warum sie erscheint.**

# A.25 Cryptsetup

**Bereich:** Verschlüsselung
**Programm:** Cryptsetup
**Baseline 20261001:** Cryptsetup 2.7.5
**Installierte Pakete:** `cryptsetup 2:2.7.5-2`, `cryptsetup-bin 2:2.7.5-2`, `cryptsetup-initramfs 2:2.7.5-2`
**Bedienung:** Kommandozeile
**Aufgabe:** Erkennung, Analyse, Öffnung und Verwaltung verschlüsselter Datenträger, insbesondere LUKS-Volumes

## Zweck

Cryptsetup ist das zentrale Linux-Kommandozeilenwerkzeug zur Verwaltung verschlüsselter Blockgeräte auf Basis des Kernel-Subsystems:

**dm-crypt**

und des darauf aufbauenden Formats:

**LUKS – Linux Unified Key Setup.**

Cryptsetup kann verschlüsselte Datenträger beziehungsweise Partitionen:

erkennen,

Metadaten untersuchen,

entsperren,

als Device-Mapper-Gerät bereitstellen,

wieder schließen

und – bei administrativer Verwendung – Verschlüsselungsheader, Schlüssel und Keyslots verwalten.

Für JUMBO BLOCK Rescue steht primär die sichere **Analyse und kontrollierte Öffnung bereits vorhandener verschlüsselter Datenträger** im Vordergrund.

## Einsatzgebiet im Rescue-System

Cryptsetup eignet sich innerhalb von JUMBO BLOCK Rescue insbesondere für:

Erkennung von LUKS-Volumes,

Untersuchung von LUKS-Metadaten,

Ermittlung von LUKS-Version und UUID,

Anzeige vorhandener Keyslots,

kontrolliertes Entsperren verschlüsselter Partitionen,

Bereitstellung entschlüsselter Blockgeräte über Device Mapper,

Read-only-orientierte Datenrettungsabläufe,

Überprüfung aktiver Cryptsetup-Mappings,

Schließen entschlüsselter Mappings,

Sicherung eines LUKS-Headers

und – nur bei ausdrücklich erforderlichen administrativen Arbeiten – Verwaltung von Passphrasen und Keyslots.

## Cryptsetup und dm-crypt

Cryptsetup führt die eigentliche Blockverschlüsselung nicht allein im Benutzerprozess durch.

Es konfiguriert unter Linux unter anderem das Kernel-Subsystem:

`dm-crypt`.

Ein erfolgreich geöffnetes verschlüsseltes Volume erscheint typischerweise als Device-Mapper-Gerät unter:

`/dev/mapper/`

Dieses entschlüsselte Blockgerät kann anschließend wie ein normales Blockgerät weiter untersucht werden.

## Wichtige Ebenentrennung

Bei einem LUKS-Datenträger müssen mehrere Ebenen unterschieden werden:

**Physisches beziehungsweise ursprüngliches Blockgerät**

zum Beispiel eine Partition.

↓

**LUKS-Verschlüsselungsschicht**

Cryptsetup entschlüsselt die Daten blockweise.

↓

**Device-Mapper-Mapping**

zum Beispiel:

`/dev/mapper/rescue`

↓

**Dateisystem**

zum Beispiel ext4, XFS oder ein anderes unterstütztes Dateisystem.

↓

**Mountpunkt**

erst hier werden Dateien und Verzeichnisse sichtbar.

Diese Ebenen dürfen in der Rescue-Diagnose nicht miteinander verwechselt werden.

## LUKS

LUKS ist ein standardisiertes Linux-Format für Blockgeräteverschlüsselung.

Im Gegensatz zu einfachem Plain-dm-crypt enthält ein LUKS-Datenträger Metadaten zur Verwaltung der Verschlüsselung.

Dazu gehören unter anderem Informationen über:

Formatversion,

Verschlüsselungsparameter,

UUID,

Keyslots

und die für die Schlüsselableitung erforderlichen Parameter.

## LUKS1 und LUKS2

Cryptsetup 2.7.5 unterstützt sowohl:

**LUKS1**

als auch:

**LUKS2.**

LUKS2 besitzt ein erweitertes Metadatenformat und zusätzliche Verwaltungsfunktionen.

Bei Rescue-Arbeiten ist deshalb zunächst festzustellen, welcher LUKS-Typ tatsächlich vorliegt.

## LUKS erkennen

Eine besonders wichtige nicht destruktive Prüfung lautet:

```
cryptsetup isLuks DEVICE
```

Der Befehl prüft, ob Cryptsetup auf dem angegebenen Gerät einen gültigen LUKS-Header erkennt.

## Ausführliche Prüfung

Mit:

```
cryptsetup -v isLuks DEVICE
```

wird zusätzlich eine textuelle Ausgabe erzeugt.

Ohne `-v` wird das Ergebnis im Wesentlichen über den Exitstatus signalisiert.

### Bedeutung von `isLuks`

Ein erfolgreiches:

`isLuks`

bedeutet:

Auf dem untersuchten Gerät wurde ein von Cryptsetup als LUKS erkannter Header gefunden.

Es bedeutet nicht:

dass eine bekannte Passphrase vorhanden ist,

dass das Dateisystem innerhalb des Containers intakt ist,

oder:

dass sämtliche verschlüsselten Nutzdaten fehlerfrei lesbar sind.

### Keine LUKS-Erkennung

Schlägt `isLuks` fehl, kann dies verschiedene Ursachen haben.

Beispielsweise:

Gerät ist nicht LUKS-formatiert,

falsches Blockgerät untersucht,

LUKS-Header beschädigt,

LUKS-Header liegt separat vor,

oder das Gerät verwendet ein anderes Verschlüsselungsformat.

Aus einem negativen `isLuks`-Ergebnis darf daher bei einem beschädigten Datenträger nicht automatisch geschlossen werden, dass niemals LUKS vorhanden war.

### Header untersuchen

Für einen erkannten LUKS-Datenträger ist:

`cryptsetup luksDump DEVICE`

eines der wichtigsten Diagnosekommandos.

`luksDump` zeigt die LUKS-Headerinformationen an.

### `luksDump`

Abhängig von LUKS-Version und Konfiguration können dabei unter anderem Informationen sichtbar werden über:

LUKS-Version,

UUID,

Verschlüsselungsparameter,

Keyslots,

Key-Derivation-Parameter

und weitere Metadaten.

### `luksDump` entschlüsselt nicht die Nutzdaten

Der Befehl:

```
cryptsetup luksDump DEVICE
```

öffnet nicht automatisch das Dateisystem und entschlüsselt nicht den gesamten Datenträger.

Er untersucht die im LUKS-Header vorhandenen Metadaten.

Damit eignet er sich besonders gut für eine erste Bestandsaufnahme.

### UUID

Mit:

```
cryptsetup luksUUID DEVICE
```

kann die UUID eines LUKS-Geräts ausgegeben werden.

Diese UUID identifiziert das LUKS-Volume.

### UUID nicht mit Dateisystem-UUID verwechseln

Ein entschlüsseltes Dateisystem kann seinerseits eine eigene UUID besitzen.

Deshalb muss unterschieden werden:

**LUKS UUID**

und:

**Dateisystem-UUID innerhalb des entschlüsselten Volumes.**

Sie gehören zu unterschiedlichen Ebenen.

### Keyslots

LUKS kann mehrere Schlüsselzugänge beziehungsweise Keyslots verwalten.

Verschiedene Passphrasen können dadurch Zugriff auf denselben zugrunde liegenden Volume Key ermöglichen.

### Passphrase ist nicht direkt der Datenverschlüsselungsschlüssel

Bei LUKS wird eine Benutzerpassphrase nicht einfach unmittelbar als Schlüssel für jeden Datenblock verwendet.

Die Passphrase dient dazu, über die LUKS-Schlüsselverwaltung Zugriff auf den eigentlichen Volume Key zu erhalten.

Die Datenverschlüsselung erfolgt mit diesem Volume Key.

### Konsequenz

Mehrere unterschiedliche Passphrasen können dasselbe verschlüsselte Volume öffnen.

Das Ändern einer Passphrase bedeutet daher nicht, dass sämtliche Nutzdaten mit einem neuen Volume Key vollständig neu verschlüsselt werden.

### LUKS-Volume öffnen

Ein LUKS-Gerät kann mit:

```
cryptsetup open DEVICE NAME
```

geöffnet werden.

Für LUKS kann der Typ ausdrücklich angegeben werden:

```
cryptsetup open --type luks DEVICE NAME
```

Die traditionelle LUKS-Syntax lautet:

```
cryptsetup luksOpen DEVICE NAME
```

Nach erfolgreicher Authentifizierung erzeugt Cryptsetup ein Mapping mit dem angegebenen Namen.

### Beispielprinzip

```
sudo cryptsetup open --type luks /dev/sdXN rescue
```

Nach erfolgreicher Öffnung steht typischerweise ein Gerät zur Verfügung:

```
/dev/mapper/rescue
```

Die tatsächliche Gerätebezeichnung muss vor jedem Rescue-Eingriff eindeutig festgestellt werden.

### Mapping-Name

Der Name:

```
rescue
```

ist frei gewählt.

Er bezeichnet nicht die ursprüngliche Partition, sondern das neu erzeugte Device-Mapper-Mapping.

### Passphrase

Wird keine andere Schlüsselquelle angegeben, fordert Cryptsetup bei einem normalen interaktiven LUKS-Open typischerweise eine Passphrase an.

Die eingegebene Passphrase wird nicht als Klartext auf der normalen Terminalausgabe dargestellt.

### Falsche Passphrase

Kann mit einer eingegebenen Passphrase kein geeigneter Keyslot erfolgreich geöffnet werden, wird kein nutzbares Mapping erzeugt.

Eine falsche Passphrase beschädigt den LUKS-Datenträger nicht.

### Read-only öffnen

Für Rescue- und Analysearbeiten ist besonders:

`--readonly`

relevant.

Beispiel:

```
sudo cryptsetup open --readonly --type luks DEVICE rescue
```

Damit wird das erzeugte Mapping als read-only eingerichtet.

### Warum Read-only wichtig ist

Bei einem beschädigten oder forensisch sensiblen Datenträger sollte grundsätzlich vermieden werden, unnötige Schreibzugriffe auf das Original zu erzeugen.

Ein read-only geöffnetes Cryptsetup-Mapping reduziert dieses Risiko auf der Verschlüsselungsebene.

### Read-only allein genügt nicht immer für einen vollständigen Schutz

Auch die nachfolgenden Ebenen müssen berücksichtigt werden.

Wird beispielsweise ein Dateisystem später auf eine Weise eingebunden, die Schreibzugriffe voraussetzt, kann dies fehlschlagen oder unerwünschte Nebeneffekte erzeugen.

Für einen kontrollierten Rescue-Workflow sollten daher sowohl:

Cryptsetup-Mapping

als auch:

Dateisystemzugriff

bewusst read-only geplant werden.

### Geöffnetes Mapping prüfen

Mit:

```
cryptsetup status NAME
```

kann der Status eines aktiven Cryptsetup-Mappings untersucht werden.

Beispiel:

```
cryptsetup status rescue
```

Die Ausgabe kann Informationen über das aktive Mapping und dessen Verschlüsselungskonfiguration enthalten.

### Status eines nicht aktiven Mappings

Ist der angegebene Mapping-Name nicht aktiv, meldet Cryptsetup entsprechend, dass kein aktives Gerät dieses Namens vorhanden ist.

### Nach dem Öffnen

Nach erfolgreichem:

```
cryptsetup open
```

ist zunächst nur die Verschlüsselungsschicht geöffnet.

Das bedeutet noch nicht, dass das darin enthaltene Dateisystem automatisch eingehängt ist.

### Dateisystem untersuchen

Das entschlüsselte Mapping kann anschließend beispielsweise mit geeigneten Systemwerkzeugen auf:

Partitionierung,

Dateisystemtyp,

Dateisystemzustand

oder enthaltene Daten

untersucht werden.

Welches Werkzeug dafür geeignet ist, hängt von der tatsächlichen Struktur innerhalb des entschlüsselten Geräts ab.

### Verschlüsselung und Dateisystemschaden unterscheiden

Ein wichtiger Rescue-Fall lautet:

LUKS lässt sich erfolgreich öffnen,

aber das Dateisystem kann nicht gemountet werden.

Dann funktioniert die Entschlüsselung grundsätzlich.

Das Problem kann auf der darüberliegenden Ebene liegen:

Dateisystem beschädigt,

innere Partitionstabelle beschädigt,

RAID/LVM-Struktur vorhanden,

oder anderer Aufbau als erwartet.

## LUKS erfolgreich öffnen bedeutet nicht „Daten gesund“

Cryptsetup beantwortet primär:

**Kann die Verschlüsselungsschicht geöffnet werden?**

Es beantwortet nicht:

**Ist das Dateisystem fehlerfrei?**

und auch nicht:

**Sind alle Dateien unbeschädigt?**

## Mapping schließen

Ein geöffnetes Mapping wird mit:

```
cryptsetup close NAME
```

wieder geschlossen.

Beispiel:

```
sudo cryptsetup close rescue
```

Die ältere kompatible Aktionsbezeichnung:

```
luksClose
```

ist ebenfalls Teil der Cryptsetup-Befehlssyntax.

## Vor dem Schließen

Ein darüber eingehängtes Dateisystem muss zuerst ordnungsgemäß ausgehängt werden.

Die Reihenfolge lautet daher grundsätzlich:

**Dateisystem unmounten**

↓

**Cryptsetup-Mapping schließen**

↓

**Datenträger gegebenenfalls sicher entfernen**

### Warum die Reihenfolge wichtig ist

Solange ein Dateisystem auf:

`/dev/mapper/NAME`

eingehängt ist, wird das Mapping noch verwendet.

Ein erzwungenes oder unsauberes Entfernen von Ebenen kann zu Fehlern beziehungsweise Datenverlust führen.

### Rescue-Grundablauf

Für einen intakten verschlüsselten Datenträger kann ein konservativer Ablauf lauten:

**1. Datenträger eindeutig identifizieren**

**2. Hardwarezustand berücksichtigen**

**3. Bei physischem Schaden zuerst Image beziehungsweise Klon erstellen**

**4. LUKS-Erkennung durchführen**

```
cryptsetup -v isLuks DEVICE
```

**5. Headerinformationen untersuchen**

```
cryptsetup luksDump DEVICE
```

**6. Volume möglichst read-only öffnen**

```
cryptsetup open --readonly --type luks DEVICE rescue
```

**7. Mapping prüfen**

```
cryptsetup status rescue
```

**8. Dateisystemebene separat untersuchen**

**9. Daten auf ein anderes Medium sichern**

**10. Dateisystem aushängen**

**11. Mapping schließen**

```
cryptsetup close rescue
```

### Beschädigter Datenträger

Bei einem physisch instabilen Datenträger sollte Cryptsetup nicht das erste Rettungswerkzeug sein.

Wiederholte Leseversuche können einen sterbenden Datenträger zusätzlich belasten.

Hier gilt weiterhin die bereits im Datenrettungsblock dokumentierte Strategie:

**Original möglichst wenig belasten → mit GNU ddrescue ein Image oder einen Klon erstellen → anschließend auf der Kopie arbeiten.**

### Verschlüsseltes Image

Ein sektorweises Image einer LUKS-Partition enthält die verschlüsselten Sektoren einschließlich der für das LUKS-Format relevanten Strukturen.

Ein solches Image bleibt verschlüsselt.

### Vorteil

Damit kann die weitere Analyse auf einer Kopie erfolgen, während das Original geschützt bleibt.

Gerade bei:

defekten SSDs,

Festplatten mit Lesefehlern,

USB-Medien mit Aussetzern

oder anderen instabilen Datenträgern

ist dies ein zentraler Rescue-Grundsatz.

### Verschlüsselung verhindert kein Imaging

Ein Blockgerät muss nicht zuerst entschlüsselt werden, um ein sektorweises Image davon zu erstellen.

GNU ddrescue arbeitet auf Blockebene und kann die verschlüsselten Sektoren kopieren.

Cryptsetup kann anschließend auf das Image beziehungsweise das daraus bereitgestellte Blockgerät angewendet werden.

### Header als kritische Struktur

Der LUKS-Header und die Keyslot-Daten sind für den Zugriff auf verschlüsselte Daten von entscheidender Bedeutung.

Die Cryptsetup-Projektdokumentation warnt ausdrücklich davor, dass Beschädigungen dieser Strukturen zu dauerhaftem Datenverlust führen können.

### LUKS1

Bei LUKS1 ist der Header besonders kritisch.

Werden wesentliche Header- beziehungsweise Keyslotinformationen zerstört und existiert kein passendes Backup, können die verschlüsselten Daten kryptografisch nicht mehr zugänglich sein.

### LUKS2

LUKS2 besitzt gegenüber LUKS1 zusätzliche Redundanz bei den Metadaten.

Auch LUKS2 ist jedoch nicht gegen jede Beschädigung geschützt.

Ein Header beziehungsweise Keyslot kann weiterhin dauerhaft zerstört werden.

### Header-Backup

Cryptsetup besitzt dafür:

`luksHeaderBackup`

Syntax:

```
cryptsetup luksHeaderBackup DEVICE --header-backup-file DATEI
```

Damit wird ein binäres Backup des LUKS-Headers und des Keyslot-Bereichs erzeugt.

### Bedeutung des Header-Backups

Ein Header-Backup kann im Katastrophenfall entscheidend sein.

Es ist jedoch selbst hochsensibles Sicherheitsmaterial.

### Header-Backup enthält Schlüsselzugangsdaten

Die Cryptsetup-Dokumentation weist ausdrücklich darauf hin:

Ein Header-Backup zusammen mit einer Passphrase, die zum Zeitpunkt des Backups gültig war, kann den Zugriff auf den verschlüsselten Datenbereich ermöglichen – selbst wenn diese Passphrase später auf dem eigentlichen Datenträger geändert oder entfernt wurde.

### Konsequenz

Ein LUKS-Header-Backup muss ähnlich geschützt werden wie anderes hochsensibles Schlüsselmaterial.

Es sollte nicht:

öffentlich gespeichert,

ungeschützt per E-Mail versendet,

auf beliebigen Cloudspeichern abgelegt

oder zusammen mit der Passphrase unsicher verwahrt

werden.

### Header-Backup verändert die Löschstrategie

Existieren Header-Backups, genügt das alleinige Zerstören der Header-/Keyslot-Strukturen auf dem ursprünglichen Datenträger nicht mehr zwingend, um den Zugriff über diese alten Schlüsselzustände auszuschließen.

Auch vorhandene Header-Backups müssen bei einer sicheren Außerbetriebnahme berücksichtigt werden.

### Header wiederherstellen

Die Gegenoperation lautet:

```
cryptsetup luksHeaderRestore DEVICE --header-backup-file DATEI
```

Sie stellt einen gesicherten LUKS-Header und Keyslot-Bereich wieder her.

### Höchste Vorsicht bei `luksHeaderRestore`

`luksHeaderRestore` ist eine **schreibende und potenziell folgenschwere Operation**.

Die vorhandenen Header-/Keyslotinformationen werden durch den Backupzustand ersetzt.

Danach gelten die Schlüsselzugänge des wiederhergestellten Backups.

### Rescue-Regel

Ein Header-Restore sollte niemals der erste Versuch bei einem unbekannten oder beschädigten LUKS-Datenträger sein.

Vor einem Restore sollten mindestens:

Originalzustand gesichert,

aktueller Header soweit möglich gesichert,

Geräteidentität eindeutig geprüft

und Kompatibilität des Backups festgestellt

werden.

### Header testen, ohne ihn sofort zurückzuschreiben

Die Cryptsetup-FAQ beschreibt die Möglichkeit, ein vorhandenes Header-Backup über:

`--header`

als separaten Header zu verwenden.

Dadurch kann unter geeigneten Bedingungen geprüft werden, ob ein Backup das Volume öffnen kann, ohne es unmittelbar auf das Originalgerät zurückzuschreiben.

Dies ist für Rescue-Arbeiten wesentlich sicherer als ein vorschneller Header-Restore.

### Detached Header

LUKS kann auch mit einem vom verschlüsselten Datenbereich getrennten Header verwendet werden.

In solchen Konfigurationen liegt der LUKS-Header nicht zwingend am Anfang des Datenblocks.

Die Option:

`--header`

ermöglicht die Angabe eines separaten Header-Geräts beziehungsweise einer Header-Datei.

### Bedeutung für `isLuks`

Wird ein Detached Header verwendet, kann eine reine Untersuchung des Datenbereichs ohne den separaten Header zu dem Eindruck führen, es liege kein normal erkennbarer LUKS-Header vor.

Die tatsächliche Systemkonfiguration ist daher bei ungewöhnlichen Volumes zu berücksichtigen.

### Keyfile

Neben interaktiv eingegebenen Passphrasen kann Cryptsetup Schlüsselmaterial auch aus Dateien beziehen.

Ein solches Keyfile ist sicherheitskritisch.

Wer Zugriff auf ein gültiges Keyfile erhält, kann – abhängig von der LUKS-Konfiguration – das Volume entsperren.

### Keyfile nicht mit Header-Backup verwechseln

Ein:

**Keyfile**

und ein:

**LUKS-Header-Backup**

sind unterschiedliche Objekte.

Beide können jedoch sicherheitskritisch sein.

### Schlüssel hinzufügen

Cryptsetup unterstützt:

`luksAddKey`

Damit kann einem bestehenden LUKS-Volume ein weiterer Schlüsselzugang hinzugefügt werden.

Diese Operation verändert die LUKS-Metadaten.

### Schlüssel entfernen

Mit:

`luksRemoveKey`

kann ein vorhandener Schlüsselzugang entfernt werden.

Auch dies ist eine schreibende Verwaltungsoperation.

### Keyslot gezielt löschen

Mit:

`luksKillSlot DEVICE NUMMER`

kann ein bestimmter Keyslot gelöscht werden.

Dies ist besonders gefährlich, wenn nicht sicher bekannt ist, welche weiteren funktionierenden Schlüsselzugänge existieren.

### Passphrase ändern

`luksChangeKey`

ändert einen vorhandenen Schlüsselzugang.

Auch dies gehört nicht zu einer normalen read-only Rescue-Analyse.

### Rescue-Grundsatz für Keyslots

Bei einer Datenrettung gilt:

**Keine Keyslots verändern, solange das Ziel ausschließlich darin besteht, vorhandene Daten zu lesen und zu sichern.**

Jede unnötige Veränderung erhöht das Risiko.

## `luksFormat`

Die Operation:

`cryptsetup luksFormat DEVICE`

initialisiert ein Gerät als neues LUKS-Volume.

Dies ist eine **destruktive Erstellungsoperation**.

### Niemals zur „Reparatur“ verwenden

`luksFormat` ist kein Reparaturbefehl für einen beschädigten vorhandenen LUKS-Header.

Wird ein vorhandener Datenträger neu mit `luksFormat` initialisiert, werden für die bisherige Verschlüsselung entscheidende Metadaten überschrieben.

Das kann die Wiederherstellung vorhandener Daten unmöglich machen.

### Besonders gefährlicher Denkfehler

Ein beschädigtes LUKS-Volume darf nicht nach dem Muster:

„Header kaputt – einfach neu formatieren“

behandelt werden.

Bei Verschlüsselung kann gerade das Überschreiben des Headers die noch vorhandene letzte Möglichkeit zur Datenrettung zerstören.

### `erase` / `luksErase`

Cryptsetup besitzt außerdem eine Operation zum Löschen der LUKS-Keyslots.

Sie macht ein LUKS-Volume ohne anderweitig verfügbaren Volume Key beziehungsweise geeignete Sicherungen unzugänglich.

Diese Funktion ist keine Rescue-Funktion.

#### Kein normales Datenlöschen

`luksErase` überschreibt nicht zwangsläufig den gesamten verschlüsselten Nutzdatenbereich.

Es zerstört den regulären Schlüsselzugang über die LUKS-Keyslots.

Vor der Verwendung muss das Sicherheitsmodell einschließlich vorhandener Header-Backups verstanden werden.

### `convert`

Cryptsetup unterstützt die Konvertierung zwischen LUKS1 und LUKS2, soweit die Voraussetzungen erfüllt sind.

Eine Formatkonvertierung verändert kritische Metadaten.

Sie gehört deshalb nicht zu einer normalen Rescue-Erstanalyse.

### `config`

Bei LUKS2 können mit:

```
cryptsetup config
```

permanente Konfigurationsoptionen im LUKS-Header verändert werden.

Auch dies ist eine schreibende Verwaltungsfunktion.

#### Tokens

LUKS2 kann Tokenobjekte verwenden, über die zusätzliche Mechanismen zur Beschaffung von Passphrasen beziehungsweise Schlüsselinformationen eingebunden werden können.

Cryptsetup besitzt dafür Token-Verwaltungsfunktionen.

Für eine Rescue-Analyse gilt wiederum:

Vorhandene Tokeninformationen untersuchen, aber nicht ohne Not verändern.

#### Plain dm-crypt

Cryptsetup unterstützt neben LUKS auch:

**plain dm-crypt.**

Plain dm-crypt besitzt im Gegensatz zu LUKS keine entsprechenden selbstbeschreibenden LUKS-Metadaten auf dem Datenträger.

### Konsequenz bei Plain dm-crypt

Ohne die korrekten extern bekannten Parameter kann ein Plain-dm-crypt-Datenträger nicht einfach anhand eines LUKS-Headers erkannt und automatisch korrekt geöffnet werden.

Benötigt werden die zur ursprünglichen Einrichtung passenden Parameter.

### Keine sichere automatische Erkennung von Verschlüsselung allein anhand „zufälliger Daten“

Verschlüsselte Daten sollen gerade wie hochentropische Daten erscheinen.

Aus einem Blockbereich, der scheinbar zufällige Bytes enthält, kann deshalb nicht zuverlässig allein geschlossen werden:

welcher Verschlüsselungsmodus,

welcher Schlüssel,

welche Parameter

oder überhaupt welches Verschlüsselungssystem

verwendet wurde.

### LUKS erleichtert die Verwaltung

Genau hier liegt ein wesentlicher Vorteil von LUKS:

Die notwendigen Verwaltungsmetadaten werden in einem definierten Format gespeichert.

Die geheimen Zugangsdaten selbst müssen dennoch vorhanden sein.

### Keine Passwortwiederherstellung

Cryptsetup ist kein Werkzeug zum „Zurücksetzen“ eines unbekannten LUKS-Passworts.

Ist kein gültiger Schlüsselzugang vorhanden, kann Cryptsetup die Verschlüsselung nicht einfach umgehen.

Das ist eine grundlegende Eigenschaft sicherer Verschlüsselung.

### Verschlüsselung und Datenrettung

Bei einem verschlüsselten Datenträger existieren zwei voneinander unabhängige Schadensklassen:

**Verschlüsselungsebene beschädigt**

beispielsweise Header oder Keyslots.

und:

**Datenebene beschädigt**

beispielsweise Dateisystem oder Dateien innerhalb des entschlüsselten Volumes.

Beide Ebenen müssen separat diagnostiziert werden.

### Beispiel 1: LUKS intakt, Dateisystem beschädigt

`isLuks` erfolgreich.

`luksDump` erfolgreich.

`open` erfolgreich.

Dateisystemprüfung zeigt Schäden.

Dann liegt das primäre Problem oberhalb der Verschlüsselungsschicht.

### Beispiel 2: LUKS-Header beschädigt

`isLuks` schlägt unerwartet fehl.

Bekannt ist jedoch, dass der Datenträger zuvor LUKS verwendet hat.

Dann sollte keinesfalls sofort formatiert werden.

Stattdessen:

Original sichern,

Image erstellen,

vorhandene Header-Backups suchen,

Systemdokumentation prüfen

und erst danach weitere Wiederherstellungsmaßnahmen planen.

### Beispiel 3: Passphrase unbekannt

LUKS wird erkannt.

Header ist lesbar.

Es ist jedoch kein gültiger Schlüsselzugang bekannt.

Dann ist die technische Verschlüsselungsstruktur vorhanden, aber Cryptsetup kann das Volume ohne gültige Authentisierung nicht regulär öffnen.

Das ist kein Dateisystemfehler.

### Beispiel 4: Physisch defektes verschlüsseltes Laufwerk

SMART beziehungsweise Hardwarediagnose zeigt Probleme.

Lesevorgänge schlagen wiederholt fehl.

Dann:

**nicht zuerst umfangreiche Cryptsetup- oder Dateisystemtests durchführen.**

Stattdessen:

**GNU ddrescue → Image/Klon → Cryptsetup auf der Kopie → Dateisystemanalyse.**

### Beispiel 5: Volume lässt sich öffnen

Passphrase funktioniert.

`/dev/mapper/rescue` erscheint.

Dann sollte zunächst geprüft werden, welche Struktur sich innerhalb dieses Mappings befindet.

Nicht automatisch formatieren oder Reparaturprogramme starten.

## Cryptsetup und GNU ddrescue

GNU ddrescue arbeitet auf der Blockebene.

Cryptsetup arbeitet auf der Verschlüsselungsebene.

Für beschädigte verschlüsselte Datenträger ist deshalb die Reihenfolge häufig:

**ddrescue → cryptsetup → Dateisystemwerkzeuge.**

## Cryptsetup und SMART/NVMe Diagnose

SMART beziehungsweise NVMe Diagnose helfen festzustellen, ob ein physischer Datenträger Anzeichen eines Hardwareproblems zeigt.

Cryptsetup kann dagegen feststellen, ob die Verschlüsselungsschicht zugänglich ist.

Beides beantwortet unterschiedliche Fragen.

## Cryptsetup und GParted

GParted kann Partitionen und bestimmte Verschlüsselungsstrukturen erkennen beziehungsweise in seine Datenträgerdarstellung einbeziehen.

Cryptsetup ist jedoch das spezialisierte Kommandozeilenwerkzeug für die eigentliche dm-crypt-/LUKS-Verwaltung.

## Cryptsetup und GNOME Disks

GNOME Disks kann verschlüsselte Volumes grafisch in den Datenträgerworkflow integrieren.

Cryptsetup bietet dagegen die präzise Kommandozeilenkontrolle und detaillierte LUKS-Verwaltungsfunktionen.

## Cryptsetup und GnuPG

Cryptsetup und GnuPG gehören beide zum Bereich Verschlüsselung, lösen jedoch unterschiedliche Aufgaben.

**Cryptsetup/LUKS**

verschlüsselt Blockgeräte beziehungsweise Datenträger.

**GnuPG**

arbeitet unter anderem mit Dateien, Nachrichten, Signaturen und OpenPGP-Schlüsseln.

Ein LUKS-Volume ist daher nicht einfach eine „mit GPG verschlüsselte Festplatte“.

## Typischer sicherer Rescue-Workflow

**1. Datenträger identifizieren**

Gerätebezeichnung, Größe und Hardware eindeutig prüfen.

**2. Hardwarezustand beurteilen**

Bei Auffälligkeiten zuerst Image erstellen.

**3. Keine schreibenden Operationen durchführen**

**4. LUKS prüfen**

```
cryptsetup -v isLuks DEVICE
```

**5. Metadaten dokumentieren**

```
cryptsetup luksDump DEVICE
```

**6. LUKS UUID dokumentieren**

```
cryptsetup luksUUID DEVICE
```

**7. Wenn gültiger Schlüssel vorhanden: read-only öffnen**

```
cryptsetup open --readonly --type luks DEVICE rescue
```

**8. Mapping kontrollieren**

```
cryptsetup status rescue
```

**9. Innere Struktur untersuchen**

**10. Daten auf separates Zielmedium retten**

**11. Dateisystem ordnungsgemäß aushängen**

**12. Mapping schließen**

```
cryptsetup close rescue
```

## Grenzen

Cryptsetup repariert keinen physisch defekten Datenträger.

Cryptsetup ersetzt GNU ddrescue nicht.

Cryptsetup repariert kein beschädigtes Dateisystem.

Cryptsetup ersetzt Dateisystemwerkzeuge nicht.

Cryptsetup kann eine unbekannte starke Passphrase nicht einfach umgehen.

Ein erfolgreicher `luksDump` beweist nicht, dass die Nutzdaten vollständig intakt sind.

Ein erfolgreiches `open` beweist nicht, dass das Dateisystem intakt ist.

Ein negativer `isLuks`-Test beweist bei einem beschädigten Datenträger nicht sicher, dass niemals LUKS vorhanden war.

Plain dm-crypt besitzt keine mit LUKS vergleichbare selbstbeschreibende Headerstruktur.

## Sicherheitshinweise

Bei Datenrettung grundsätzlich zunächst read-only denken.

Gerätebezeichnungen vor jedem Befehl doppelt prüfen.

`luksFormat` niemals als Reparaturversuch für ein vorhandenes verschlüsseltes Volume verwenden.

Keyslots nicht während einer normalen Datenrettung verändern.

`luksKillSlot`, `luksRemoveKey`, `luksErase`, `convert`, `config` und Header-Restore nur mit vollständigem Verständnis der Auswirkungen einsetzen.

Vor kritischen Änderungen Header-Backup und vollständige Datensicherung vorsehen.

Header-Backups wie sensibles Schlüsselmaterial schützen.

Passphrasen und Keyfiles niemals in Dokumentationen oder Logs aufnehmen.

Debugging-Werkzeuge können sensible Daten offenlegen; insbesondere externe Trace-Ausgaben sind mit großer Vorsicht zu behandeln.

Bei physisch beschädigten Medien zuerst ein Image beziehungsweise einen Klon erstellen.

Nicht auf dem Original experimentieren, wenn eine Kopie möglich ist.

Ein entsperrtes `/dev/mapper`-Gerät enthält die Daten aus Sicht nachfolgender Programme entschlüsselt und muss entsprechend geschützt werden.

## Technische Einordnung in JUMBO BLOCK Rescue

Cryptsetup eröffnet im JUMBO BLOCK Rescue Handbuch den Bereich:

**Verschlüsselung.**

Seine zentrale Aufgabe ist die kontrollierte Überbrückung zwischen:

**verschlüsseltem Blockgerät**

und:

**entschlüsselter Datenebene.**

Die Rescue-Kette lautet bei einem verschlüsselten beschädigten Datenträger idealtypisch:

**SMART / NVMe Diagnose**

physische Situation beurteilen.

↓

**GNU ddrescue**

bei instabilem Medium zunächst eine sektorweise Arbeitskopie erzeugen.

↓

**Cryptsetup `isLuks` / `luksDump`**

Verschlüsselungsstruktur erkennen und dokumentieren.

↓

**Cryptsetup `open --readonly`**

Volume mit vorhandener gültiger Authentisierung kontrolliert öffnen.

↓

**/dev/mapper/...**

entschlüsselte Blockebene bereitstellen.

↓

**Dateisystemanalyse**

innere Struktur untersuchen.

↓

**Datenrettung**

Dateien auf ein separates Zielmedium sichern.

Damit nimmt Cryptsetup eine entscheidende Position im Rescue-System ein:

Ohne das Öffnen der Verschlüsselungsschicht sehen nachfolgende Dateisystem- und Datenrettungswerkzeuge lediglich die verschlüsselten Blöcke.

Die wichtigste Regel für den Rescue-Einsatz lautet deshalb:

**Erkennen und lesen vor verändern.**

Und bei beschädigten verschlüsselten Medien noch strenger:

**Zuerst sichern – dann entschlüsseln – erst danach reparieren.**

# A.26 GnuPG

**Bereich:** Verschlüsselung
**Programm:** GnuPG – GNU Privacy Guard
**Baseline 20261001:** GnuPG 2.4.7
**Installierte Kernpakete:** `gnupg 2.4.7-21+deb13u1`, `gnupg2 2.4.7-21+deb13u1`, `gpg 2.4.7-21+deb13u1+b5`, `gpg-agent 2.4.7-21+deb13u1+b5`, `gpgv 2.4.7-21+deb13u1+b5`, `gpgsm 2.4.7-21+deb13u1+b5`, `gpgconf 2.4.7-21+deb13u1+b5`
**Bedienung:** Kommandozeile
**Aufgabe:** OpenPGP-Signaturprüfung, Datei-Ver- und Entschlüsselung sowie Schlüsselverwaltung

## Zweck

GnuPG – GNU Privacy Guard – ist eine freie Implementierung kryptografischer Standards wie OpenPGP und S/MIME.

Für JUMBO BLOCK Rescue ist insbesondere das Programm:

`gpg`

relevant.

Es ermöglicht unter anderem:

digitale Signaturen zu prüfen,

Dateien zu signieren,

Dateien für Empfänger zu verschlüsseln,

verschlüsselte Dateien zu entschlüsseln,

symmetrisch mit einer Passphrase zu verschlüsseln,

öffentliche Schlüssel zu importieren und zu exportieren,

Fingerprints anzuzeigen

und Schlüsselbestände zu verwalten.

Zusätzlich steht mit:

`gpgv`

ein spezialisiertes Werkzeug ausschließlich zur OpenPGP-Signaturprüfung zur Verfügung.

## Einsatzgebiet im Rescue-System

GnuPG eignet sich innerhalb von JUMBO BLOCK Rescue insbesondere für:

Prüfung digital signierter Downloads,

Verifikation von Softwarepaketen und Images,

Prüfung signierter Herstellerdateien,

Kontrolle von OpenPGP-Signaturen,

Verifikation einer Datei gegen eine separate `.sig`- oder `.asc`-Datei,

Identifikation des verwendeten Signaturschlüssels,

Vergleich von Schlüsselfingerprints,

Entschlüsselung autorisiert vorliegender OpenPGP-Dateien,

Erzeugung eigener Signaturen,

Verschlüsselung sensibler Rescue-Daten

und sichere Übertragung von Dateien an bekannte Empfänger.

## Cryptsetup und GnuPG unterscheiden

Obwohl beide Programme im Bereich Verschlüsselung stehen, erfüllen sie grundlegend unterschiedliche Aufgaben.

**Cryptsetup**

arbeitet primär auf Blockgeräteebene.

Typischer Anwendungsfall:

verschlüsselte Festplatte oder Partition.

**GnuPG**

arbeitet primär mit Daten, Dateien, Nachrichten, Signaturen und kryptografischen Schlüsseln.

Typischer Anwendungsfall:

`rescue-image.iso`

plus:

`rescue-image.iso.sig`

Cryptsetup beantwortet damit beispielsweise:

**Wie öffne ich dieses verschlüsselte Volume?**

GnuPG beantwortet beispielsweise:

**Ist diese Datei tatsächlich mit dem erwarteten OpenPGP-Schlüssel signiert worden?**

## Digitale Signatur

Eine digitale Signatur verbindet kryptografisch:

Daten

und:

einen privaten Signaturschlüssel.

Die Signatur kann anschließend mit dem zugehörigen öffentlichen Schlüssel überprüft werden.

## Was eine erfolgreiche Signaturprüfung technisch aussagt

Eine erfolgreiche kryptografische Prüfung zeigt, dass:

die geprüften Daten zur Signatur passen

und:

die Signatur mit dem zum verwendeten öffentlichen Schlüssel gehörenden privaten Schlüssel erzeugt wurde.

Entscheidend bleibt jedoch die nächste Frage:

**Ist dieser öffentliche Schlüssel tatsächlich der Schlüssel der erwarteten Person, Organisation oder des erwarteten Herstellers?**

## Signaturprüfung ist daher zweistufig

Eine belastbare Prüfung besteht aus zwei getrennten Fragen:

**1. Ist die Signatur kryptografisch gültig?**

und:

**2. Ist der verwendete Schlüssel authentisch der erwarteten Quelle zugeordnet?**

Nur die erste Frage allein reicht für eine sichere Herkunftsbewertung nicht aus.

## Detached Signature

Bei Softwaredownloads wird häufig eine:

**Detached Signature**

verwendet.

Dabei liegen Nutzdatei und Signatur getrennt vor.

Beispiel:

`rescue.iso`

und:

`rescue.iso.sig`

Die Signaturdatei enthält nicht die komplette Nutzdatei, sondern die für die Signaturprüfung erforderlichen OpenPGP-Daten.

## Detached Signature prüfen

Die offizielle GnuPG-Syntax lautet:

```
gpg --verify SIGNATUR DATEI
```

Beispielprinzip:

```
gpg --verify rescue.iso.sig rescue.iso
```

GnuPG empfiehlt bei Detached Signatures ausdrücklich, beide Dateinamen anzugeben.

### Warum beide Dateien ausdrücklich angeben?

GnuPG besitzt historische Mechanismen, bei nur einer angegebenen Signaturdatei den Namen der zugehörigen Datendatei abzuleiten.

Die aktuelle GnuPG-Dokumentation rät bei Detached Signatures ausdrücklich davon ab, sich darauf zu verlassen.

Für einen reproduzierbaren Rescue-Workflow sollte deshalb immer angegeben werden:

**Signaturdatei**

und:

**zu prüfende Datendatei.**

### Beispiel für eine ISO-Prüfung

Ein typischer Ablauf lautet:

```
gpg --verify image.iso.sig image.iso
```

GnuPG prüft dabei die Signatur gegen exakt die angegebene ISO-Datei.

Schon eine signaturrelevante Veränderung der Datei führt dazu, dass die Signatur nicht mehr korrekt verifiziert.

### „Good signature“

Eine Meldung über eine gute Signatur bestätigt die kryptografische Übereinstimmung.

Sie darf jedoch nicht isoliert betrachtet werden.

Entscheidend ist zusätzlich:

**Welcher Schlüssel hat signiert?**

### Fingerprint

Ein öffentlicher OpenPGP-Schlüssel besitzt einen Fingerprint.

Dieser Fingerprint dient als wesentlich präzisere Identifikation des Schlüssels als beispielsweise:

Name,

E-Mail-Adresse

oder eine kurze Key-ID.

Fingerprints können mit:

```
gpg --fingerprint
```

angezeigt werden. Die offizielle GnuPG-Dokumentation führt diese Funktion ausdrücklich als Standardverfahren zur Schlüsseldarstellung auf.

### Warum der Fingerprint wichtig ist

Ein Schlüssel kann eine User ID enthalten wie:

```
Example Vendor <security@example.com>
```

Diese Zeichenfolge allein beweist nicht, dass der Schlüssel tatsächlich vom genannten Hersteller stammt.

Jeder könnte grundsätzlich einen eigenen OpenPGP-Schlüssel mit einer frei gewählten User ID erzeugen.

Deshalb muss die Schlüsselidentität über eine vertrauenswürdige Quelle überprüft werden.

### Vertrauenswürdiger Vergleich

Der Fingerprint sollte möglichst über einen unabhängigen, authentisierten Kanal bezogen werden.

Beispiele können sein:

offizielle Herstellerdokumentation,

bereits vertrauenswürdig vorhandene Schlüssel,

offiziell veröffentlichte Releaseinformationen

oder ein anderer unabhängig geprüfter Kommunikationskanal.

### Zentrale Rescue-Regel

**Eine gültige Signatur mit einem unbekannten Schlüssel beweist nicht automatisch die erwartete Herkunft.**

Sie beweist zunächst nur die kryptografische Beziehung zwischen:

Datei,

Signatur

und:

dem verwendeten Schlüssel.

### Öffentliche Schlüssel anzeigen

Mit:

```
gpg --list-keys
```

werden öffentliche Schlüssel im aktuellen GnuPG-Schlüsselbestand angezeigt.

Ein bestimmter Schlüssel kann durch eine geeignete Suchangabe eingeschränkt werden.

### Geheime Schlüssel anzeigen

Mit:

```
gpg --list-secret-keys
```

werden vorhandene geheime Schlüssel beziehungsweise geheime Schlüsselkomponenten aufgelistet.

Diese Information ist sicherheitskritischer als eine reine Liste öffentlicher Schlüssel.

### Schlüssel importieren

Ein öffentlicher Schlüssel kann beispielsweise mit:

```
gpg --import DATEI
```

importiert werden.

Vor dem Vertrauen in diesen Schlüssel muss seine Authentizität separat geprüft werden.

### Import ist kein Vertrauensbeweis

Ein erfolgreicher:

```
gpg --import vendor-key.asc
```

bedeutet lediglich:

GnuPG konnte die enthaltenen Schlüsseldaten in den Schlüsselbestand importieren.

Es bedeutet nicht:

**Dieser Schlüssel gehört wirklich zum angegebenen Hersteller.**

### Schlüssel vor dem Import untersuchen

Bei fremden Schlüsseldateien ist es sinnvoll, deren Inhalt beziehungsweise Fingerprint zunächst zu untersuchen, bevor daraus Vertrauen abgeleitet wird.

Die Entscheidung über die Authentizität muss anhand einer unabhängigen Referenz erfolgen.

### Öffentlichen Schlüssel exportieren

Öffentliche Schlüssel können exportiert werden.

Eine menschenlesbare ASCII-Armor-Darstellung kann beispielsweise mit:

```
--armor
```

erzeugt werden.

ASCII-armored OpenPGP-Daten sind typischerweise an Blöcken wie:

`-----BEGIN PGP PUBLIC KEY BLOCK-----`

erkennbar.

### `--armor`

Mit:

`--armor`

beziehungsweise:

`-a`

erzeugt GnuPG ASCII-armored statt binärer OpenPGP-Ausgabe, soweit die betreffende Operation dies unterstützt.

Dies erleichtert beispielsweise die Übertragung kryptografischer Daten über textorientierte Kanäle.

### Binär und ASCII Armor

Eine `.asc`-Datei ist häufig ASCII-armored.

Eine `.sig`-Datei kann eine binäre Detached Signature enthalten.

Die Dateiendung allein ist jedoch kein kryptografischer Beweis für den tatsächlichen Inhalt.

### Datei signieren

Mit:

`gpg --sign DATEI`

kann eine Datei signiert werden.

Dabei entsteht eine OpenPGP-signierte Datenstruktur.

### Detached Signature erzeugen

Für Softwaredateien ist häufig eine separate Signatur sinnvoll:

`gpg --detach-sign DATEI`

Kurzform:

`gpg -b DATEI`

Die offizielle GnuPG-Dokumentation beschreibt `--detach-sign` beziehungsweise `-b` ausdrücklich als Erzeugung einer Detached Signature.

### ASCII-armored Detached Signature

In Verbindung mit:

`--armor`

kann eine textbasierte Detached Signature erzeugt werden.

Beispiel:

```
gpg --armor --detach-sign DATEI
```

### Signaturschlüssel auswählen

Wenn mehrere geheime Schlüssel vorhanden sind, kann mit:

```
--local-user
```

beziehungsweise:

```
-u
```

der gewünschte Signaturschlüssel ausgewählt werden.

Dies verhindert, dass versehentlich ein anderer Standardschlüssel verwendet wird.

### Cleartext Signature

Mit:

```
gpg --clear-sign DATEI
```

kann eine lesbare Textdatei mit einer OpenPGP-Cleartext-Signatur versehen werden.

Der Text bleibt dabei grundsätzlich direkt lesbar.

### Cleartext Signatures und Binärdateien

Für Binärdateien wie:

ISO-Images,

Archive,

Programme

oder Firmware

ist eine Detached Signature typischerweise die klarere Struktur:

Originaldatei bleibt unverändert,

Signatur liegt separat vor.

### Verifikation verändert die Datei nicht

```
gpg --verify
```

prüft eine Signatur, ohne die zu prüfende Datei umzuschreiben.

Damit eignet sich der Befehl besonders für Rescue- und Integritätsprüfungen.

## GnuPG und Prüfsummen

Eine kryptografische Prüfsumme und eine digitale Signatur erfüllen unterschiedliche Aufgaben.

Ein veröffentlichter SHA-256-Wert kann beantworten:

**Hat meine Datei denselben Hash wie der veröffentlichte Referenzwert?**

Eine korrekt authentisierte digitale Signatur kann zusätzlich beantworten:

**Wurde diese Datenfassung mit dem privaten Schlüssel der erwarteten Quelle signiert?**

## Prüfsumme ohne authentisierte Quelle

Wenn ein Angreifer sowohl die Download-Datei als auch den daneben veröffentlichten Hashwert manipulieren kann, schützt der bloße Vergleich beider Werte nicht vor dieser gemeinsamen Manipulation.

Eine digitale Signatur verwendet dagegen einen privaten Schlüssel, der nicht auf dem Downloadserver vorhanden sein muss.

## Herstellerdownloads

Ein belastbarer Downloadworkflow kann deshalb lauten:

**1. Datei aus der vorgesehenen Quelle beziehen**

**2. Signaturdatei beziehen**

**3. Hersteller-Schlüssel aus vertrauenswürdiger Quelle beziehen**

**4. Fingerprint unabhängig prüfen**

**5. Signatur prüfen**

```
gpg --verify SIGNATUR DATEI
```

**6. Ergebnis einschließlich Signaturschlüssel dokumentieren**

## Bereits vertrauenswürdiges GnuPG verwenden

Das GnuPG-Projekt weist bei seiner eigenen Integritätsprüfung auf einen wichtigen Bootstrapping-Grundsatz hin:

Eine neu heruntergeladene GnuPG-Version sollte nicht selbst als alleinige Grundlage dienen, um ihre eigene Authentizität zu bestätigen.

Zur Prüfung sollte eine bereits vertrauenswürdige GnuPG-Installation verwendet werden.

## Bedeutung für JUMBO BLOCK Rescue

Eine fest in der geprüften Rescue-Baseline enthaltene GnuPG-Installation kann deshalb ein wertvolles Werkzeug sein, um später heruntergeladene signierte Dateien unabhängig vom Zielsystem zu überprüfen.

Voraussetzung bleibt, dass auch die verwendeten Signaturschlüssel authentisch geprüft wurden.

### Datei für einen Empfänger verschlüsseln

Mit:

`gpg --encrypt`

beziehungsweise:

`gpg -e`

können Daten für einen oder mehrere öffentliche Schlüssel verschlüsselt werden.

Ein Empfänger wird typischerweise mit:

`--recipient`

beziehungsweise:

`-r`

angegeben.

### Prinzip der Public-Key-Verschlüsselung

Zum Verschlüsseln wird der öffentliche Schlüssel des Empfängers verwendet.

Zum Entschlüsseln wird der passende geheime Schlüssel benötigt.

Dadurch muss der geheime Schlüssel nicht an den Absender weitergegeben werden.

### Vor dem Verschlüsseln Empfängerschlüssel prüfen

Auch hier gilt:

Ein importierter öffentlicher Schlüssel mit passendem Namen ist nicht automatisch authentisch.

Wird für den falschen Schlüssel verschlüsselt, kann die Datei möglicherweise nur der Besitzer dieses falschen privaten Schlüssels entschlüsseln.

Der Fingerprint des Empfängerschlüssels muss deshalb vor sicherheitskritischer Nutzung geprüft werden.

### Entschlüsseln

Mit:

`gpg --decrypt DATEI`

beziehungsweise:

`gpg -d DATEI`

werden OpenPGP-verschlüsselte Daten entschlüsselt.

Standardmäßig schreibt `--decrypt` die entschlüsselten Daten auf die Standardausgabe, sofern kein anderes Ausgabeverhalten angegeben wird.

### Ausgabe in Datei

Mit:

```
--output DATEI
```

beziehungsweise:

```
-o DATEI
```

kann eine Ausgabedatei angegeben werden.

Beispielprinzip:

```
gpg --output daten.tar --decrypt daten.tar.gpg
```

### Signierte verschlüsselte Datei

Ist eine entschlüsselte Datei zusätzlich signiert, prüft `gpg --decrypt` die enthaltene Signatur ebenfalls.

Das Ergebnis dieser Signaturprüfung muss wiederum hinsichtlich des verwendeten Schlüssels bewertet werden.

### Symmetrische Verschlüsselung

GnuPG kann Daten auch ohne Empfänger-Public-Key symmetrisch mit einer Passphrase verschlüsseln:

```
gpg --symmetric DATEI
```

Kurzform:

```
gpg -c DATEI
```

Die Entschlüsselung erfordert dann die passende Passphrase.

### Symmetrische Verschlüsselung im Rescue-Einsatz

Dies kann beispielsweise sinnvoll sein, wenn sensible Diagnose- oder Exportdateien auf einem Transportmedium geschützt werden müssen und kein geeignetes Public-Key-Verfahren vorgesehen ist.

### Passphrase

Die Sicherheit symmetrischer Verschlüsselung hängt wesentlich von der Qualität und Geheimhaltung der Passphrase ab.

Eine kurze oder leicht erratbare Passphrase kann den Schutz erheblich schwächen.

### Passphrase nicht in Kommandozeile schreiben

Sensible Passphrasen sollten nicht als Klartext in Shell-Kommandos, Skripten oder Dokumentationen hinterlegt werden.

Kommandozeilen können abhängig vom System in:

Shell-History,

Prozessinformationen,

Logs

oder Dokumentationen

auftauchen.

## `gpg-agent`

Die Baseline enthält:

`gpg-agent`.

GnuPG verwendet den Agenten unter anderem für die Verwaltung geheimer Schlüsseloperationen und Passphraseninteraktion.

### Passphrase-Caching

Der Agent kann Passphrasen zeitweise zwischenspeichern.

Das erhöht den Bedienkomfort, bedeutet aber:

Eine einmal eingegebene Passphrase muss während der Cache-Lebensdauer möglicherweise nicht bei jeder weiteren Operation erneut eingegeben werden.

### Bedeutung im Rescue-System

Bei gemeinsam verwendeten oder unbeaufsichtigten Rescue-Systemen sollte berücksichtigt werden, dass eine zuvor freigeschaltete Schlüsseloperation gegebenenfalls noch über den Agenten verfügbar sein kann.

### Agent beenden

GnuPG stellt über:

`gpgconf`

Mechanismen zur Verwaltung seiner Hintergrundkomponenten bereit.

Nach sensiblen Arbeiten kann es sinnvoll sein, vorhandene Agentzustände bewusst zu beenden beziehungsweise das Rescue-System kontrolliert herunterzufahren.

### Private Schlüssel

Geheime OpenPGP-Schlüssel sind hochsensibles Material.

Wer einen privaten Signaturschlüssel kontrolliert, kann gültige Signaturen unter diesem Schlüssel erzeugen.

Wer einen privaten Entschlüsselungsschlüssel kontrolliert, kann für diesen Schlüssel verschlüsselte Daten entschlüsseln, soweit die übrigen Voraussetzungen erfüllt sind.

### Private Schlüssel nicht unnötig importieren

Für die reine Prüfung einer Herstellersignatur wird nur der öffentliche Schlüssel benötigt.

Es gibt keinen Grund, dafür private Schlüssel auf das Rescue-System zu übertragen.

### Public-Key-Verifikation ist read-only bezüglich des geprüften Artefakts

Eine Signaturprüfung benötigt keine Veränderung der zu prüfenden Datei.

Dies ist für Rescue-Arbeiten besonders wertvoll:

Ein Image, Archiv oder Firmwarepaket kann geprüft werden, ohne seinen Inhalt zu verändern.

## gpgv

Die Baseline enthält außerdem:

`gpgv 2.4.7`.

`gpgv` ist eine reduzierte GnuPG-Komponente, die ausschließlich OpenPGP-Signaturen überprüft.

### Unterschied gpg und gpgv

`gpg`

ist das umfassende OpenPGP-Werkzeug für:

Schlüsselverwaltung,

Signieren,

Verschlüsseln,

Entschlüsseln

und Verifizieren.

`gpgv`

ist auf:

**Signaturverifikation**

beschränkt.

### Vertrauensmodell von gpgv

Ein wichtiger Unterschied:

gpgv behandelt die Schlüssel seines verwendeten Trusted-Keyrings als vertrauenswürdig.

Es verwendet damit ein bewusst vereinfachtes Vertrauensmodell und prüft laut GnuPG-Dokumentation insbesondere nicht selbst, ob diese Schlüssel abgelaufen oder widerrufen sind.

### Konsequenz

Bei gpgv ist besonders wichtig, dass der verwendete Keyring bereits kontrolliert und vertrauenswürdig zusammengestellt wurde.

gpgv beantwortet nicht die vorgelagerte Frage:

**Wie wurde die Echtheit dieses öffentlichen Schlüssels festgestellt?**

### gpgv mit Keyring

Mit:

`--keyring DATEI`

kann ein bestimmter vertrauenswürdiger Keyring für die Prüfung angegeben werden.

Mehrere `--keyring`-Optionen können kombiniert werden.

### Detached Signature mit gpgv

Das Grundprinzip lautet:

`gpgv SIGNATUR DATEI`

Auch hier sollten Signatur und Datendatei bei Detached Signatures ausdrücklich angegeben werden.

### Rückgabewert von gpgv

Die offizielle Dokumentation nennt:

`0` – Prüfung erfolgreich,

`1` – mindestens eine Signatur war ungültig,

andere Fehlercodes – schwerwiegende Fehler.

### gpg und Exitcode

Bei gpg ist ein bloßer Prozess-Exitcode für automatisierte Signaturentscheidungen nicht ausreichend.

Die GnuPG-Dokumentation weist ausdrücklich darauf hin, dass bei Skripten bekannt sein muss:

**was signiert wurde**

und:

**wer signiert hat.**

Für automatisierte Verifikation sollen geeignete Statusinformationen oder das speziell dafür vorgesehene `gpgv` verwendet werden.

### Signaturprüfung in Skripten

Ein Skript sollte deshalb nicht lediglich:

```
gpg --verify ...
```

ausführen und bei Exitcode 0 pauschal:

„Herstellerdatei authentisch“

melden.

Zusätzlich muss sichergestellt werden, dass der tatsächliche Signaturschlüssel der erwartete Schlüssel ist.

### Schlüssel-ID versus Fingerprint

Kurze Key-IDs sind für eine belastbare Schlüsselidentifikation ungeeignet.

Für sicherheitsrelevante Prüfungen sollte mit dem vollständigen Fingerprint gearbeitet werden.

### User IDs

Ein OpenPGP-Schlüssel kann eine oder mehrere User IDs enthalten.

Typischerweise enthalten diese:

Name,

E-Mail-Adresse

und optional weitere Angaben.

Diese Angaben sind Bestandteil der Schlüsselidentität innerhalb OpenPGP, aber nicht automatisch unabhängig verifiziert.

### Vertrauen

GnuPG verwaltet neben der rein kryptografischen Schlüsselstruktur auch Vertrauensinformationen.

Dabei muss zwischen zwei Fragen unterschieden werden:

**Ist eine Signatur mathematisch gültig?**

und:

**Wie stark vertraut GnuPG der Zuordnung dieses Schlüssels zu einer Identität?**

Diese Aussagen sind nicht identisch.

### „WARNING: This key is not certified with a trusted signature“

Eine solche Warnung bedeutet nicht automatisch:

**Die Dateisignatur ist kryptografisch falsch.**

Sie kann vielmehr bedeuten, dass GnuPG keine ausreichende Vertrauenskette für die Identitätszuordnung des Schlüssels kennt.

Deshalb müssen:

Signaturgültigkeit

und:

Schlüsselvertrauen

getrennt interpretiert werden.

### Rescue-Workflow ohne aufgebautes Web of Trust

Für Herstellerdateien ist häufig ein pragmatischer und belastbarer Ansatz:

**1. Hersteller-Public-Key beschaffen**

**2. vollständigen Fingerprint über einen unabhängigen vertrauenswürdigen Kanal verifizieren**

**3. Datei mit genau diesem Schlüssel prüfen**

Damit hängt die Herkunftsbewertung nicht davon ab, ob auf dem Rescue-System bereits ein persönliches Web of Trust aufgebaut wurde.

### Schlüsselserver

GnuPG kann über `dirmngr` öffentliche Schlüssel aus Netzwerkquellen beziehungsweise Schlüsselverzeichnisdiensten beziehen.

Die Baseline enthält `dirmngr` aus der GnuPG-2.4.7-Paketgeneration.

### Schlüsselserver lösen das Authentizitätsproblem nicht

Dass ein Schlüssel von einem öffentlichen Schlüsselserver heruntergeladen werden kann, beweist nicht, dass er zur erwarteten Organisation gehört.

Auch nach einem Netzwerkabruf muss die Identität des Schlüssels anhand eines vertrauenswürdigen Fingerprints geprüft werden.

### Netzwerk und Offline-Betrieb

Für die eigentliche Signaturprüfung benötigt GnuPG keine Internetverbindung, wenn:

zu prüfende Datei,

Signatur

und:

benötigter öffentlicher Schlüssel

bereits lokal vorliegen.

Dies ist für JUMBO BLOCK Rescue besonders wertvoll.

## Offline-Verifikation

Ein vollständig offline durchführbarer Prüfablauf lautet:

**DATEI**

- 

**SIGNATUR**

- 

**bereits verifizierter öffentlicher Schlüssel**

→

**GnuPG-Signaturprüfung**

Damit muss das kompromittierungsverdächtige Zielsystem nicht für die Integritätsprüfung verwendet werden.

## Beispiel: Rescue-ISO prüfen

Angenommen, es liegen vor:

`rescue.iso`

`rescue.iso.sig`

und ein zuvor authentisierter Hersteller-Public-Key.

Dann:

```
gpg --verify rescue.iso.sig rescue.iso
```

Anschließend wird geprüft:

Ist die Signatur gut?

Welcher Schlüssel wurde verwendet?

Stimmt dessen vollständiger Fingerprint mit der verifizierten Herstellerreferenz überein?

## Beispiel: Signatur ist ungültig

Meldet GnuPG eine ungültige Signatur, darf die Datei nicht als unverändert gegenüber dem signierten Original behandelt werden.

Mögliche Ursachen sind:

Datei verändert,

falsche Signaturdatei,

Signatur beschädigt,

falsche Kombination von Datei und Signatur.

### Beispiel: Public Key fehlt

Kann GnuPG die Signatur nicht prüfen, weil der öffentliche Schlüssel fehlt, ist dies nicht dasselbe wie:

**Bad signature.**

Die kryptografische Prüfung kann dann mangels Schlüssel nicht vollständig durchgeführt werden.

Der benötigte Schlüssel muss aus einer vertrauenswürdigen Quelle beschafft und authentisiert werden.

### Beispiel: Good Signature, unbekannter Schlüssel

GnuPG meldet eine gute Signatur.

Der verwendete Schlüssel ist jedoch nicht als erwarteter Herstellerschlüssel verifiziert.

Dann lautet das Ergebnis nicht:

**Herstellerdatei sicher bestätigt.**

Sondern:

**Signatur technisch gültig; Identität des Signaturschlüssels noch zu verifizieren.**

### Beispiel: Download mit Chromium

Ein typischer kombinierter Rescue-Workflow lautet:

**Chromium**

Datei und Signatur von der vorgesehenen Quelle herunterladen.

↓

**GnuPG**

Signatur prüfen.

↓

**Fingerprint**

gegen eine unabhängige Herstellerreferenz vergleichen.

↓

**ClamAV**

Datei bei Bedarf zusätzlich auf bekannte Schadsoftware untersuchen.

Diese Prüfungen beantworten unterschiedliche Fragen.

## GnuPG und ClamAV

**GnuPG**

prüft kryptografische Signaturen beziehungsweise schützt Daten kryptografisch.

**ClamAV**

sucht nach bekannten beziehungsweise erkennbaren Schadsoftwaremerkmalen.

Eine gültig signierte Datei kann theoretisch trotzdem unerwünschtes Verhalten enthalten, wenn der legitime Herausgeber eine problematische Datei signiert hat oder sein Signaturschlüssel kompromittiert wurde.

Umgekehrt beweist ein fehlender Malwarefund nicht die Herkunft einer Datei.

## GnuPG und Hashwerte

**Hashvergleich**

prüft Übereinstimmung mit einem Referenzhash.

**Digitale Signatur**

bindet die Daten kryptografisch an einen Signaturschlüssel.

Beides kann kombiniert werden.

## GnuPG und Cryptsetup

Cryptsetup schützt:

**Blockgeräte.**

GnuPG schützt beziehungsweise signiert:

**Daten und Dateien.**

Ein sinnvoller Rescue-Fall kann deshalb beide Werkzeuge kombinieren:

Daten aus einem LUKS-Volume retten,

ausgewählte sensible Rescue-Dateien anschließend mit GnuPG verschlüsseln,

und diese kontrolliert an einen berechtigten Empfänger übertragen.

## GnuPG und SSH

Der nachfolgende Remote-Bereich verwendet mit SSH ebenfalls Public-Key-Kryptografie.

OpenPGP-Schlüssel und normale SSH-Schlüssel sind jedoch unterschiedliche Schlüssel- und Protokollsysteme.

Sie dürfen nicht einfach gleichgesetzt werden.

### Metadaten bleiben relevant

Verschlüsselung schützt nicht zwangsläufig sämtliche Metadaten außerhalb der verschlüsselten Nutzdaten.

Beispielsweise können:

Dateiname,

Dateigröße,

Zeitpunkte,

Speicherort

oder Übertragungsmetadaten

abhängig vom verwendeten Workflow weiterhin sichtbar sein.

### Sichere Löschung

Das Löschen einer unverschlüsselten Originaldatei nach GnuPG-Verschlüsselung garantiert auf modernen Dateisystemen und insbesondere SSDs nicht automatisch, dass alle früheren Datenkopien physisch überschrieben wurden.

GnuPG ist kein Werkzeug für garantiertes physisches Secure Erase.

### Rescue-System und Schlüsselmaterial

Werden eigene geheime Schlüssel auf JUMBO BLOCK Rescue importiert, muss bedacht werden, dass Schlüsselmaterial beziehungsweise Agentzustände auf dem verwendeten System verarbeitet werden.

Für reine Hersteller-Signaturprüfungen sollten deshalb ausschließlich öffentliche Schlüssel verwendet werden.

### Grenzen

GnuPG repariert keine beschädigten Dateien.

GnuPG ist kein Malware-Scanner.

GnuPG ersetzt ClamAV nicht.

GnuPG ist kein Datenträgerverschlüsselungswerkzeug im Sinne von LUKS.

GnuPG ersetzt Cryptsetup nicht.

Eine gültige Signatur beweist nicht ohne zusätzliche Schlüsselprüfung die reale Identität des Herausgebers.

Ein importierter Public Key ist nicht automatisch vertrauenswürdig.

Ein Name oder eine E-Mail-Adresse in einer User ID ist kein Identitätsbeweis.

Ein Schlüsselserver ist keine automatische Vertrauensinstanz.

Verschlüsselung garantiert nicht die Löschung unverschlüsselter Vorversionen.

Verifikation kann nur die tatsächlich signierten Daten bewerten.

## Sicherheitshinweise

Für sicherheitskritische Signaturprüfungen den vollständigen Fingerprint des erwarteten Schlüssels verifizieren.

Fingerprints möglichst über einen unabhängigen vertrauenswürdigen Kanal beziehen.

Bei Detached Signatures Signaturdatei und Nutzdatei ausdrücklich angeben.

Eine „Good signature“-Meldung niemals ohne Prüfung des Signaturschlüssels als vollständigen Herkunftsbeweis behandeln.

Private Schlüssel nur auf ein Rescue-System übertragen, wenn dies wirklich erforderlich ist.

Für reine Verifikation werden keine privaten Schlüssel benötigt.

Passphrasen nicht in Shell-History, Skripten oder Dokumentationen hinterlegen.

Schlüsseldateien und Backups sicher schützen.

Bei gemeinsam genutzten Rescue-Systemen mögliche Passphrase-Caches des `gpg-agent` berücksichtigen.

Heruntergeladene Schlüssel nicht allein aufgrund ihrer User ID als authentisch betrachten.

Unbekannte Schlüssel nicht blind als vertrauenswürdig markieren.

Signaturprüfung möglichst mit einer bereits vertrauenswürdigen GnuPG-Installation durchführen.

## Technische Einordnung in JUMBO BLOCK Rescue

GnuPG schließt innerhalb von JUMBO BLOCK Rescue den Bereich:

**Verschlüsselung.**

Die beiden Programme dieses Bereichs besitzen klar getrennte Aufgaben:

**A.25 Cryptsetup**

Blockgeräteverschlüsselung und LUKS.

**A.26 GnuPG**

Dateiverschlüsselung, OpenPGP-Signaturen und Schlüsselprüfung.

Für JUMBO BLOCK Rescue ist insbesondere die Signaturprüfung von großer Bedeutung.

Ein typischer sicherer Downloadworkflow lautet:

**Chromium**

Datei herunterladen.

↓

**GnuPG**

digitale Signatur prüfen.

↓

**Fingerprint**

Signaturschlüssel unabhängig authentisieren.

↓

**ClamAV**

Datei gegebenenfalls zusätzlich auf bekannte Schadsoftware untersuchen.

↓

**erst danach**

Datei verwenden.

Für verschlüsselte Rescue-Daten lautet ein anderer Workflow:

**Daten retten**

↓

**Empfängerschlüssel authentisieren**

↓

**mit GnuPG verschlüsseln**

↓

**verschlüsselte Datei übertragen**

↓

**berechtigter Empfänger entschlüsselt mit seinem privaten Schlüssel.**

Die wichtigste Grundregel für GnuPG innerhalb von JUMBO BLOCK Rescue lautet:

**Eine kryptografisch gültige Signatur ist nur so aussagekräftig wie die verifizierte Identität des Schlüssels, mit dem sie erzeugt wurde.**

Damit liefert GnuPG nicht lediglich Verschlüsselung.

Es schafft bei korrekt verifiziertem Schlüssel eine kryptografisch prüfbare Verbindung zwischen:

**Datei**

und:

**erwarteter Quelle.**

# A.27 SSH / OpenSSH

**Bereich:** Remote
**Programm:** SSH – OpenSSH Client
**Baseline 20261001:** OpenSSH 10.0p1
**Installiertes Kernpaket:** `openssh-client 1:10.0p1-7+deb13u4`
**Zusätzlich belegt:** `ssh-askpass-gnome 1:10.0p1-7+deb13u4`
**Bedienung:** Kommandozeile
**Aufgabe:** Verschlüsselter Remote-Zugang zu autorisierten SSH-Systemen, Remote-Kommandos, sichere Dateiübertragung und Tunnel

## Zweck

SSH – Secure Shell – stellt eine verschlüsselte Verbindung zu einem entfernten System her.

Der in JUMBO BLOCK Rescue installierte OpenSSH-Client kann insbesondere verwendet werden für:

Remote-Shell-Zugriffe,

Ausführung einzelner Kommandos auf entfernten Systemen,

Authentisierung per Passwort oder Public Key, soweit der Server dies zulässt,

Übertragung von Dateien,

SFTP-Verbindungen,

Portweiterleitungen,

Zugriff über Jump Hosts

und Diagnose von SSH-Verbindungen.

## Baseline-Abgrenzung

Die Baseline 20261001 belegt:

`openssh-client 1:10.0p1-7+deb13u4`

und:

`ssh-askpass-gnome 1:10.0p1-7+deb13u4`.

Ein installiertes Paket:

`openssh-server`

ist durch die vorliegende Paketbaseline nicht belegt.

Daraus folgt für dieses Kapitel:

JUMBO BLOCK Rescue wird hier als **SSH-Client-System** dokumentiert.

Eine standardmäßig auf JUMBO BLOCK Rescue laufende SSH-Serverfunktion wird nicht behauptet.

### Grundsyntax

Eine einfache SSH-Verbindung wird aufgebaut mit:

```
ssh HOST
```

Soll ein bestimmter Benutzer verwendet werden:

```
ssh USER@HOST
```

Beispielprinzip:

```
ssh admin@server.example
```

### Benutzername

Der Benutzername bezeichnet das Benutzerkonto auf dem entfernten System.

Er muss nicht mit dem lokalen Benutzernamen des Rescue-Systems identisch sein.

Alternativ kann der Remote-Benutzer über die SSH-Konfiguration festgelegt werden.

### IP-Adresse statt Hostname

SSH kann auch direkt mit einer IP-Adresse verwendet werden:

```
ssh USER@IP-ADRESSE
```

Damit kann die DNS-Auflösung des Zielnamens für diesen Verbindungstest umgangen werden.

### DNS und SSH

Wird ein Hostname verwendet, muss dieser zunächst in eine erreichbare Adresse aufgelöst werden.

Bei Problemen kann deshalb mit:

**Dig**

geprüft werden, ob die Namensauflösung korrekt funktioniert.

### Standardport

SSH-Server verwenden üblicherweise TCP-Port:

**22**.

Ein Server kann jedoch auf einem anderen Port konfiguriert sein.

### Anderen Port verwenden

Mit:

`-p PORT`

wird ein abweichender Zielport angegeben.

Beispiel:

```
ssh -p 2222 USER@HOST
```

### SSH besteht aus mehreren Sicherheitsstufen

Beim Aufbau einer SSH-Verbindung müssen insbesondere zwei unterschiedliche Authentisierungsfragen getrennt werden:

**1. Ist der Server wirklich der erwartete Server?**

→ Host-Key-Prüfung.

**2. Darf der Benutzer auf diesen Server zugreifen?**

→ Benutzer-Authentisierung.

Diese beiden Vorgänge dürfen nicht miteinander verwechselt werden.

### Serverauthentisierung

SSH-Server besitzen kryptografische Host Keys.

Der Client verwendet diese Schlüssel, um die Identität des Servers über wiederholte Verbindungen hinweg zu überprüfen.

Der Host Key authentisiert dabei den Server gegenüber dem Client.

### Benutzer-Authentisierung

Anschließend muss sich der Benutzer gegenüber dem Server authentisieren.

Mögliche Verfahren hängen von der Serverkonfiguration ab.

Dazu gehören beispielsweise:

Public-Key-Authentisierung,

Passwortauthentisierung

und weitere vom Server unterstützte SSH-Authentisierungsverfahren.

### Host Key und Benutzer-Key sind unterschiedliche Schlüssel

Ein häufiger Denkfehler ist, beide Schlüsselarten gleichzusetzen.

**Host Key**

identifiziert den SSH-Server.

**User Key**

authentisiert einen Benutzer gegenüber dem SSH-Server.

Beide lösen unterschiedliche Aufgaben.

### Erste Verbindung

Bei der ersten Verbindung zu einem bisher unbekannten SSH-Host besitzt der Client normalerweise noch keinen gespeicherten Host Key für dieses Ziel.

Der Benutzer erhält dann abhängig von der Konfiguration Informationen zum neuen Host Key beziehungsweise dessen Fingerprint.

### Fingerprint prüfen

Vor der erstmaligen Annahme eines sicherheitskritischen Host Keys sollte dessen Fingerprint über einen vertrauenswürdigen unabhängigen Kanal überprüft werden.

Beispielsweise über:

lokale Serverkonsole,

Dokumentation,

Configuration Management,

Inventarsystem

oder eine andere bereits authentisierte Informationsquelle.

### Warum das wichtig ist

Wird bei der ersten Verbindung blind ein fremder Host Key akzeptiert, kann der Client nicht wissen, ob er tatsächlich mit dem erwarteten Server oder einer zwischengeschalteten Gegenstelle verbunden ist.

Verschlüsselung allein löst dieses Identitätsproblem nicht.

### `known_hosts`

Bekannte SSH-Host Keys werden typischerweise im Benutzerkontext in:

`~/.ssh/known_hosts`

gespeichert.

OpenSSH kann außerdem globale Known-Hosts-Dateien verwenden.

Die Host-Key-Datenbank ermöglicht den Vergleich bei späteren Verbindungen.

### Wiederholte Verbindung

Bei späteren Verbindungen kann SSH prüfen, ob der vom Server präsentierte Host Key zum gespeicherten Schlüssel passt.

Stimmt er überein, besitzt der Client eine Kontinuität zur zuvor akzeptierten Serveridentität.

### Host Key hat sich geändert

Ändert sich ein zuvor gespeicherter Host Key, erzeugt OpenSSH eine deutliche Warnung.

Eine solche Warnung darf nicht routinemäßig beseitigt werden.

### Legitime Ursachen einer Host-Key-Änderung

Eine Änderung kann beispielsweise auftreten nach:

Neuinstallation des Servers,

Austausch des Systems,

bewusster Regeneration der Host Keys,

Wiederherstellung aus einer anderen Konfiguration

oder Änderung der SSH-Infrastruktur.

### Sicherheitsrelevante Ursache

Ein unerwartet geänderter Host Key kann jedoch auch auf eine falsche Gegenstelle oder einen Man-in-the-Middle-Angriff hinweisen.

Deshalb gilt:

**Erst Ursache klären, dann Known-Hosts-Eintrag ändern.**

### Gespeicherten Host suchen

Mit:

```
ssh-keygen -F HOST
```

kann nach einem Host in einer Known-Hosts-Datei gesucht werden.

Diese Funktion funktioniert auch bei entsprechend gehashten Hostnamen.

### Alten Host-Key-Eintrag entfernen

Mit:

```
ssh-keygen -R HOST
```

können gespeicherte Schlüssel für einen Host aus `known_hosts` entfernt werden.

### Sicherheitsregel für `ssh-keygen -R`

Der Befehl sollte nicht einfach ausgeführt werden, um eine Host-Key-Warnung „wegzubekommen“.

Zuerst muss geklärt werden, warum sich der Host Key geändert hat.

Erst wenn die neue Serveridentität unabhängig bestätigt wurde, ist das Entfernen des veralteten Eintrags ein sinnvoller administrativer Schritt.

### `StrictHostKeyChecking`

Das Verhalten bei unbekannten beziehungsweise geänderten Host Keys wird unter anderem über:

`StrictHostKeyChecking`

gesteuert.

Diese Einstellung ist ein zentraler Bestandteil der SSH-Serveridentitätsprüfung.

### Host-Key-Prüfung nicht deaktivieren

Optionen, die Host-Key-Prüfungen pauschal umgehen, sollten nicht Bestandteil eines normalen Rescue-Workflows sein.

Gerade ein Rescue-System kann in unbekannten oder kompromittierten Netzwerken eingesetzt werden.

Die Serverauthentisierung ist dort besonders wichtig.

### Public-Key-Authentisierung

Bei der Public-Key-Authentisierung besitzt der Benutzer ein kryptografisches Schlüsselpaar:

**privater Schlüssel**

und:

**öffentlicher Schlüssel.**

Der Server kennt beziehungsweise autorisiert den öffentlichen Schlüssel.

Der Client beweist den Besitz des zugehörigen privaten Schlüssels.

### Privater Schlüssel

Der private Schlüssel ist geheim.

Er darf nicht:

unkontrolliert kopiert,

öffentlich gespeichert,

in Tickets eingefügt,

per ungeschützter Nachricht versendet

oder auf unbekannten Systemen abgelegt

werden.

### Öffentlicher Schlüssel

Der öffentliche Schlüssel darf an Systeme verteilt werden, auf denen der Benutzer damit autorisiert werden soll.

Er ermöglicht nicht unmittelbar die Rekonstruktion des privaten Schlüssels.

### Identity-Datei auswählen

Mit:

`-i`

kann eine bestimmte Identity-Datei für die Public-Key-Authentisierung ausgewählt werden.

Beispiel:

```
ssh -i ~/.ssh/id_ed25519 USER@HOST
```

OpenSSH unterstützt mehrere Identity-Dateien und kann diese auch hostbezogen konfigurieren.

### Passphrase für private Schlüssel

Private SSH-Schlüssel können mit einer Passphrase geschützt werden.

Dadurch ist der Besitz der Schlüsseldatei allein nicht automatisch ausreichend, um sie zu verwenden.

### Schlüssel ohne Passphrase

Ein privater Schlüssel ohne Passphrase kann automatisierte Nutzung vereinfachen.

Wird die Datei jedoch kopiert oder gestohlen, fehlt eine zusätzliche Schutzschicht.

Auf einem Rescue-System sollte privates Schlüsselmaterial deshalb besonders restriktiv behandelt werden.

## `ssh-agent`

OpenSSH kann private Schlüssel über:

`ssh-agent`

bereitstellen.

Der Agent hält Identitäten für SSH-Authentisierung verfügbar.

## `ssh-add`

Mit:

`ssh-add`

können Schlüssel einem laufenden Agenten hinzugefügt werden.

Mit:

`ssh-add -l`

werden die Fingerprints der aktuell im Agenten vorhandenen Identitäten angezeigt.

### Agentzustand beachten

Wurde ein Schlüssel in den Agenten geladen, kann er für weitere SSH-Verbindungen verfügbar bleiben.

Auf einem Rescue-System ist daher nach sicherheitskritischen Arbeiten zu berücksichtigen, welche Identitäten noch im Agenten geladen sind.

### Agent Forwarding

Mit:

```
ssh -A
```

kann die Verbindung zum lokalen Authentisierungsagenten an das entfernte System weitergereicht werden.

Dadurch muss der private Schlüssel selbst nicht auf dem Remote-System gespeichert werden.

### Agent Forwarding ist sicherheitskritisch

OpenSSH warnt ausdrücklich vor unbedachter Verwendung von Agent Forwarding.

Ein Benutzer mit ausreichenden Rechten auf dem entfernten System kann auf den weitergereichten Agent-Socket zugreifen und den Agenten für Authentisierungsoperationen missbrauchen.

Der Angreifer erhält dadurch zwar nicht unmittelbar das private Schlüsselmaterial, kann aber Signatur-/Authentisierungsoperationen mit den geladenen Identitäten auslösen.

### Rescue-Regel

Bei einem möglicherweise kompromittierten Zielsystem:

**kein Agent Forwarding verwenden.**

### Jump Host statt Agent Forwarding

Für den Zugriff auf ein internes Ziel über ein vorgeschaltetes SSH-System kann:

```
-J
```

verwendet werden.

Beispielprinzip:

```
ssh -J USER@JUMPHOST USER@ZIEL
```

OpenSSH verbindet sich dabei zunächst mit dem Jump Host und richtet darüber eine TCP-Weiterleitung zum endgültigen Ziel ein.

### Vorteil

Für diesen Anwendungsfall muss der lokale Authentisierungsagent nicht zwingend an den Jump Host weitergereicht werden.

Die OpenSSH-Dokumentation nennt Jump Hosts ausdrücklich als sicherere Alternative in Situationen, in denen sonst Agent Forwarding verwendet würde.

### Remote-Shell

Der Standardaufruf:

```
ssh USER@HOST
```

öffnet nach erfolgreicher Verbindung und Authentisierung normalerweise eine interaktive Remote-Sitzung.

Die dort eingegebenen Kommandos laufen auf dem entfernten System.

### Sehr wichtige Rescue-Regel

Nach dem Login muss jederzeit klar sein:

**Arbeite ich gerade lokal auf JUMBO BLOCK Rescue oder remote auf dem Zielsystem?**

Ein versehentlich auf dem falschen System ausgeführter Befehl kann erhebliche Schäden verursachen.

### Remote-Kommando

SSH kann auch direkt ein einzelnes Kommando auf dem entfernten System ausführen.

Grundprinzip:

```
ssh USER@HOST COMMAND
```

In diesem Fall wird das angegebene Kommando remote ausgeführt, anstatt eine normale Login-Shell zu starten.

### Beispielprinzip

```
ssh admin@server.example uname -a
```

Das Kommando:

```
uname -a
```

läuft auf dem entfernten Server.

Nicht auf dem Rescue-System.

### Vorteil

Dies eignet sich beispielsweise für gezielte Diagnoseabfragen, ohne eine längere interaktive Remote-Shell zu öffnen.

### Vorsicht mit Remote-Kommandos

Ein Remote-Kommando besitzt die Rechte des authentisierten Remote-Benutzers.

Mit `sudo` oder einem privilegierten Konto können entsprechend weitreichende Änderungen ausgelöst werden.

### Pseudo-Terminal

Mit:

`-t`

kann SSH die Zuweisung eines Pseudo-Terminals erzwingen.

Dies ist beispielsweise für bestimmte interaktive Remote-Kommandos erforderlich.

Mit:

`-T`

wird die Pseudo-Terminal-Zuweisung deaktiviert.

### IPv4 erzwingen

Mit:

`-4`

kann der SSH-Client auf IPv4 beschränkt werden.

Dies kann bei Diagnosefällen hilfreich sein, in denen IPv4 und IPv6 getrennt untersucht werden sollen.

### IPv6 erzwingen

Mit:

`-6`

wird entsprechend IPv6 verwendet.

### Verbose-Modus

Für die SSH-Diagnose ist:

`-v`

besonders wichtig.

Beispiel:

```
ssh -v USER@HOST
```

Mehrere `-v` erhöhen die Ausführlichkeit bis maximal drei Stufen:

`-vv`

`-vvv`.

### Was Verbose-Ausgabe hilft zu unterscheiden

Die Debugausgabe kann unter anderem helfen festzustellen:

welche Konfiguration verwendet wird,

welche Adresse kontaktiert wird,

wie weit der Verbindungsaufbau kommt,

welche Host-Key- beziehungsweise Authentisierungsschritte stattfinden

und warum eine Anmeldung möglicherweise scheitert.

### Debugausgabe kann sensible Informationen enthalten

Verbose-Logs sollten nicht unkontrolliert veröffentlicht werden.

Sie können Informationen enthalten über:

Benutzernamen,

Hostnamen,

interne Adressen,

Schlüsseldateinamen,

Authentisierungsmethoden

und Infrastrukturdetails.

### Effektive SSH-Konfiguration anzeigen

Mit:

```
ssh -G HOST
```

gibt OpenSSH die nach Auswertung der Konfiguration resultierenden Clientparameter aus und beendet sich anschließend.

### Warum `ssh -G` hilfreich ist

SSH-Konfiguration kann aus mehreren Quellen stammen.

Bei einem unerwarteten Verhalten lässt sich damit beispielsweise prüfen:

welcher Benutzer,

welcher Port,

welche Identity-Dateien,

welche Proxy-/Jump-Einstellungen

oder andere Parameter effektiv verwendet werden.

### Benutzerkonfiguration

Die benutzerspezifische SSH-Konfiguration liegt typischerweise unter:

`~/.ssh/config`

Dort können hostbezogene Einstellungen definiert werden.

### Beispielprinzip

Eine Konfiguration kann sinngemäß enthalten:

```
Host rescue-server
HostName 192.0.2.10
User admin
Port 22
IdentityFile ~/.ssh/id_ed25519
```

Anschließend genügt:

```
ssh rescue-server
```

### Vorteil

Wiederkehrende Verbindungen müssen nicht jedes Mal mit sämtlichen Parametern auf der Kommandozeile angegeben werden.

### Systemweite Konfiguration

OpenSSH berücksichtigt zusätzlich systemweite Clientkonfiguration.

Für eine genaue Diagnose sollte deshalb nicht angenommen werden, dass ausschließlich `~/.ssh/config` das Verhalten bestimmt.

`ssh -G HOST` ist gerade hierfür hilfreich.

### Dateiübertragung mit `scp`

Das OpenSSH-Clientpaket stellt auch:

`scp`

zur Verfügung.

Damit können Dateien über eine SSH-Verbindung übertragen werden.

### Moderne `scp`-Übertragung

Seit OpenSSH 9.0 verwendet `scp` standardmäßig das SFTP-Protokoll für Dateiübertragungen.

Damit gilt dies auch für die hier dokumentierte OpenSSH-10.0p1-Generation.

### Lokale Datei auf Remote-System kopieren

Grundprinzip:

```
scp DATEI USER@HOST:/ZIELPFAD/
```

### Remote-Datei lokal sichern

Grundprinzip:

```
scp USER@HOST:/PFAD/DATEI ./
```

### Verzeichnisse

Mit:

```
scp -r
```

können Verzeichnisbäume rekursiv übertragen werden.

### `scp` verwendet SSH-Sicherheit

`scp` verwendet die SSH-Verbindung und damit dieselbe grundlegende Authentisierung und Transportsicherheit wie eine SSH-Sitzung.

### Port bei `scp`

Bei `scp` wird ein abweichender SSH-Port mit:

```
-P
```

angegeben.

Dies unterscheidet sich von:

`ssh -p`.

Groß-/Kleinschreibung ist hier relevant.

## `sftp`

OpenSSH stellt außerdem den interaktiven SFTP-Client:

```
sftp
```

bereit.

SFTP ist ein Dateiübertragungsprotokoll über SSH.

### SFTP ist nicht FTP über TLS

SFTP darf nicht mit:

FTP,

FTPS

oder einem einfachen „verschlüsselten FTP“

verwechselt werden.

SFTP ist ein eigenes Protokoll, das über SSH transportiert wird.

### Rescue-Einsatz für Dateiübertragung

SSH-basierte Dateiübertragung eignet sich beispielsweise für:

Logs vom Zielsystem holen,

Konfigurationsdateien sichern,

Diagnoseergebnisse übertragen,

kleine Reparaturdateien bereitstellen

oder Daten auf ein autorisiertes Remote-System kopieren.

### Vor dem Upload auf ein beschädigtes System

Bei einem Rescue-Fall sollte zunächst geklärt werden, ob auf das Zielsystem geschrieben werden darf.

Ein Upload verändert den Zielzustand.

Für:

forensische Untersuchungen,

Beweissicherung

oder besonders sensible Incident-Response-Fälle

kann dies unerwünscht sein.

### Port Forwarding

SSH kann TCP-Verbindungen durch den verschlüsselten SSH-Kanal weiterleiten.

OpenSSH unterstützt unter anderem:

lokale,

entfernte

und dynamische Portweiterleitung.

### Local Forwarding

Mit:

-L

wird ein lokaler Port über die SSH-Verbindung zu einem Ziel auf der entfernten Seite weitergeleitet.

Grundprinzip:

```
ssh -L LOKALER_PORT:ZIELHOST:ZIELPORT USER@SSH-HOST
```

### Beispielanwendung

Ein internes Webinterface ist nur vom SSH-Server aus erreichbar.

Ein lokaler Tunnel kann dieses Interface kontrolliert über die SSH-Verbindung erreichbar machen.

Danach kann beispielsweise Chromium lokal auf den weitergeleiteten Port zugreifen.

### Remote Forwarding

Mit:

`-R`

wird auf der entfernten Seite ein Port beziehungsweise Socket bereitgestellt, dessen Verbindungen durch den SSH-Kanal zur lokalen Seite weitergeleitet werden.

### Sicherheitsrisiko bei Forwardings

Ein Forwarding kann Netzwerkgrenzen überbrücken.

Dadurch können Dienste erreichbar werden, die ohne den Tunnel nicht erreichbar wären.

Portweiterleitungen dürfen deshalb nur:

autorisiert,

bewusst,

mit kontrollierten Bind-Adressen

und für den notwendigen Zeitraum

eingerichtet werden.

### Local Forwarding bindet standardmäßig lokal

OpenSSH bindet lokale Portweiterleitungen standardmäßig an die Loopback-Adresse, sofern die Konfiguration dies nicht verändert.

Das reduziert unbeabsichtigte Erreichbarkeit durch andere Rechner.

### Dynamisches Forwarding

SSH unterstützt außerdem dynamische Portweiterleitung.

Dabei kann der SSH-Client als lokaler SOCKS-Proxy für Verbindungen über den SSH-Kanal dienen.

Auch dies kann Netzwerkgrenzen verändern und gehört nur in einen bewusst geplanten Rescue-Workflow.

`-N`

Bei reinen Forwarding-Verbindungen kann SSH so verwendet werden, dass kein Remote-Kommando ausgeführt wird.

Dies ist beispielsweise für einen ausschließlich als Tunnel verwendeten SSH-Kanal sinnvoll.

### SSH-Tunnel ist kein VPN-Ersatz für jede Situation

SSH-Portweiterleitungen transportieren gezielt bestimmte Verbindungen.

Sie ersetzen nicht automatisch eine vollständige VPN-Konfiguration.

### SSH und Nmap

Nmap kann prüfen, ob ein SSH-Dienst am autorisierten Ziel sichtbar ist.

Beispielsweise kann ein offener TCP-Port 22 auf einen möglichen SSH-Dienst hinweisen.

SSH übernimmt anschließend den eigentlichen authentisierten Verbindungsaufbau.

### SSH und Dig

Dig kann prüfen, ob der SSH-Hostname korrekt aufgelöst wird.

Wenn:

```
ssh server.example
```

fehlschlägt, aber:

```
ssh IP-ADRESSE
```

funktioniert, ist die Namensauflösung eine wichtige Untersuchungsrichtung.

### SSH und MTR

MTR kann bei entfernten Zielen helfen, Netzwerkpfad und Latenz zu untersuchen.

Ein SSH-Timeout muss nicht durch SSH selbst verursacht werden.

Das Problem kann bereits auf der Netzwerkebene liegen.

### SSH und GUFW

Eine lokale Firewall kann ausgehende oder eingehende Netzwerkkommunikation beeinflussen.

Bei unerwarteten SSH-Verbindungsproblemen sollte deshalb auch die Firewallkonfiguration berücksichtigt werden.

### SSH und TCPDump/Wireshark

TCPDump und Wireshark können den SSH-Verbindungsaufbau auf Paketebene untersuchen.

Die Nutzdaten einer korrekt verschlüsselten SSH-Sitzung sind dabei nicht einfach als Klartext sichtbar.

Dies ist gerade ein Zweck des Protokolls.

### SSH und Chromium

Über Local Port Forwarding kann ein Webdienst hinter einem SSH-System lokal verfügbar gemacht und anschließend mit Chromium untersucht werden.

Damit können beide Programme in einem Rescue-Workflow kombiniert werden.

### SSH und SSHFS

Das folgende Kapitel A.28 behandelt:

**SSHFS.**

SSHFS verwendet SSH/SFTP, um ein entferntes Dateisystem lokal einzubinden.

Die SSH-Sicherheitsgrundlagen dieses Kapitels gelten deshalb auch für SSHFS:

Host-Key-Prüfung,

Benutzerauthentisierung,

Schlüsselmanagement

und Netzwerkerreichbarkeit.

### Möglicherweise kompromittiertes Zielsystem

Remote-Rescue-Arbeiten haben eine besondere Sicherheitsdimension.

Ist das Zielsystem möglicherweise kompromittiert, darf ihm nicht automatisch vertraut werden.

### Keine privaten Schlüssel auf das Ziel kopieren

Ein privater SSH-Schlüssel des Rescue-Systems sollte nicht auf einen möglicherweise kompromittierten Remote-Host kopiert werden.

Sonst könnte ein Angreifer diesen Schlüssel übernehmen.

### Kein Agent Forwarding zum kompromittierten Ziel

Ebenso sollte Agent Forwarding nicht zu einem nicht vertrauenswürdigen System aktiviert werden.

Der private Schlüssel selbst wird zwar nicht übertragen, aber der Remote-Host kann unter den von OpenSSH beschriebenen Bedingungen den weitergeleiteten Agenten für Authentisierungsoperationen verwenden.

## Remote-Ausgaben sind nicht automatisch vertrauenswürdig

Wenn ein kompromittiertes System remote Befehle ausführt, stammen die Ergebnisse aus diesem möglicherweise manipulierten System.

Ein Angreifer mit ausreichender Kontrolle kann beispielsweise:

Programme verändern,

Ausgaben manipulieren,

Logs verändern

oder Prozesse verstecken.

SSH sichert die Verbindung zum Host.

SSH garantiert nicht die Integrität des Betriebssystems, das auf diesem Host läuft.

## Wichtige Unterscheidung

**SSH-Transport sicher**

bedeutet nicht:

**Remote-System vertrauenswürdig.**

## Typischer Rescue-Ablauf: Server erreichbar, SSH funktioniert nicht

**1. DNS prüfen**

```
dig HOST
```

**2. Netzwerkpfad prüfen**

MTR beziehungsweise geeignete Netzdiagnose.

**3. Port prüfen**

Nmap am autorisierten Ziel.

**4. SSH ausführlich starten**

```
ssh -vvv USER@HOST
```

**5. Fehlerstufe identifizieren**

Namensauflösung?

TCP-Verbindung?

Host Key?

Schlüsselaustausch?

Benutzerauthentisierung?

**6. Konfiguration prüfen**

```
ssh -G HOST
```

### Typischer Rescue-Ablauf: neuer Server

1. Zieladresse eindeutig feststellen
2. SSH-Verbindung starten
3. angezeigten Host-Key-Fingerprint nicht blind akzeptieren
4. Fingerprint über unabhängige vertrauenswürdige Quelle prüfen
5. erst danach Host Key übernehmen
6. Benutzer authentisieren

### Typischer Rescue-Ablauf: Host-Key-Warnung

1. Verbindung nicht einfach erzwingen
2. prüfen, ob Server neu installiert oder ersetzt wurde
3. aktuellen Host-Key-Fingerprint über unabhängigen Kanal beschaffen
4. alten und neuen Schlüssel vergleichen
5. nur bei bestätigter Änderung alten Known-Hosts-Eintrag entfernen

```
ssh-keygen -R HOST
```

6. neue Verbindung aufbauen und neuen Fingerprint prüfen

### Typischer Rescue-Ablauf: Datei vom Server sichern

1. Serveridentität prüfen
2. mit autorisiertem Benutzer verbinden
3. Quelldatei möglichst nicht verändern
4. Datei mit `scp` oder `sftp` auf ein separates Rescue-Ziel kopieren
5. bei kritischen Dateien anschließend Hash beziehungsweise Signatur dokumentieren

### Typischer Rescue-Ablauf: internes Webinterface

1. SSH-Zugriff auf autorisierten Jump-/Administrationshost herstellen
2. lokalen Portforward einrichten
3. nur benötigten lokalen Port binden
4. Chromium gegen den lokalen Forward verwenden
5. Tunnel nach Abschluss schließen

### Typischer Rescue-Ablauf: kompromittierungsverdächtiger Server

1. Zielidentität prüfen
2. dedizierte Rescue-Zugangsdaten verwenden

**3. kein Agent Forwarding**

**4. keine unnötigen privaten Schlüssel übertragen**

**5. Remote-Kommandos auf das erforderliche Minimum beschränken**

**6. Ergebnisse nicht ungeprüft als vertrauenswürdige Systemwahrheit behandeln**

**7. relevante Daten auf separates Analysemedium sichern**

## Grenzen

SSH repariert kein Netzwerk.

SSH ersetzt Nmap, MTR oder Dig nicht.

SSH garantiert nicht, dass das entfernte Betriebssystem vertrauenswürdig ist.

Eine verschlüsselte SSH-Verbindung schützt nicht vor einem bereits kompromittierten Remote-System.

Eine erfolgreiche Benutzeranmeldung beweist nicht allein, dass der Server der erwartete Server ist – dafür ist die Host-Key-Prüfung zuständig.

Eine erfolgreiche Host-Key-Prüfung beweist nicht, dass der angemeldete Benutzer autorisiert handeln darf – dafür ist die Benutzer-Authentisierung beziehungsweise Berechtigungsstruktur zuständig.

SSH-Portweiterleitungen können Netzwerkgrenzen verändern.

Agent Forwarding kann zusätzliche Sicherheitsrisiken erzeugen.

## Sicherheitshinweise

Host-Key-Warnungen niemals routinemäßig ignorieren.

Fingerprints neuer sicherheitskritischer Hosts unabhängig verifizieren.

`known_hosts` nicht pauschal löschen, um Verbindungsprobleme zu beseitigen.

Private SSH-Schlüssel vertraulich behandeln.

Private Schlüssel möglichst mit Passphrase schützen.

Keine privaten Schlüssel auf kompromittierungsverdächtige Zielsysteme kopieren.

Agent Forwarding nur einsetzen, wenn es wirklich erforderlich und das Ziel vertrauenswürdig ist.

Bei kompromittierungsverdächtigen Systemen Agent Forwarding vermeiden.

Remote-Kommandos vor Ausführung sorgfältig prüfen.

Immer unterscheiden, ob ein Kommando lokal oder remote ausgeführt wird.

Portforwardings nur autorisiert und mit minimal notwendiger Reichweite einrichten.

Verbose-Logs vor Weitergabe auf sensible Infrastrukturinformationen prüfen.

## Technische Einordnung in JUMBO BLOCK Rescue

SSH eröffnet im JUMBO BLOCK Rescue Handbuch den Bereich:

**Remote.**

Die Baseline belegt dafür den:

**OpenSSH Client 10.0p1.**

Sie belegt dagegen in der vorliegenden Paketliste keinen OpenSSH-Server.

Die technische Rolle von A.27 lautet deshalb:

**JUMBO BLOCK Rescue → sichere ausgehende Remote-Verbindung → autorisiertes Zielsystem.**

Ein typischer Diagnosepfad lautet:

**Dig**

Zielnamen prüfen.

↓

**MTR**

Netzwerkpfad untersuchen.

↓

**Nmap**

SSH-Port beziehungsweise Dienst prüfen.

↓

**SSH**

Serveridentität über Host Key prüfen.

↓

**Benutzer authentisieren**

Passwort, Public Key oder anderes zugelassenes Verfahren.

↓

**Remote-Diagnose**

Shell oder gezielte Kommandos.

↓

**SCP/SFTP**

benötigte Dateien kontrolliert übertragen.

↓

**SSHFS**

bei Bedarf im nächsten Kapitel ein entferntes Dateisystem lokal einbinden.

Die zentrale Sicherheitslogik von SSH besteht dabei aus zwei unabhängigen Vertrauensrichtungen:

**Host Key**

Der Client prüft:

**Bin ich mit dem erwarteten Server verbunden?**

und:

**Benutzer-Authentisierung**

Der Server prüft:

**Darf dieser Benutzer auf mich zugreifen?**

Für den Rescue-Einsatz gilt deshalb als wichtigste Regel:

**Eine verschlüsselte Verbindung ist erst dann wirklich hilfreich, wenn auch die Identität der Gegenstelle geprüft wurde.**

# A.28 SSHFS

**Area:** Remote
**Program:** SSHFS
**Baseline 20261001:** SSHFS 3.7.3
**Installed package:** `sshfs 3.7.3-1.2~deb13u1`
**FUSE Baseline:** `fuse3 3.17.2-3`, `libfuse3-4 3.17.2-3`
**Operation:** Command line
**Task:** Mount remote directories as a local file system via SSH/SFTP

## Purpose

SSHFS allows you to mount a directory on a remote system locally as a file system via SSH.

Technically, SSHFS uses the following for this:

**SSH File Transfer Protocol – SFTP**

and makes the remote files available via:

**FUSE – file system in user space**

available in the local directory tree.

This allows programs on JUMBO BLOCK Rescue to access a remote directory in a similar way to a locally mounted file system.

## Use in the rescue system

SSHFS is particularly suitable within JUMBO BLOCK Rescue for:

controlled access to files on a remote Linux/Unix system,

Mounting a remote directory as a local mount point,

Copying diagnostic and log files,

Backing up selected remote files to local disks,

Provisioning a remote backup destination,

read-only oriented examination of remote directory holdings

and access to files with local tools without having to transfer them individually beforehand using `scp` or `sftp`.

## SSHFS builds on SSH

SSHFS does not replace SSH.

It uses the SSH mechanisms already described in A.27.

This includes in particular:

encrypted connection,

Host key verification,

User authentication,

SSH configuration

and SFTP.

The basic safety rules from A.27 therefore also apply to SSHFS.

### SSHFS is not a separate server service.

SSHFS is the client side on JUMBO BLOCK Rescue.

The remote system must provide suitable SSH/SFTP access.

SSHFS itself does not turn JUMBO BLOCK Rescue into an SSH file server.

### SFTP instead of remote shell

SSHFS requires the SFTP subsystem of the remote SSH system for normal file system access.

It thus integrates a remote file repository.

This is different from:

```
ssh USER@HOST
```

with an interactive remote shell.

### Basic syntax

The documented SSHFS syntax is:

```
sshfs [user@]host:[dir] mountpoint [options]
```

A simple example principle is:

```
sshfs USER@HOST:/REMOTE/PFAD /LOKALER/MOUNTPUNKT
```

### Example

```
sshfs admin@server.example:/var/log ~/remote-logs
```

The remote directory then appears:

```
/var/log
```

under the local mount point:

```
~/remote-logs
```

### Local Mount Point

The local mount point must exist and be suitable for the user in use.

For example:

```
mkdir ~/remote-logs
```

Afterward:

```
sshfs admin@server.example:/var/log ~/remote-logs
```

According to the project documentation, SSHFS is normally run as a regular user and not as `root`; the mount point must therefore belong to the user.

### Omit username

If no remote user is specified, SSHFS uses the local username for the connection by default.

Therefore, in administrative rescue situations, the explicit form is often clearer:

```
USER@HOST
```

### Omit remote directory

If no remote directory is specified after the colon, the home directory of the remote user is used by default.

Example:

```
sshfs admin@server.example: ~/remote
```

### Entire root file system

An authorised user can – provided their remote permissions allow it – also specify a path such as:

```
/
```

Specify as a starting point.

Schematic example:

```
sshfs USER@HOST:/ /MOUNTPUNKT
```

However, this does not automatically mean that unrestricted access to all files on the remote system is granted.

The rights of the authenticated remote user remain crucial.

### Remote permissions remain decisive

SSHFS does not bypass permissions on the remote system.

Which operations are actually permitted on the remote file system depends on the permissions that the remote access used has.

### Important feature of local laws

SSHFS ignores local file permission checks on the FUSE side by default.

Crucial for the actual remote operation is what the remote server allows for the access data used.

### `default_permissions`

With:

```
-o default_permissions
```

Local authorisation checks can also be activated.

However, this does not replace the authorisation check of the remote server.

It complements them on the local page.

### By default, only for the mounting user

By default, only the user who mounted the SSHFS file system can access it.

This is a sensible safety limit for a rescue system.

### `allow_other`

FUSE can be accessed via:

```
allow_other
```

to allow access for additional local users.

This expands the local access circle.

This should only be used in a rescue operation if it is absolutely necessary.

### Basic rule

**Allow as few local users as possible to access the remote mount.**

### Host key verification

Since SSHFS uses SSH, the host key rules from A.27 apply.

The identity of the remote SSH server must be verified during the first connection.

For security-critical systems, a host key fingerprint should be verified by an independent, trusted source.

### Host key warning

A warning about a changed host key should not be ignored, even with SSHFS.

Possible causes include:

Server reinstalled,

Host keys deliberately changed,

System replaces

or a false or manipulated counterpart.

The stored host key state may only be adjusted after clarification.

### Using SSH options

SSHFS can pass numerous SSH options to the underlying SSH connection.

This allows, for example:

Port,

Identity file,

Keepalive settings

or other SSH connection parameters

to be configured.

### Different SSH port

A different port can be used, for example, with:

```
-o port=PORT
```

-o port=PORT

Example:

```
sshfs -o port=2222 USER@HOST:/PFAD /MOUNTPUNKT
```

### Use a specific private key

Since SSHFS uses SSH, a specific identity file can be selected via the corresponding SSH option.

Schematic example:

```
sshfs -o IdentityFile=/PFAD/ZUM/SCHLUESSEL USER@HOST:/PFAD
/MOUNTPUNKT
```

The security rules for private SSH keys from A.27 remain unchanged.

### Read-only mount

For rescue operations, the following is particularly important:

```
-o ro
```

This mounts the SSHFS file system in read-only mode.

Schematic example:

```
sshfs -o ro USER@HOST:/PFAD /MOUNTPUNKT
```

### Significance for JUMBO BLOCK Rescue

If a remote system is to be examined and no modification is required, the following should generally be considered:

**Mount in read-only mode.**

This reduces the risk of accidental file changes via the SSHFS mount.

### Read-only protects only this access path.

An SSHFS read-only mount does not make the remote system completely immutable.

Other users,

other SSH sessions,

local processes on the server

or other network services

They can still modify the remote files.

### No forensic snapshot

Therefore, an SSHFS mount is not a frozen image of a remote file system.

Files may change during the investigation.

This is especially important in the case of:

Logs,

Databases,

Mail spools,

temporary files

and running applications.

### Live system stays live

When a running server is examined via SSHFS, processes of the server can be:

write files

delete files

Rotate files,

Rename files

or modify metadata.

The local view via SSHFS can therefore change over time.

### Consequences for securing evidence

For strictly forensic security, an SSHFS mount alone is not sufficient.

SSHFS is primarily a remote file access tool.

A legally sound acquisition requires a properly controlled and documented security process.

### Copy files

After mounting, normal local file programs can access the mount point.

For example, files can be copied from the SSHFS mount to a local rescue disk.

### Important perspective

A local command such as:

```
cp ~/remote-logs/system.log /media/rescue/
```

In this example, SSHFS actually reads data from the remote server.

The source appears local, but is physically located remotely.

### Writing feels remote

The opposite direction is even more important.

A local command such as:

```
rm ~/remote/datei
```

can delete a file on the remote system when using a read-write mounted SSHFS file system.

### Rescue basic rule

**An SSHFS mount should never be confused with a harmless local working directory.**

Operations within the mount point can trigger real changes on the remote system.

### Graphical file managers

Since SSHFS provides the remote directory as a local file system, local file managers can also access the mount point.

This includes, for example:

Copy,

Open,

Rename

or delete

This is possible via a graphical interface, provided that mount mode and remote permissions allow it.

### Danger from graphical programs

Even graphical programs can generate unintended write operations.

For example, by:

Save,

Rename,

Delete

or creating additional files.

For purely analytical purposes, a read-only SSHFS mount is therefore particularly useful.

### Applying programs to remote files

Local programs can read files from an SSHFS mount.

This allows, for example, the following:

Viewing text files

Analyse logs,

Calculate hashes

or copy files to a local medium for further examination.

### The network remains part of file access.

Any access to a file that is not yet available locally requires communication with the remote system.

SSHFS makes a network file system appear local.

It does not technically make it a local storage device.

### performance

The performance depends on, among other things:

Network bandwidth

Latency,

SSH encryption,

Remote server,

File sizes,

Number of file operations

and SSHFS/FUSE behaviour.

A directory containing many small files may therefore behave differently than the transfer of a single large file.

### Connection interrupted

One important feature of SSHFS concerns connection interruptions.

The documentation indicates that file system operations may be blocked if the connection is interrupted.

Programs accessing the mount point may appear to "freeze" as a result.

### Caused

SSHFS basically follows the SSH behaviour without automatically enforced network timeouts.

If a connection is interrupted unnoticed, ongoing accesses can therefore wait.

### `ServerAliveInterval`

The SSHFS documentation recommends the following for such situations, for example:

```
-o ServerAliveInterval=15
```

This causes the underlying SSH connection to regularly send keepalive messages.

### Example

```
sshfs -o ServerAliveInterval=15 USER@HOST:/PFAD /MOUNTPUNKT
```

### `reconnect`

With:

```
-o reconnect
```

SSHFS can attempt to automatically restore an interrupted connection.

Example:

```
sshfs -o reconnect USER@HOST:/PFAD /MOUNTPUNKT
```

### combination

A typical robust connection pattern might look like this:

```
sshfs -o reconnect,ServerAliveInterval=15 USER@HOST:/PFAD
/MOUNTPUNKT
```

### Important limitation of `reconnect`

Files that were already open before the reconnect may produce errors after the reconnect and must be reopened.

### No protection against data loss during a shutdown.

An automatic reconnection does not mean that a currently running file operation will continue seamlessly and without loss.

The SSHFS documentation explicitly warns that data being read or written during an interruption may be lost.

### Importance for rescue copies

For important files, one should not rely solely on the assumption that the copy is complete after an unstable connection.

A subsequent integrity check is advisable.

### Hash check

For example, after a critical file transfer, a cryptographic hash of the local copy can be compared with a hash determined independently on the source.

It must be taken into account whether the source file could have been modified during the backup process.

## `delay_connect`

With:

```
-o delay_connect
```

The SSH connection is not established immediately upon mounting.

The connection is only established when the mount point is actually addressed for the first time.

## `sshfs_sync`

With:

```
-o sshfs_sync
```

Synchronous write operations can be enabled.

This can slow down processing.

Naturally, this write option plays no central role in a purely read-only rescue mount.

## `no_readahead`

With:

```
-o no_readahead
```

SSHFS only reads the data that is actually requested, instead of proactively reading additional data.

This may be relevant in specific diagnostic or network conditions.

## `sync_readdir`

With:

```
-o sync_readdir
```

The reading of directory contents is performed synchronously.

This option can also slow down processing.

### Caching

SSHFS uses caching mechanisms.

This means that, with a remote file system that is changing simultaneously, it must be taken into account that the local view does not correspond to a transactional real-time snapshot at every moment.

### SSHFS is not a cluster file system

SSHFS is not designed to provide complex consistency guarantees for multiple systems writing simultaneously.

For rescue work, it should be understood as a practical remote file access method.

### Remote file is modified during reading.

If a file on the server is modified while being read or copied via SSHFS, the resulting copy may not represent a consistent point in time.

Particularly problematic are:

active databases,

virtual machines,

Mail databases,

large logs

and other continuously written files.

### Database files

A running database should not be backed up solely by copying its active files over SSHFS.

Whether such a copy is consistent depends on the specific database and its backup procedure.

For production databases, the designated backup/snapshot procedures should be used.

### Unmount

After completing the work, the SSHFS file system must be properly unmounted.

The documented Linux syntax for FUSE3 is:

```
fusermount3 -u MOUNTPUNKT
```

### Baseline and `fusermount3`

The JUMBO BLOCK Rescue Baseline contains:

`fuse3 3.17.2-3`.

This confirms the FUSE3 basis of the installed SSHFS stack.

### Example

```
fusermount3 -u ~/remote-logs
```

After that, the remote directory is no longer mounted via this mount point.

### Before the unmount

Programs should no longer have any files open within the SSHFS mount.

For a clean workflow, a terminal should also no longer be located within the mount point with its current working directory.

### "Device or resource busy"

If a mount cannot be unmounted, it may still be in use by a process.

First, it should be determined which process is still using the mount.

Do not unmount immediately by force.

### Connection interruption and blocked processes

In the event of a sudden network loss, processes that access SSHFS files may become blocked.

The SSHFS documentation points out that even a forced lazy unmount with `fusermount -zu` does not necessarily terminate blocked read/write operations in such a situation.

### consequence

For longer-lasting rescue connections:

`ServerAliveInterval`

and, if applicable:

`reconnect`

more important than the assumption that a subsequent force-unmount will reliably eliminate every problem.

### Read-only Rescue Workflow

A conservative workflow is as follows:

**1. Clearly identify the remote target**

**2. Check SSH accessibility**

**3. Verify the host key fingerprint**

**4. Select a suitable remote user**

**5. Create a local mount point**

**6. Mount remote directory as read-only**

```
sshfs -o ro USER@HOST:/PFAD /MOUNTPUNKT
```

**7. Examine the required files or copy them to a separate rescue medium.**

**8. Check the integrity of critical copies**

**9. Move all programs away from the mount point**

**10. Carefully unhook the mount.**

```
fusermount3 -u /MOUNTPUNKT
```

### Workflow with an unstable network connection

If connection interruptions are likely:

```
sshfs -o ro,reconnect,ServerAliveInterval=15 USER@HOST:/PFAD
/MOUNTPUNKT
```

It should be noted that:

A reconnect does not guarantee a lossless continuation of a currently running file operation.

### Workflow: Backing up logs from a remote server

**1. Verify host identity**

**2. Mount remote log directory read-only**

**3. Copy the required logs to the local rescue medium.**

**4. Document or hash local copies**

**5. Cleanly disconnect the SSHFS mount.**

This allows the original files on the server to remain largely unchanged, unless other processes modify them.

### Workflow: Remote backup destination

SSHFS can also be used to mount a remote directory as a destination for recovered files.

Then the mount is intentionally read-write.

However, the following must be checked beforehand:

available storage space,

Remote permissions,

Network stability,

Server trustworthiness

and data protection requirements.

### Sensitive Rescue Data

When sensitive recovered data is transferred to a remote system via SSHFS, SSH protects the data during transport.

However, after being written, they are stored there according to the configuration of the target system.

### Transport encryption is not storage encryption.

SSH protects the network connection.

SSHFS does not guarantee that the data on the remote disk is stored encrypted.

If necessary, the target system itself must provide suitable storage encryption, or the files must be encrypted before transfer, for example using GnuPG.

### Combination with GnuPG

One possible workflow is:

**Create rescue files locally**

↓

**Encrypt with GnuPG for the authorised recipient**

↓

**Transfer encrypted files to remote storage via SSHFS**

This combines two different levels of protection:

**SSH**

Transport encryption.

**GnuPG**

File encryption.

### SSHFS and SCP

Both use SSH-based transmission, but they adhere to different operating models.

**SCP**

selectively copies files between two locations.

**SSHFS**

permanently integrates a remote directory into the local directory tree.

### When SCP is easier

It should simply:

a single file

or:

a clearly defined directory inventory

Since the transfer only happens once, `scp` can be simpler and clearer.

### When SSHFS is useful

SSHFS is particularly useful when:

Several local tools are to repeatedly access remote files,

Directories must be searched.

or a remote file tree should be available locally for the duration of an analysis.

### SSHFS and SFTP

SSHFS uses SFTP as its underlying file transfer protocol.

the OpenSSH client `sftp` provides an interactive file transfer interface.

SSHFS translates these remote file operations into a local file system model.

### SSHFS and Midnight Commander

The later documented:

**Midnight Commander**

can use an SSHFS mount like a local directory tree.

This allows remote files to be examined and copied via a file manager interface.

However, the following also applies to read-write mounts:

Deleting, renaming, and overwriting can trigger real remote operations.

### SSHFS and ClamAV

Files within an SSHFS mount can generally be read by local tools.

However, a virus scan via a remote mount generates network access and can read significant amounts of data via the SSH connection.

This must be consciously taken into account for large collections.

### SSHFS and hash calculation

Even a local hash command on a remote file reads its entire contents over the network.

For a 50 GB file, correspondingly large amounts of data must be transferred to create a complete hash.

"Locally mounted" does not mean "locally stored".

### SSHFS and network diagnostics

In case of problems, the tools already documented can be combined:

**Dig**

Check name resolution.

**MTR**

Examine network path.

**Nmap**

Check the SSH port at the authorised destination.

**SSH**

Test direct registration.

**SSHFS**

then establish file system access.

### Diagnostic principle

If SSHFS is not working, you should first check whether a normal SSH/SFTP connection to the same destination works.

This allows us to distinguish between:

basic SSH/network problem

and:

SSHFS/FUSE-specific problem.

### IPv6 address

For a numeric IPv6 address, the SSHFS syntax requires that the address be enclosed in square brackets.

This prevents ambiguities with the colon used in the SSHFS path syntax.

## `/etc/fstab`

SSHFS can also be integrated via `/etc/fstab`.

The documented file system type is:

`sshfs`

for compatibility reasons also:

`fuse.sshfs`

is supported.

### Rescue classification of `/etc/fstab`

For a temporary rescue system, manually and deliberately set up mounts are often more transparent than permanently configured remote mounts.

A permanent `/etc/fstab` configuration is therefore not a necessary part of a normal rescue workflow.

### Version 3.7.3

The installed baseline version 3.7.3 was released on May 26, 2022 and, according to the release history, contained minor bug fixes.

This manual refers exclusively to the version actually installed in Baseline 20261001.

### Version boundary

Later SSHFS versions may include additional options, security changes, and bug fixes.

These are not retrospectively documented as features of SSHFS 3.7.3.

### Limitations

SSHFS is not a local file system.

SSHFS is not a backup method in itself.

SSHFS does not create a consistent snapshot of a running remote system.

SSHFS does not bypass remote permissions.

SSHFS does not guarantee an uninterrupted network connection.

Reconnect does not guarantee a lossless continuation of already opened files.

SSHFS does not guarantee memory encryption on the remote system.

A read-only SSHFS mount does not prevent other processes from modifying the remote file system.

SSHFS does not replace forensic acquisition.

SSHFS does not make a potentially compromised remote system trustworthy.

### Safety instructions

Verify the host key fingerprint before security-critical initial connections.

Do not ignore host key warnings.

Protect your private SSH keys.

Do not copy unnecessary private keys to remote systems.

Do not use agent forwarding on systems suspected of being compromised.

For pure analysis, mount in read-only mode if possible.

Always remember that local file operations within the mount point can trigger remote operations.

Do not accidentally delete, rename, or overwrite any files.

Transfer sensitive Rescue data only to authorised remote destinations.

Do not confuse transport encryption with storage encryption.

If the connection is unstable, check the integrity of the copied data afterwards.

Cleanly separate the mounts after completion using `fusermount3 -u`.

## Role in JUMBO BLOCK Rescue

SSHFS covers the following area within JUMBO BLOCK Rescue:

**Remote.**

The two programs in this area complement each other directly:

**A.27 SSH**

Provides secure remote connection, authentication, remote shell, and file transfer.

**A.28 SSHFS**

uses SSH/SFTP to mount a remote directory tree as a local FUSE file system.

The typical technical process is as follows:

**Dig / MTR / Nmap**

Check network and accessibility.

↓

**SSH**

Verify host identity and user access.

↓

**SSHFS**

Mount the authorised remote directory locally.

↓

**Read-only**

Prevent write access during pure examination.

↓

**local rescue tools**

Examine files or copy them to a separate medium.

↓

**Integrity control**

Verify critical backups.

↓

**fusermount3**

Cleanly unmount the remote file system.

This creates a particularly useful rescue mechanism:

A remote file system can be made available to local tools without having to manually transfer all files individually first.

The crucial safety rule is:

**An SSHFS directory looks local – but the file operations performed within it affect a remote real system.**

Therefore, for pure rescue analysis:

**Authenticate the remote access unambiguously, mount it in read-only mode if possible, back up the necessary data, and then cleanly detach the mount.**

# A.29 HWInfo

**Area:** Hardware
**Program:** HWInfo
**Baseline 20261001:** HWInfo 21.82
**Installed package:** `hwinfo 21.82-1.1+b2`
**Associated library:** `libhd21t64 21.82-1.1+b2`
**Operation:** Command line
**Task:** Detect, inventory, and document hardware for diagnostic and support purposes

## Purpose

HWInfo is a hardware detection tool for Linux.

The program examines the hardware detectable by the running system and can generate a detailed overview from this.

The Debian packet description includes information on:

Hard drives and partitions,

CD-ROM and other drives,

Network cards

Graphics cards,

Monitors,

Cameras,

Mice,

Sound hardware,

BIOS,

CPU,

USB devices,

memory

and multiprocessor systems.

HWInfo can also generate a system overview log, which can then be used for diagnostic and support purposes, for example.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, HWInfo serves in particular to answer the question:

**What hardware does the started rescue system actually detect?**

Typical use cases include:

To record the hardware inventory of an unknown computer,

Identify CPU and memory

Detect data carriers

Capture partitions,

Examine the storage controller,

List PCI devices,

Identify USB devices,

Determine network hardware

Detect graphics hardware

Examine monitor information,

Capture BIOS-related information,

Identify sound hardware

and create a hardware report for further diagnosis.

## Hardware detection instead of health diagnosis

HWInfo is primarily a:

**Hardware Detection Tool.**

This must be distinguished from a diagnosis of condition or health.

HWInfo can, for example, detect and write to a data carrier.

However, for its SMART status:

**A.8 SMART Diagnosis**

responsible.

For NVMe-specific status information:

**A.9 NVMe Diagnosis.**

## Basic call

HWInfo is accessed via:

`hwinfo`

called up.

The man page describes a call without options as roughly equivalent to:

```
hwinfo --all --log=-
```

This performs a comprehensive hardware detection and outputs the results.

### Scope of the output

A complete HWInfo output can be very extensive.

For rescue operations, it is therefore often more sensible to first query only the relevant hardware class.

### Hardware classes

HWInfo supports targeted hardware queries according to the following scheme:

`hwinfo --HARDWARE_ITEM`

Documented hardware classes include, among others:

`--bios`

`--block`

`--cpu`

`--disk`

`--gfxcard`

`--memory`

`--monitor`

`--netcard`

`--network`

`--partition`

`--pci`

`--scsi`

`--sound`

`--storage-ctrl`

`--usb`

`--usb-ctrl`

--wifi

as well as numerous other classes.

### Combining multiple classes

Hardware classes can be specified multiple times.

This allows multiple relevant areas to be examined in a single call.

This is useful, for example, if:

Data storage devices,

Storage controller

and:

Partitions

should be considered together.

### Brief view

With:

`--short`

HWInfo generates a more compact summary.

This option is used in addition to a hardware class.

Example:

```
hwinfo --short --disk
```

### Quick system overview

For a concise overview, for example:

```
hwinfo --short --block
```

The HWInfo man page uses exactly this call as an example for an overview of block devices.

### CPU

With:

```
hwinfo --cpu
```

The detected processor information is examined.

This can provide a quick answer to the question when dealing with an unknown target system:

**Which CPU does the rescue system detect?**

### RAM

With:

```
hwinfo --memory
```

HWInfo can output memory information.

### Important distinction regarding RAM

Hardware detection is not a RAM stress test.

The fact that HWInfo detects RAM does not prove that all memory cells are functioning correctly.

The Baseline includes separate tools for more advanced memory diagnostics, such as `memtest86+`and `memtester`; however, these are not among the 35 programs in this Block A manual.

### Data carrier

With:

```
hwinfo --disk
```

Detected data carriers will be examined.

The man page explicitly cites this call as an example:

```
hwinfo --disk
```

### Examine individual data carriers

HWInfo supports:

```
--only DEVNAME
```

This allows the output to be restricted to suitable devices.

The corresponding hardware class must also be specified.

Documented example:

```
hwinfo --disk --only /dev/sdb
```

### Significance in a rescue situation

If multiple drives are present, a targeted query can help to more accurately identify a specific device.

This is especially important before operations that rely on device pathways such as:

```
/dev/sda
```

```
/dev/sdb
```

or:

```
/dev/nvme...
```

access.

### Never simply copy device names from a previous session

Linux device names can depend on the detected hardware configuration.

Therefore, the current allocation must be checked again before critical rescue operations.

HWInfo can provide an additional source of information.

### Block devices

With:

```
hwinfo --block
```

Block devices are being examined.

In short:

```
hwinfo --short --block
```

A compact overview is created.

### Partitions

With:

```
hwinfo --partition
```

It is possible to specifically query the partition hardware or structure detected by HWInfo.

### Storage controller

With:

```
hwinfo --storage-ctrl
```

Storage controllers are being examined.

This is particularly important for rescue operations when a data carrier:

does not appear

behind a controller

connected via RAID/storage hardware

or the connection between the data carrier and the operating system needs to be examined.

### SCSI

With:

```
hwinfo --scsi
```

The relevant device class can be examined.

It should be noted that the Linux storage layer can also represent devices via SCSI-like interfaces that do not physically correspond to a classic parallel SCSI system.

### RAID devices

The HWInfo man page explicitly states that RAID devices are not normally output.

With:

```
--listmd
```

They can also be included in the output.

### Meaning

Therefore, if software RAID structures are relevant:

```
--listmd
```

represent an important addition.

However, the actual RAID analysis or repair is not the task of HWInfo.

### PCI devices

With:

```
hwinfo --pci
```

Detected PCI devices will be examined.

This can be helpful, for example, in the following situations:

Storage controllers,

network adapters,

Graphics controllers

and other PCI/PCIe devices.

### USB devices

With:

```
hwinfo --usb
```

Detected USB devices will be queried.

This is particularly useful in a rescue environment because additional devices are often connected:

USB hard drives,

USB SSDs,

Flash drives,

USB network adapter,

Adapter housing

or other peripherals.

### USB controller

With:

```
hwinfo --usb-ctrl
```

USB controllers can be examined separately.

This allows us to distinguish between:

connected USB device

and:

The system's USB controller

differentiate.

### Network hardware

With:

```
hwinfo --netcard
```

Detected network adapters can be examined.

This helps, for example, with the question:

**Which network interfaces or network devices does the rescue system detect at the hardware level?**

### Network and Netcard

HWInfo recognizes hardware classes such as:

`netcard`

as well as:

`network`.

The output should therefore be selected specifically according to the respective diagnostic purpose.

### Wi-Fi

With:

```
hwinfo --wlan
```

Specifically, WLAN-related hardware can be examined.

### Hardware detected, but network still doesn't work.

The fact that a network adapter is detected by HWInfo does not prove that:

an IP address is present,

the network is reachable,

DNS is working

or an internet connection exists.

The network tools from A.17 to A.24 are responsible for this.

### graphics card

With:

```
hwinfo --gfxcard
```

The detected graphics hardware can be examined.

This can happen, for example, in the following cases:

Display problems,

incorrect resolution,

multiple graphics adapters

or driver problems

provide initial clues.

### monitor

With:

```
hwinfo --monitor
```

Monitor information is being examined.

HWInfo also has special mechanisms for monitor detection.

### BIOS

With:

```
hwinfo --bios
```

BIOS-related information can be queried.

For detailed SMBIOS/DMI structures, JUMBO BLOCK Rescue also includes:

**A.32 DMI Decode – dmidecode.**

### Sound

With:

```
hwinfo --sound
```

Detected sound hardware can be examined.

### camera

HWInfo also supports the hardware class:

`camera`.

### Bluetooth

The documented hardware classes include:

`bluetooth`.

### Keyboard and mouse

HWInfo supports, among other things:

`keyboard`

and:

`mouse`.

### Printers and scanners

Also:

`printer`

and:

`scanner`

are among the documented hardware classes.

This means HWInfo goes significantly beyond pure PC core components.

## `--all`

With:

```
hwinfo --all
```

A comprehensive hardware detection process is triggered.

## `--reallyall`

Additionally, the hardware class exists:

`reallyall`.

It is part of the documented list of available hardware items.

For normal rescue tasks, however, a targeted query is usually preferable, as complete outputs can be very extensive.

### Log output

With:

```
--log FILE
```

HWInfo writes log information to a file.

Additionally, a hardware class must be specified in order for the corresponding hardware detection to be triggered.

Example principle:

```
hwinfo --all --log hwinfo.log
```

### Using a hardware log

A hardware log can, for example:

handed over to support

added to a rescue process,

before and after a hardware change compared

or saved for later analysis

become.

### Data protection of the log

A complete hardware log can contain identifying technical information.

Depending on the detected hardware, this could include, for example:

Device names,

Manufacturer information

Model information,

Hardware identifiers,

Network interface information

and further system details.

Therefore, such a log should be checked before being made public.

### Save configuration

HWInfo supports:

```
--save-config SPEC
```

This allows you to configure a specific device or all devices under:

```
/var/lib/hardware
```

be saved.

This option must be used in conjunction with hardware detection.

### View saved configuration

With:

`--show-config UDI`

Previously saved configuration data for a corresponding UDI can be displayed.

### Compare device assignments

HWInfo also owns:

`--map`

If disk names have changed, for example after a kernel change, this information can be used to generate an assignment.

The prerequisite is that you have previously:

`--save-config`

The corresponding configuration data was saved.

### Importance for a live rescue system

The functions:

`--save-config`

and:

`--map`

require a previously stored reference state.

This reference state typically does not yet exist when a rescue system is started for the first time on an unknown computer.

For a spontaneous initial diagnosis, normal hardware queries are therefore more important.

### Debug output

With:

`--debug N`

A debug level can be set.

According to the man page, debug information is output to the log. If a log file is specified, an appropriate debug level is automatically set.

## `--verbose`

The option:

`--verbose`

According to the man page, verbosity is only increased when used in conjunction with:

`--map`.

Therefore, it should not be understood as a general "show all hardware details" switch.

### Hardware database

HWInfo uses a hardware database.

The man page mentions, among other things:

`/var/lib/hardware/hd.ids`

as an external hardware database.

### Output database

With:

`--dump-db N`

The hardware database can be output.

This refers to:

`0`

the external database

and:

`1`

the internal database.

### View version

With:

`hwinfo --version`

The version of the underlying libhd is output.

In the JUMBO BLOCK Rescue Baseline, `hwinfo`and libhd21t64 are both in version:

**21.82-1.1+b2**

available.

### Root privileges

Some hardware information may require privileged access, depending on the device, kernel, and access rights.

If a normal user query does not provide all the expected information, targeted privileged diagnostics may be necessary.

### Do not use root access indiscriminately.

For a purely hardware overview, not every command should automatically be executed with root privileges.

The principle is:

**only as many rights as are necessary for the respective diagnosis.**

### Detection depends on the running kernel.

HWInfo operates from within the running Rescue system.

Therefore, the visible hardware also depends on what:

Kernel,

Driver,

Firmware

and hardware platform

make it available to the system.

### "Not detected by HWInfo" does not necessarily mean "not present".

The HWInfo man page explicitly mentions the following limitation:

**Not all hardware can be detected.**

Therefore, a missing entry is not absolute proof that the hardware in question does not physically exist.

### Possible causes of lack of detection

For example, a device can:

Not recognized by the kernel,

not addressed correctly due to lack of support

hidden behind another hardware layer

or not fully visible for other technical reasons

be.

The specific cause needs to be investigated using further diagnostic tools.

### HWInfo does not modify partitions.

HWInfo's normal hardware probe functions are used for detection and output.

HWInfo is not a partitioning tool.

For example, for partition changes:

**A.6 GParted**

responsible.

### HWInfo does not repair data carriers

HWInfo can detect data carriers.

However, it does not repair:

Partition tables,

File systems,

SMART errors

no physical data carrier damage.

### Typical rescue scenario: unknown computer

A useful initial overview could be:

```
hwinfo --short --block
hwinfo --short --cpu
hwinfo --short --netcard
hwinfo --short --gfxcard
```

This first determines which essential components the rescue system recognizes.

### Typical rescue case: Hard drive missing

A data carrier is expected, but does not appear as planned.

The diagnostic process could then be, for example:

**1. Check block devices**

```
hwinfo --short --block
```

**2. Check the data carrier**

```
hwinfo --disk
```

**3. Check the storage controller**

```
hwinfo --storage-ctrl
```

**4. Check PCI hardware**

```
hwinfo --pci
```

**5. Check USB hardware when using a USB connection**

```
hwinfo --usb
```

**6. Then use specialized storage tools**

### Typical rescue scenario: USB drive is not recognized

**1. Detect USB devices**

```
hwinfo --usb
```

**2. Check the USB controller**

```
hwinfo --usb-ctrl
```

**3. Check block devices**

```
hwinfo --short --block
```

**4. Next, examine the data carrier or file system level.**

This allows the diagnosis to proceed step by step from the hardware connection to the memory structure.

### Typical rescue scenario: Network not working

**1. Identify network hardware**

```
hwinfo --netcard
```

**2. Check WLAN hardware if necessary.**

```
hwinfo --wlan
```

**3. Next, examine the interface and IP configuration.**

**4. Then continue working with Ping, Dig, MTR or other network tools.**

HWInfo primarily answers the hardware question.

### Typical rescue case: incorrect screen resolution

**1. Check graphics hardware**

```
hwinfo --gfxcard
```

**2. Check the monitor**

```
hwinfo --monitor
```

**3. Next, examine the driver, kernel, and display configuration.**

### Typical Rescue case: Support report

A comprehensive hardware report can be generated, for example, by a targeted combination of hardware classes or a complete data collection.

Any sensitive or identifying information contained within must be checked before it is shared.

### HWInfo and Inxi

The following program:

**A.30 Inxi**

It also generates extensive system information.

The tools partially overlap, but have different display and query models.

HWInfo is specifically designed for detailed hardware identification by device class.

## HWInfo and LSHW

### A.31 LSHW

Represents hardware information in a hierarchical hardware structure.

In contrast, HWInfo works extensively with specifically selectable hardware classes.

Both tools can complement each other in diagnosis.

## HWInfo and dmidecode

### A.32 dmidecode

It reads SMBIOS/DMI structures and thus displays hardware information provided by the firmware.

HWInfo pursues broader hardware detection.

## HWInfo and lm-sensors

### A.33 lm-sensors

focuses on sensor values such as temperatures, voltages and fan data, provided the hardware provides and supports corresponding sensors.

HWInfo is not a replacement for this ongoing sensor monitoring.

## HWInfo and SMART

### HWInfo

answered:

**Which storage medium is recognized?**

### SMART Diagnostics

answered:

**What status and error data does a supported storage device report?**

These questions are separate.

## HWInfo and NVMe diagnostics

The following applies analogously:

HWInfo can make NVMe-related hardware visible in the system context.

`nvme-cli`is the specialized tool for NVMe controllers, namespaces, SMART/health information, error logs and other NVMe functions.

## Hardware detection before repair

Before any critical rescue operation, the following should be clearly established:

which device is the target

which device is the source medium

and:

which device is the backup medium.

HWInfo can provide an additional, independent hardware view for this purpose.

### Don't guess the device name

When dealing with multiple drives of similar size, a decision should never be made based solely on assumption:

" `/dev/sda`will already be the defective disk."

Before any writing process can take place, the device identity and role must be clearly verified.

### Especially critical with ddrescue

With GNU ddrescue, a mix-up of source and destination can overwrite the wrong medium.

HWInfo can therefore be used together with other data carrier tools for unambiguous identification before the actual recovery process.

### Hardware report before expansion

Before physically replacing a component, an HWInfo report can document the detected initial state.

After the change, re-capturing the data can help with comparison.

### Live system as an advantage

Because JUMBO BLOCK Rescue is started independently of the installed target operating system, HWInfo can also collect hardware information even if the installed operating system:

no longer starts

is damaged

or its own driver/system configuration is faulty.

### Live system as a boundary

At the same time, HWInfo views the hardware from the perspective of:

**JUMBO BLOCK Rescue Kernels and its drivers.**

The result therefore does not necessarily have to completely match the view of another operating system.

## Limitations

HWInfo detects hardware but does not repair it.

HWInfo is not a SMART diagnostic program.

HWInfo is not a specialized NVMe tool.

HWInfo is not a RAM stress test.

HWInfo is not a CPU stress test.

HWInfo is not a sensor monitoring program.

HWInfo is not a partitioning tool.

HWInfo is not a file system repair tool.

HWInfo does not guarantee complete detection of all existing hardware.

A recognized data carrier is not automatically healthy.

A detected network device does not automatically mean that network communication is working.

A recognized memory area does not automatically mean error-free RAM.

## Safety instructions

Verify device identities before performing critical data storage operations.

Do not confuse hardware detection with hardware health.

Do not perform unnecessary full scans or stress tests on physically conspicuous data carriers.

Use SMART or NVMe diagnostics for disk health.

If the data is unstable, it is best to continue working on a copy using GNU ddrescue.

Check hardware logs for sensitive system information before forwarding them externally.

Only use root privileges if they are necessary for the specific diagnosis.

Do not hastily conclude that the hardware is not physically present simply because an entry is missing in HWInfo.

## Technical classification in JUMBO BLOCK Rescue

HWInfo opens the following section in the JUMBO BLOCK Rescue manual:

**Hardware.**

This area consists of:

**A.29 HWInfo**

Broad hardware detection by device class.

**A.30 Inxi**

compact to detailed system overview.

**A.31 LSHW**

hierarchical hardware inventory.

**A.32 DMI Decode**

SMBIOS/DMI information.

**A.33 Sensors**

Temperature, voltage and fan sensors.

A typical hardware diagnostic procedure might look like this:

**HWInfo**

What hardware does the rescue system recognize?

↓

**Inxi**

What does the compact overall configuration look like?

↓

**LSHW**

How is the hardware structured hierarchically?

↓

**dmidecode**

What system, board, BIOS and memory information does SMBIOS/DMI provide?

↓

**lm-sensors**

Which supported sensor values are currently being reported?

For storage problems, the following steps may follow:

**SMART / NVMe Diagnostics**

↓

**GNU ddrescue**

↓

**File system and data recovery tools.**

The central role of HWInfo is therefore:

**Make the hardware visible and identifiable before working on it.**

Especially in rescue operations, this inventory is crucial because a repair or data recovery can only be reliably planned once it is clearly known:

**What hardware does the system actually see?**

# A.30 Inxi

**Area:** Hardware
**Program:** Inxi
**Baseline 20261001:** Inxi 3.3.38
**Installed package:** `inxi 3.3.38-1-1`
**Operation:** Command line
**Task:** Output consolidated system, hardware, and configuration information for diagnostics and support

## Purpose

Inxi is a command-line tool for outputting extensive system information.

It summarizes information from different areas of the running system in a uniform and easily readable presentation.

These include, among others:

System and kernel,

computer or motherboard,

CPU,

RAM,

Graphic,

Audio,

Network,

Data storage devices,

Partitions,

RAID

USB,

Battery,

Sensors,

driver

and other system information.

Inxi is also explicitly used as a diagnostic tool for technical support to quickly capture the system configuration of a computer.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, Inxi is particularly suitable for:

a quick overview of an unknown system,

Documentation of the configuration detected by the rescue system,

Identification of CPU, graphics, network and storage,

Determining the kernel drivers used,

Overview of storage media and partitions,

Capture of RAM information,

Display of supported sensor information,

Diagnosis of graphics and network problems

and the creation of a system report suitable for support purposes.

## Difference to HWInfo

A.29 HWInfo and Inxi partially overlap, but pursue different priorities.

**HWInfo**

is strongly focused on targeted hardware detection by device class.

**Inxi**

creates a coherent system report that displays hardware, drivers, kernel, desktop environment, and other system information together.

HWInfo is more likely to answer:

**What hardware does the system recognize?**

Inxi also answers:

**What does the current system configuration look like as a whole?**

## Basic call

The simplest call is:

```
inxi
```

Without any further options, Inxi generates a short system report.

## Base report

With:

```
inxi -b
```

A basic report will be generated.

This contains, among other things, information on:

System,

Machine,

Battery, if available

CPU,

Graphic,

network devices,

Data storage devices

and general system information.

### Importance for Rescue

`inxi -b`

This is a suitable starting point if a compact overview is needed initially, without immediately generating a very comprehensive report.

### Extended Report

The 3.3.x generation supports a more comprehensive overall report.

Traditionally, the following were often used for this purpose:

`inxi -F`

used.

The newer Inxi documentation now marks `-F` as obsolete and uses:

`-e`

or rather:

`--expanded`

for the extended report.

For the frozen baseline, the crucial factor is which syntax the installed version 3.3.38 actually provides.

### Levels of detail with `-v`

Inxi supports graded verbosity levels:

`-v 0`

until:

`-v 8`.

As the level increases, additional information areas and levels of detail are activated.

**`-v 0`**

It's equivalent to a simple report.

**`-v 1`**

The report expands to include, among other things:

System,

basic CPU data,

Graphic,

basic data carrier data

and general information.

**`-v 2`**

Includes, among other things:

Machine information

Battery, if available,

and network devices.

This level corresponds to the basic report:

`inxi -b`.

### Higher levels of detail

With increasing verbosity, the following may be added, among other things:

more complete CPU data,

Network information

Data storage devices,

Partitions,

RAM,

Audio,

Bluetooth

RAID

Swap,

Sensors,

USB,

CPU flags

and other administrative information.

### Rescue Rule

Do not automatically generate the maximum possible report.

Better is:

**First a concise overview → then targeted, more detailed information.**

This simplifies the evaluation and reduces unnecessarily extensive diagnostic outputs.

### CPU

With:

```
inxi -C
```

CPU information is output.

Depending on the available data, this may include, among other things:

CPU model,

Topology,

kernels,

Threads,

Cache information

and tempo information.

### CPU flags

With:

```
inxi -f
```

CPU flags or CPU features can be displayed in more detail.

This issue can become very extensive.

It is not always necessary for a normal hardware inventory.

### CPU detection is not a CPU test.

Inxi can display CPU properties.

However, it does not perform a stress test on the CPU.

A correctly identified CPU is not automatically demonstrably error-free.

### machine

With:

```
inxi -M
```

Machine information is output.

Depending on the platform, this may include information such as:

System,

Mainboard

and firmware/BIOS context

belong.

### DMI demarcation

JUMBO BLOCK Rescue also offers the following for targeted analysis of the SMBIOS/DMI tables:

#### A.32 dmidecode

available.

Inxi, on the other hand, compiles such information into a larger system report.

### RAM

With:

```
inxi -m
```

More detailed storage information is requested.

The information that is actually available depends on the system and the accessible data sources.

### RAM detection is not a RAM test.

Inxi displays information about RAM.

However, it does not check all memory cells for errors.

A RAM module that is recognized can still be defective.

### graphic

With:

```
inxi -G
```

Graphic information is displayed.

Inxi can provide information about, among other things:

Graphics devices,

Driver,

Display servers or Wayland components,

Resolutions

and available graphics APIs

merge.

### Rescue operation graphic

This is especially helpful in the following situations:

black screen,

incorrect resolution,

Driver problems,

multiple GPUs,

Wayland/X11 questions

or unknown graphics hardware.

### Audio

With:

```
inxi -A
```

Audio information is output.

This includes detected audio devices and associated drivers; depending on the system, active audio APIs and sound servers may also appear.

### Network devices

With:

```
inxi -N
```

Network devices are displayed.

This is primarily a hardware/driver perspective.

### Extended network information

With:

```
inxi -n
```

Further network information will be displayed.

### IP addresses

Inxi can also display IP-related information.

This information can be useful for troubleshooting, but it may be relevant to data protection or security when publishing a report.

### Data protection

Inxi therefore has special filter functions for support reports.

This is one of the most important features of the program for JUMBO BLOCK Rescue.

### `-z` – filter sensitive information

With:

```
inxi -z
```

or rather:

```
--filter
```

Sensitive information is filtered.

The Inxi documentation specifically mentions:

IP addresses,

MAC addresses,

Serial numbers,

Local information,

Usernames in the home path

and hostnames.

### Example

For example, a comprehensive report with additional filtering can be generated for a support report.

Basic principle:

```
inxi -Fz
```

or, with appropriate syntax:

```
inxi -ez
```

### Central Support Rule

**Filter sensitive information before forwarding an Inxi report and still double-check the finished report.**

Automatic filtering does not replace a final visual inspection.

### `-Z`

Inxi also has an option to override certain automatic filters.

This can be helpful for internal diagnosis, but should not be used carelessly for publicly distributed reports.

### Data carrier

With:

```
inxi -D
```

Data carrier information is displayed.

This includes basic information about detected drives and available storage.

### Data carrier information is not a complete condition diagnosis.

Inxi can display information about a drive.

However, for a targeted health diagnosis:

**A.8 SMART Diagnosis**

and:

**A.9 NVMe Diagnosis**

the specialized tools.

### Partitions

Inxi has various options for displaying partitions and mount states.

This allows a system report to include information on:

System partitions,

mounted partitions

and other storage structures

add to.

### RAID

With:

```
inxi -R
```

RAID-related information can be displayed, provided that corresponding structures are present and identifiable.

### RAID output is not RAID repair.

Inxi can compile RAID information.

However, it is not a tool for:

Generate,

Repair,

Resynchronize

or reconfigure

a RAID array.

### USB

With:

```
inxi -J
```

USB data is output.

This allows USB hubs and connected USB devices to be displayed.

### Importance for Rescue

This is especially useful in the following situations:

external storage devices,

USB-SATA adapters,

USB NVMe enclosures,

network adapters,

keyboards,

mice

and other rescue peripherals.

### Bluetooth

With:

```
inxi -E
```

Bluetooth information can be displayed.

Depending on available system support, this includes Bluetooth devices, drivers, and status information.

### battery

With:

```
inxi -B
```

Battery information will be displayed if a battery is detected.

This can be important for notebooks in rescue mode.

### Rescue rule for notebooks

A while ago:

Data recovery,

Copying operations,

Image creation

or data carrier checks

A stable power supply should be ensured.

A battery indicator does not replace ensuring a reliable power supply.

### Sensors

With:

`inxi -s`

Sensor information can be output.

Inxi can combine data from available system sources or sensor tools for this purpose.

### Distinction from A.33

For the actual sensor query and configuration:

**A.33 lm-sensors**

the specialized tool.

In contrast, Inxi integrates sensor information into a system report.

### Additional details with `-x`

Inxi has three additional levels of detail:

`-x`

`-xx`

`-xxx`.

Depending on the specific information area, these add further data.

### Example principle

`inxi -Cxxx`

provides significantly more CPU details than:

`inxi -C`.

Similarly, the additional levels of detail can be combined with other categories.

### More details may contain sensitive data

Higher levels of detail may reveal additional information, such as:

Serial numbers,

Device identifiers,

Bus information

or other technical identifiers.

Filtering is therefore particularly important before forwarding such a report.

### `-a` – administrative details

With:

`-a`

or rather:

`--admin`

Additional administrative system information will be activated.

The current documentation explicitly describes this data as extended information for system administrators.

### Meaning

An administrative report can be valuable when dealing with complex problems.

However, it is more comprehensive and potentially more sensitive than a normal baseline report.

### Do not automatically use `-a`

For the initial Rescue inventory, the following is usually sufficient:

`inxi -b`

or a specific category.

`The -a option` should be used if the additional technical details are actually needed.

### Driver information

One of Inxi's key strengths is its ability to link hardware information with the drivers used.

This can reveal, for example:

which network device was detected

and:

which driver is used for this.

### Hardware detected, but incorrect driver

Especially in:

Graphic,

Wi-Fi

Ethernet

Audio

or other PCI/USB devices

This combination can be crucial.

Simply knowing which device is present is not enough to solve a driver problem.

### Kernel

Inxi provides information about the running kernel.

In JUMBO BLOCK Rescue mode, this refers to the kernel of the started rescue system.

### A very important distinction

When examining a damaged installed Linux system with JUMBO BLOCK Rescue, Inxi basically describes:

**the currently running rescue system.**

It does not automatically describe the kernel, drivers, or desktop of the unbooted operating system on the hard drive.

### Example

If the installed target system has an old kernel, but JUMBO BLOCK Rescue was started with a newer kernel, Inxi will display the following:

**Rescue kernel.**

This must be taken into account when interpreting a report.

### The same applies to drivers.

The active drivers shown by Inxi belong to the running Rescue environment.

They do not prove that the installed target operating system has or uses the same drivers.

### Desktop environment

Inxi can also output information about the currently running desktop/window manager environment.

These details also refer to JUMBO BLOCK Rescue during live operation.

### Operating system information

The corresponding system information also refers to the active environment.

Therefore, an Inxi report from JUMBO BLOCK Rescue must not be referred to as a report of the operating system installed on the hard drive without being explicitly labeled as such.

### Rescue basic rule

Each saved Inxi report should clearly document the following:

**This report was generated from the ongoing JUMBO BLOCK Rescue environment.**

### Version without colors

Color codes may be undesirable for logs and files.

Inxi supports:

`-c 0`

The output without color codes.

### Benefits for saved reports

For a purely text-based report, a colorless version is often easier:

to save

to search

to compare

and to process further.

### Redirect output

As with other command-line programs, the output can be written to a file via the shell.

Example principle:

```
inxi -b -c 0 > inxi-report.txt
```

### Filtered support report

Example principle:

```
inxi -Fz -c 0 > inxi-support.txt
```

The file should then be checked before being passed on.

### Do not publish report blindly

Even a report filtered with `-z` can contain technical information about the system.

These include, for example:

Hardware models,

Kernel version,

Driver,

Storage sizes

and system architecture.

The filter function reduces sensitive identifiers, but does not make the report information-free.

### Output for internal documentation

For purely internal documentation, additional details may be useful.

Then, for example:

`-x`

`-xx`

or:

`-xxx`

to be added.

The specific level of detail should depend on the diagnostic purpose.

### Typical rescue process: initial system overview

**1. Baseline Report**

`inxi -b`

**2. Identify conspicuous areas**

CPU?

R.A.M?

Storage?

Graphic?

Network?

**3. Examine the relevant category in more detail**

for example:

`inxi -C`

`inxi -m`

`inxi -D`

```
inxi -G
```

```
inxi -N
```

**4. Increase the level of detail if necessary.**

```
-x
```

```
-xx
```

```
-xxx
```

**5. Use a specialized tool**

HWInfo,

SMART

NVMe,

dmidecode,

lm-sensors

or network diagnostics.

## Typical rescue scenario: Network not working

**1.**

```
inxi -N
```

Examine network hardware and drivers.

**2.**

Check extended network information.

**3.**

Then proceed with specialized network tools.

For example:

Dig,

MTR

Nmap,

ARP Scan

TCPDump

or Wireshark.

## Typical rescue case: Graphics problem

**1.**

```
inxi -G
```

**2.**

if needed:

```
inxi -Gx
```

or a higher level of detail.

**3.**

Compare hardware and active drivers.

**4.**

Supplement with HWInfo if necessary.

### Typical rescue case: Storage

**1.**

```
inxi -D
```

**2.**

Examine partitions and mount states.

**3.**

Use HWInfo to add the device identity.

**4.**

Check SMART or NVMe health data.

**5.**

Avoid creating unnecessary stress in cases of suspected physical damage.

**6.**

If necessary, first create an image using GNU ddrescue.

### Typical rescue scenario: Hardware report for support

**1.** Determine the required level of detail.

**2.** Activate filtering of sensitive data.

**3.** Disable color output for text files.

**4.** Save report.

**5.** Manually check the report for sensitive information.

**6.** Only then pass it on.

### Typical rescue case: Notebook

A report may contain information on:

CPU,

Graphic,

Network,

Battery,

Storage

and sensors

merge.

This quickly creates a usable technical overview of the currently running system.

## Inxi and HWInfo

**HWInfo**

Provides detailed hardware detection by device class.

**Inxi**

It condenses much of this information into a readable overall report and supplements system, driver and runtime information.

In difficult cases, both results can be compared.

## Inxi and LSHW

**LSHW**

represents the hardware hierarchically.

**Inxi**

It compiles the information more according to functional categories and as a system report.

## Inxi and dmidecode

**dmidecode**

Delivers targeted SMBIOS/DMI data.

Inxi can include parts of such system information in a larger report.

For precise DMI analysis, dmidecode remains the specialized tool.

## Inxi and lm-sensors

Inxi can display sensor values.

`lm-sensors` is the actual sensor tool.

This means the roles are divided as follows:

**Inxi**

Sensor information in the overall report.

**lm-sensors**

Targeted sensor query and sensor context.

## Inxi and SMART/NVMe

Inxi can display disk data and – depending on options and available tools – additional storage information.

However, the following remain necessary for a reliable, specialized diagnosis:

`smartctl`

and:

`nvme-cli`

the key tools of this Rescue manual.

## External utilities

Inxi gathers information from various system sources and can use existing utilities for certain functions.

Therefore, the actual available output may depend on:

operating system,

Hardware,

Kernel,

Permissions

and installed auxiliary tools.

## Root privileges

Access to certain more in-depth information may require elevated rights or privileged data sources.

However, not every Inxi output requires root privileges.

## Rescue Rule

Don't automatically start with root access.

First, generate a normal report.

Use elevated privileges only if the required information truly necessitates them.

## Limitations

Inxi is an information and diagnostic tool.

Inxi does not repair hardware.

Inxi does not repair file systems.

Inxi does not repair partition tables.

Inxi does not perform a full RAM test.

Inxi does not perform a CPU stress test.

Inxi does not replace SMART.

Inxi does not replace `nvme-cli`.

Inxi does not replace `lm-sensors`.

Inxi does not replace `dmidecode` for targeted SMBIOS/DMI analysis.

Inxi can only display information that is accessible to the running environment.

A live inxi report primarily describes the running rescue environment and its view of the hardware.

## Safety instructions

Always check Inxi reports before forwarding them externally.

sensitive information for support reports using `-z`.

Treat IP addresses, MAC addresses, serial numbers, hostnames and user information as potentially sensitive.

Use high levels of detail only when necessary.

Do not automatically conclude that a hardware component is free of defects just because it has been identified.

For storage problems, use Inxi only for identification and overview; perform health diagnostics with specialized tools.

Avoid generating unnecessary queries or loads on physically unstable data carriers.

For live systems, always document that the kernel, drivers, and desktop information belong to the rescue environment.

## Technical classification in JUMBO BLOCK Rescue

Inxi is the second tool in the area:

**Hardware.**

The roles are divided as follows:

**A.29 HWInfo**

Targeted hardware detection by device class.

↓

**A.30 Inxi**

Consolidated system and hardware report.

↓

**A.31 LSHW**

hierarchical hardware structure.

↓

**A.32 DMI Decode**

SMBIOS/DMI data.

↓

**A.33 Sensors**

Ongoing sensor information.

A sensible rescue procedure can therefore begin with:

```
inxi -b
```

for a quick overview.

Thereafter:

**HWInfo**

for a specific hardware class.

**LSHW**

for hierarchical device assignment.

**dmidecode**

for firmware/DMI information.

**lm-sensors**

for temperature and other supported sensor values.

Follow the steps for storage problems:

**SMART / NVMe**

and, if applicable:

**GNU ddrescue.**

Inxi's particular strength therefore lies not in a single hardware test, but in combining many sources of information into a quickly readable overall technical picture.

For JUMBO BLOCK Rescue, the core question of Inxi is:

**What does the currently running rescue system look like as a whole, considering the hardware it has detected, its drivers, and its essential system parameters?**

# A.31 LSHW

**Area:** Hardware
**Program:** LSHW – List Hardware
**Baseline 20261001:** 02.19.git.2021.06.19.996aaad9c7
**Installed package:** `lshw 02.19.git.2021.06.19.996aaad9c7-2.1`
**Operation:** Command line
**Task:** Detailed and hierarchical inventory of the hardware detected by the running system

## Purpose

LSHW stands for:

**List Hardware.**

The program extracts detailed information about a computer's hardware configuration and arranges detected components in a hierarchical device tree structure.

The Debian man page that corresponds exactly to the installed baseline provides information about, among other things:

Storage configuration,

Firmware version,

Mainboard configuration,

CPU version and speed,

Cache configuration,

Bus speeds

and other hardware features.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, LSHW serves in particular to:

to inventory the hardware structure of an unknown system,

To identify relationships between devices and higher-level components,

Examine storage controllers and data carriers together,

To capture network hardware and its configuration,

Documenting the CPU and memory structure,

Assigning PCI, USB and SCSI devices,

To determine bus addresses,

To examine driver and device configurations

and to generate a hardware report for diagnostics or documentation.

## Difference to HWInfo and Inxi

The three existing hardware tools have different focuses.

**HWInfo**

Enables targeted hardware detection by device class.

**Inxi**

generates a consolidated and easily readable system report.

**LSHW**

It represents hardware particularly as a hierarchical device tree.

The central question for LSHW is therefore:

**How are the hardware components detected by the system structurally connected and configured?**

## Basic call

The basic call to action is:

`lshw`

LSHW thus generates a detailed hardware tree.

## Root privileges

The man page accompanying the baseline explicitly states that LSHW must be run as superuser if complete information is desired.

Without the appropriate rights, only a portion of the available information will be displayed.

## Rescue Rule

A report initially generated without privileges may suffice for a quick overview.

If a complete hardware inventory is required, it must be taken into account that missing information may simply be due to restricted access rights.

## Hierarchical representation

The standard LSHW output is hierarchically structured.

This allows components to be viewed not only individually, but also within their hardware context.

For example, a structure might contain the following meaning:

system

→ Bus

→ Storage controller

→ Data carrier

or:

system

→ Bus

→ Network controller

→ Network interface.

### Importance in rescue operations

This hierarchy can be crucial if, for example, several:

Storage controller,

Network adapter,

USB devices

or data carrier

are available.

The question then is not only:

**What device exists?**

rather:

**Which bus or controller is this device connected to?**

### Brief view

With:

```
lshw -short
```

The hardware tree is output in a compact form.

The man page describes this representation using the hardware paths and explicitly refers to it as a compact hardware list.

### Typical entry point

Therefore, for an initial overview:

```
lshw -short
```

especially suitable.

After that, interesting hardware classes can be specifically investigated.

### Hardware path

The quick view shows hardware paths.

These paths help to identify the hierarchical position of a device within the hardware tree determined by LSHW.

They are not compatible with normal file system paths such as:

`/home/...`

or:

`/dev/...`

to equate.

### Bus information

With:

`lshw -businfo`

A device list with bus information is generated.

The man page specifically lists addresses for:

SCSI,

USB,

IDE

and PCI.

### Meaning

This allows, for example, investigation into which:

PCI device,

USB device

or SCSI path

a recognized hardware component is assigned.

## `-short` and `-businfo`

Both presentations provide a compact overview, but focus on different aspects:

`-short`

→ Hardware paths.

`-businfo`

→ Bus assignments.

### Hardware classes

With:

`-class CLASS`

The output can be limited to a specific hardware class.

The short form is:

`-C CLASS`.

### Determine classes

The available classes, or those occurring in the specific system, can be determined based on:

`lshw -short`

or:

`lshw -businfo`

to be determined.

### Combining multiple classes

`-class`

can be specified multiple times.

The man page gives the following as an example:

`lshw -class disk -class storage`

This allows data carriers and storage controllers to be displayed together.

### Storage diagnostics

This combination is particularly useful for JUMBO BLOCK Rescue:

`lshw -class disk -class storage`

She answers two related questions:

**Which storage media does the system recognize?**

and:

**Which storage controllers does the system detect?**

### Network hardware

A targeted query can be performed, for example, using:

`lshw -class network`

This restricts output to network hardware.

### Network report as HTML

The man page accompanying the baseline explicitly states:

```
lshw -html -class network
```

as an example of outputting network hardware information in HTML.

### Configuring a network device

LSHW can display not only the device identity of network hardware, but also configuration data depending on the available information.

This can include, for example, information about the interface used and driver assignment.

### Hardware detection does not mean network is working.

The same applies as with HWInfo and Inxi:

A detected network controller does not prove that:

a correct IP configuration is present,

DNS is working,

the gateway is reachable

or a network connection can be successfully established.

The network tools from A.17 to A.24 are available for this purpose.

### CPU

LSHW can capture detailed processor information.

The man page explicitly states:

CPU version

and:

CPU speed

as examples of the obtainable hardware data.

### CPU detection is not a stress test

LSHW inventories the CPU.

However, it does not test the stability of the processor under load.

A correctly identified CPU is therefore not automatically demonstrably error-free.

### Cache

LSHW can also capture the cache configuration.

This allows the CPU hierarchy to contain additional cache components.

## RAM

LSHW can output detailed information about the memory configuration.

On supported systems, various storage structures and memory banks can become visible.

## SPD recognition

LSHW has a hardware test:

`spd`

for:

**Serial Presence Detect**

of memory modules.

This test can be performed via the general mechanisms:

`-enable`

or rather:

`-disable`

can be controlled.

## RAM inventory is not a RAM test.

Even detailed information about memory banks or modules does not prove that the RAM is working flawlessly.

LSHW does not perform a full memory error test.

## Firmware and motherboard

LSHW can capture firmware and motherboard information.

The man page states:

Firmware version

and:

Motherboard configuration

explicitly as typical expenses.

## DMI

LSHW supports the following on x86 and IA-64 systems:

DMI or SMBIOS.

This information can form part of the hardware inventory.

## Differentiation from dmidecode

The following chapter:

**A.32 DMI Decode**

Uses `dmidecode`, a specialised tool for SMBIOS/DMI structures.

LSHW, on the other hand, uses DMI as one of several information sources and integrates the results into its hardware tree.

## Supported information sources

The man page that corresponds exactly to the baseline lists support for:

DMI

OpenFirmware Device Tree,

PCI/AGP

CPUID,

IDE/ATA/ATAPI,

PCMCIA

SCSI

and USB.

## Linux system information

LSHW also accesses various kernel and system information.

The man page mentions, among other things:

`/sys/*`

as well as various `/proc` and device paths as information sources.

## Rescue perspective

LSHW thus describes the hardware from the perspective of the currently running system:

**JUMBO BLOCK Rescue Systems.**

As with Inxi, the following applies:

Driver and runtime information refers to the rescue environment and not automatically to an operating system installed on an examined disk.

## PCI and USB

LSHW can detect PCI and USB devices and integrate them into the hardware tree.

For a compact classification, the following is particularly suitable:

```
lshw -businfo
```

### Numerical device identifiers

With:

```
-numeric
```

Additional numerical IDs for PCI and USB devices are displayed.

### Using numeric IDs

Manufacturer and device identifiers can be helpful if:

a device is only recognized generically,

A driver problem is being investigated.

or the exact PCI/USB device identity is required.

### Example principle

```
lshw -numeric -class network
```

can represent network hardware along with additional numerical device identifiers.

### Data carrier

The class:

```
disk
```

It is used for the targeted display of detected data carriers.

### Storage controller

The class:

```
storage
```

Used to display corresponding storage hardware.

### combination

Therefore, the following is particularly useful in a rescue situation:

```
lshw -class disk -class storage
```

This combination is even documented as an example in the official man page of the installed version.

### Detected data carrier does not mean data carrier is healthy.

LSHW can provide model, device mapping, and other hardware data.

However, it is not a tool for analyzing:

SMART attributes,

NVMe health data,

Media errors

or physical wear and tear.

### Condition diagnosis

The following are available for this purpose in JUMBO BLOCK Rescue:

**A.8 SMART Diagnosis**

and:

**A.9 NVMe Diagnosis.**

### Hardware tree and partitions

LSHW can represent storage structures in the hardware tree.

However, for targeted partition editing:

**A.6 GParted**

responsible.

For graphical disk management:

**A.7 GNOME Disks.**

### HTML output

With:

`-html`

The device tree is output as an HTML page.

This can be useful for readable hardware documentation.

### XML output

With:

`-xml`

The device tree is output as an XML structure.

This is particularly suitable for further machine processing.

### JSON output

With:

`-json`

The device tree is output as JSON.

This allows the hardware structure to be further processed by scripts or other tools.

### Choose output format deliberately

Different formats are therefore suitable for different rescue purposes:

**normal output**

human detailed analysis.

**-short**

quick overview.

**-businfo**

Bus assignment.

**HTML**

readable report.

**XML / JSON**

structured further processing.

### Save report to file

As with other command-line programs, the normal output via the shell can be redirected to a file.

Example:

```
lshw -short > lshw-short.txt
```

### HTML report

Example:

```
lshw -html > lshw-report.html
```

### JSON report

Example:

```
lshw -json > lshw-report.json
```

### Sensitive information

Hardware reports may contain identifying data.

This could include, for example:

Serial numbers,

IP addresses,

Hardware identifiers

and other system-specific information.

## `-sanitize`

LSHW therefore has the option:

`-sanitize`

It removes potentially sensitive information from the output.

The man page explicitly states:

IP addresses,

Serial numbers

and similar data.

### Support report

For a report to be shared externally, a combination with the following is therefore possible, for example:

`-sanitize`

It would be useful.

### Example principle

```
lshw -sanitize -short
```

or:

```
lshw -sanitize -html > hardware.html
```

### Check the filtering anyway

Even a sanitized hardware report still contains technical information about the system.

Therefore, the report should also be manually reviewed before publication.

## `-notime`

With:

`-notime`

Volatile time-related attributes such as timestamps are excluded from the output.

### Benefits for comparative reports

This can be helpful when two hardware reports need to be automatically compared and fleeting time information would unnecessarily alter the comparison.

### `-quiet`

With:

`-quiet`

Status messages are suppressed.

This can be especially true in the following cases:

File output,

scripts

or automated processing

be useful.

## Enable or disable hardware tests

LSHW supports:

`-enable TEST`

and:

`-disable TEST`.

This allows certain detection mechanisms to be switched on or off.

## Documented tests

The Baseline man page states:

`dmi`

`device-tree`

`spd`

`memory`

`cpuinfo`

`cpuid`

`pci`

`isapnp`

`pcmcia`

`ide`

`usb`

`scsi`

and:

`network`.

### Meaning of "test"

The term should not be confused with a hardware stress test.

These tests identify detection mechanisms or information sources of LSHW.

### Example

The man page states:

```
lshw -disable dmi
```

as an example.

This means that DMI is not used for hardware detection.

### Why disable information sources?

This can happen, for example, in the following cases:

Troubleshooting,

Comparison of different detection methods

or problematic sources of information

be useful.

However, for a normal Rescue report, it is not necessary to disable detection methods without a specific reason.

### SQLite dump

LSHW supports:

`-dump DATEINAME`.

This allows the collected information to be written to an SQLite database.

### Meaning

This allows for structured storage of hardware information for later evaluation.

However, for a normal, human-readable Rescue report, the following are required:

Text,

HTML

or:

JSON

often usable more directly.

### Version query

With:

```
lshw -version
```

The LSHW version will be output.

### Help

With:

```
lshw -help
```

The available command-line options will be displayed.

### Graphical user interface

LSHW knows:

```
-X
```

to launch an X11 GUI, if one is available.

### Baseline demarcation from the GUI

The packet baseline confirms:

```
lshw
```

in version:

`02.19.git.2021.06.19.996aaad9c7-2.1`.

This package line alone does not imply an additional `lshw-gtk` interface being installed for JUMBO BLOCK Rescue.

Therefore, this chapter assumes only the documented command-line function of `lshw`.

### Typical rescue scenario: unknown computer

A sensible procedure is as follows:

**1. Compact hardware overview**

```
lshw -short
```

**2. Bus assignment**

```
lshw -businfo
```

**3. Identify conspicuous hardware class**

**4. Examine this class specifically**

for example:

```
lshw -class network
```

or:

```
lshw -class disk -class storage
```

**5. Supplement with HWInfo, Inxi or a special tool if necessary.**

### Typical rescue scenario: Data carrier is missing

**1.**

```
lshw -class disk -class storage
```

**2.**

Bus assignment with:

```
lshw -businfo
```

check.

**3.**

Identify storage controllers.

**4.**

Supplement with HWInfo if necessary.

**5.**

When the storage device becomes visible, perform SMART or NVMe diagnostics.

### Typical rescue case: Network adapter not working

**1.**

```
lshw -class network
```

**2.**

Check device, bus assignment, and available configuration information.

**3.**

if needed:

```
lshw -numeric -class network
```

use.

**4.**

Verify using HWInfo or Inxi.

**5.**

Then examine the network function using the network tools.

### Typical Rescue case: USB device unknown

**1.**

```
lshw -businfo
```

**2.**

Identify USB bus assignment.

**3.**

Add numerical device identifiers if necessary.

**4.**

using HWInfo `--usb`.

### Typical rescue case: Hardware documentation

For example, documentation to be shared can be:

```
lshw -sanitize -html
```

used and the output written to an HTML file.

Before sharing:

**Manually check the report for sensitive information.**

### Typical Rescue case: machine comparison

For structured further processing, the following can be used:

JSON

or:

XML

be used.

With:

`-notime`

Time-dependent attributes can be hidden if they are disruptive to a comparison.

### HWInfo, Inxi and LSHW together

The three tools can be deliberately combined in complex cases.

**HWInfo**

It specifically queries device classes.

**Inxi**

It provides a compact overall report containing hardware and system information.

**LSHW**

shows the hierarchical device and bus structure.

If all three tools detect the same hardware, a good technical basis for further diagnosis is created.

## Divergent results

Differences between the tools do not automatically mean that one of them is malfunctioning.

They can be different:

Information sources,

Representation models,

Access rights

and detection methods

use.

The cause of any deviation must therefore be investigated.

## Limitations of hardware detection

The Baseline man page explicitly mentions limitations.

LSHW does not recognize FireWire/IEEE-1394 devices, among others.

Furthermore, not all architectures supported by GNU/Linux are fully supported; the man page cites CPU detection as an example.

According to the documentation, virtual SCSI interfaces for SCSI emulation via IDE are also not displayed correctly.

## Key conclusion

**“Not present in LSHW” does not automatically mean “physically not present”.**

The output describes what LSHW was able to detect using its supported detection mechanisms.

## Hardware health

LSHW does not perform a full health check of the components.

It does not test:

Data carrier surfaces,

SMART status,

NVMe-Health,

RAM cells,

CPU stability

or electrical sensor limits.

### Writing effect

The usual inventory calls are used to read and display hardware information.

LSHW is not a tool for partitioning or file system repair.

### Nevertheless, use with caution.

Diagnostic access can also have an impact on unstable or faulty hardware.

Therefore, when a data carrier is physically failing, a large number of different diagnostic programs should not be applied to it repeatedly and unnecessarily.

If data recovery is a priority, early acquisition with GNU ddrescue may be more important.

### Live system perspective

Like HWInfo and Inxi, LSHW operates from within the running Rescue environment.

The detected devices and active configurations are thus, from the perspective of:

**JUMBO BLOCK Rescue Kernels**

determined.

### Installed target system

Therefore, an LSHW report from JUMBO BLOCK Rescue is not automatically a report about the driver configuration of a damaged Linux system on the hard drive.

It documents the current Rescue view of the hardware.

### Safety instructions

Use root privileges deliberately, as required for complete hardware information.

Treat hardware reports with `-sanitize` and additionally check them before forwarding.

Consider serial numbers, IP addresses, and device identifiers as potentially sensitive data.

Before performing any data write operations, verify the device identity and bus assignment.

Do not confuse data carrier detection with data carrier health.

Avoid unnecessary diagnostic accesses on physically abnormal data carriers.

A missing device in LSHW should not automatically be interpreted as evidence of physical absence.

Mark live system reports as viewed by JUMBO BLOCK Rescue.

## Technical classification in JUMBO BLOCK Rescue

LSHW is the third program in the hardware block.

The roles are divided as follows:

**A.29 HWInfo**

Broad hardware detection by device class.

↓

**A.30 Inxi**

consolidated system report.

↓

**A.31 LSHW**

hierarchical hardware and bus structure.

↓

**A.32 DMI Decode**

Targeted SMBIOS/DMI information.

↓

**A.33 Sensors**

Current supported sensor values.

LSHW complements the two previous programs, especially where not only the identity of a device, but also its:

**Position and relationship within the hardware structure**

is relevant.

A typical storage diagnostics path is, for example:

```
lshw -short
```

↓

```
lshw -class disk -class storage
```

↓

```
lshw -businfo
```

↓

**HWInfo**

further device detection.

↓

**SMART / NVMe**

Health.

↓

**GNU ddrescue**

in the case of an unstable source medium.

The central role of LSHW in JUMBO BLOCK Rescue is therefore:

**Don't just list the hardware, but make its technical structure and classification visible.**

This is crucial, especially before critical rescue operations:

**What device is present – and where is it located in the system?**

# A.32 DMI Decode

**Area:** Hardware
**Program:** dmidecode
**Baseline 20261001:** dmidecode 3.6
**Installed package:** dmidecode 3.6-2
**Operation:** Command line
**Task:** Read SMBIOS/DMI tables and display them in human-readable form

## Purpose

dmidecode reads a computer's DMI/SMBIOS tables and decodes their contents into a human-readable representation.

SMBIOS stands for:

**System Management BIOS**

DMI for:

**Desktop Management Interface.**

The tables contain descriptions of the system hardware provided by BIOS or UEFI/firmware.

This typically includes information about:

System manufacturer,

System model,

Serial number

System UUID,

BIOS or firmware,

Mainboard,

Housing,

Processor,

Cache,

Memory configuration,

slots,

Connections

and further platform information.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, dmidecode is particularly suitable for:

Identification of an unknown computer,

Determination of manufacturer and system model,

Documentation of BIOS/firmware information,

Identification of the motherboard,

Determination of case information,

Examination of the CPU configuration described by the SMBIOS,

Detection of memory banks and memory modules,

Determination of serial numbers and asset tags,

Display of system UUIDs,

Examination of existing slots

and documentation of the hardware configuration reported by the firmware.

## Key difference to HWInfo and LSHW

dmidecode does not primarily examine the hardware through active hardware detection.

Instead, it reads the one provided by the system:

**SMBIOS/DMI table.**

This has one important advantage:

The information can be read quickly without having to query the actual hardware individually.

But it also has a crucial limit:

**The firmware data may be incorrect, incomplete, or unreliable.**

The dmidecode documentation explicitly points out this fact.

## Basic rule

**dmidecode shows what the firmware claims about the hardware – not necessarily what is actually physically present.**

## Basic call

The full appeal reads:

dmidecode

The program attempts to find and read the DMI table and decode all supported entries.

## Root privileges

Full access to SMBIOS/DMI data requires privileged access on typical Linux systems.

A typical rescue call therefore reads:

sudo dmidecode

### Data source under Linux

Under Linux, dmidecode first attempts to access the DMI table via:

**sysfs**

to read.

If this access fails, it then attempts direct memory access.

### Direct memory access

For direct access, dmidecode uses the following by default:

/dev/mem

With:

-d FILE

or rather:

--dev-mem FILE

A different device file can be specified.

For a normal rescue operation, a change to this source is usually not necessary.

### Structure of a DMI entry

A DMI dataset contains, among other things:

a handle

a type

a size

and:

decoded values.

A handle is a unique identifier that DMI entries can use to reference each other.

For example, a processor entry can reference a cache entry.

### DMI types

SMBIOS defines different data record types.

Among those particularly important for rescue operations are:

0 – BIOS

1 – System

2 – Baseboard

3 – Chassis

4 – Processor

7 – Cache

9 – System Slots

16 – Physical Memory Array

17 – Memory Device

22 – Portable Battery

32 – System Boot

38 – IPMI Device.

### Targeted query with --type

With:

-t TYPE

or rather:

--type TYPE

The output can be restricted to specific DMI types.

TYPE can do this:

a type number

a comma-separated list of type numbers

or a supported keyword

be.

### Supported type keywords

Version 3.6 supports the following keywords for this purpose:

BIOS

system

baseboard

chassis

processor

memory

cache

connector

slot.

### BIOS

For BIOS/firmware information:

dmidecode --type bios

The keyword encompasses the corresponding BIOS-related DMI types.

### system

For system information:

dmidecode --type system

Information such as manufacturer, product name, version, serial number or UUID may appear here, provided the firmware provides this data.

### Mainboard

For motherboard information:

dmidecode --type baseboard

This allows information such as the manufacturer, product name, version, and serial number of the baseboard to be determined.

### Housing

For case information:

dmidecode --type chassis

### processor

For processor information:

dmidecode --type processor

### memory

Particularly important for Rescue diagnoses is:

dmidecode --type memory

The keyword encompasses several memory-related DMI types. In version 3.6, these include types 5, 6, 16, and 17.

### Physical Memory Array

DMI type 16 describes a physical storage array.

It may contain information such as the total possible capacity or the number of storage devices.

### Memory Device

DMI type 17 describes individual storage devices or memory slots.

Depending on the firmware, information such as the following may appear there:

Size,

Form factor

Locator,

Type,

Speed,

Manufacturer,

Serial number

Part Number

and other storage properties.

### Empty memory banks

SMBIOS can also write to slots or memory devices in which no memory module is currently installed.

This allows dmidecode to do both:

existing modules

as well as:

Memory locations described by the firmware

show.

### Important: Maximum capacity is not installed storage.

The DMI table can describe not only the current configuration, but also possible expansion states.

The dmidecode documentation explicitly lists the following as examples:

the fastest supported CPU

or:

the maximum supported storage capacity.

Therefore, a distinction must be made between:

**installed**

and:

**According to the firmware, it is supported**

They can be distinguished.

### RAM data is not a memory test

If dmidecode reports a memory module, this does not mean that the module is working correctly.

dmidecode does not perform a memory load or memory error test.

### Individual values with --string

With:

-s KEYWORD

or rather:

--string KEYWORD

A single defined DMI value can be output.

### System manufacturer

dmidecode -s system-manufacturer

### System model

dmidecode -s system-product-name

### System version

dmidecode -s system-version

### Serial number

dmidecode -s system-serial-number

### System UUID

dmidecode -s system-uuid

### BIOS manufacturers

dmidecode -s bios-vendor

### BIOS version

dmidecode -s bios-version

### BIOS date

dmidecode -s bios release date

### Motherboard manufacturer

dmidecode -s baseboard-manufacturer

### Mainboard model

dmidecode -s baseboard-product-name

### Motherboard serial number

dmidecode -s baseboard-serial-number

### Case information

Version 3.6 supports, among other things:

chassis manufacturer

chassis type

chassis version

chassis serial number

and:

chassis-asset-tag.

### Processor information

Supported string keys include:

processor family

processor manufacturer

processor version

and:

processor frequency.

### Multiple processors

A single string keyword can return multiple results on multiprocessor systems.

The documentation explicitly mentions processor-version as an example.

### Show available string keys

Version 3.6 included:

--list-strings

a.

This allows the following:

--string

Available keywords are listed.

### Show available types

Also introduced with version 3.6:

--list-types

This allows you to see the available keywords for:

--type

be listed.

### Show specific handle

With:

-H HANDLE

or rather:

--handle HANDLE

The DMI entry whose 16-bit handle corresponds to the specified identifier can be displayed.

### Using the handles

Handles are particularly useful for detailed technical analysis when one DMI entry refers to another.

This allows relationships within the SMBIOS structure to be traced.

### Quiet mode

With:

-q

or rather:

--quiet

The output will be reduced.

According to the documentation, this includes, among other things:

unknown,

inactive

and OEM-specific entries

not displayed.

Furthermore, metadata and handle references are hidden.

### Rescue operation of --quiet

Quiet mode can be helpful for a quick, readable report.

However, for a complete technical investigation, it should be taken into account that this deliberately omits information.

### Raw data as hex dump

With:

-u

or rather:

--dump

The entries are not decoded normally, but output in hexadecimal format.

Appended strings appear in both hexadecimal and ASCII formats.

According to the documentation, this function is primarily intended for debugging.

### Binary DMI fuse

With:

--dump-bin FILE

The DMI structure can be written to a file in binary format.

The generated file can later be used again with:

--from-dump FILE

be read in.

### Important file rule

For:

--dump-bin FILE

dmidecode 3.6 requires that the target file does not already exist.

### Offline evaluation

A sensible procedure could therefore be:

dmidecode --dump-bin system-dmi.bin

The saved DMI structure can then be used later with:

dmidecode --from-dump system-dmi.bin

to be decoded again.

### Benefits for Rescue documentation

This allows the SMBIOS/DMI data set read at the time of the investigation to be saved and evaluated later.

This is especially helpful if:

the computer will later no longer be available,

Hardware is being replaced

or that the DMI data should be archived separately.

### A DMI dump is not a disk image.

A file created with --dump-bin contains the DMI data.

It does not contain:

Hard drive contents,

RAM contents,

File systems

or a complete firmware image.

### OEM Strings

With:

--oem-string N

A specific OEM string can be output.

The numbering starts at 1.

With:

--oem-string count

The number of existing OEM strings can be determined.

### OEM data

OEM-specific entries may contain manufacturer-dependent information.

dmidecode can only fully decode such data if corresponding manufacturer information or support is available.

### --no-quirks

With:

--no-quirks

The table values are decoded as directly as possible, without correcting known firmware errors or hiding irrelevant fields.

The documentation describes this mode primarily as a function for firmware developers.

### Rescue classification of --no-quirks

For a normal hardware inventory:

--no-quirks

not required.

This option is more helpful when the unmodified firmware display itself is the subject of investigation.

### --no-sysfs

With:

--no-sysfs

dmidecode is instructed not to read the DMI data via sysfs.

According to the documentation, this option is primarily intended for debugging.

### Firmware quality is crucial

The reliability of the output depends significantly on what data the manufacturer has entered into SMBIOS.

Potentially problematic values include, for example:

missing serial numbers,

generic model names,

Unmaintained asset tags,

Incorrect RAM specifications,

Placeholder

or incomplete tables.

### Typical placeholders

On real systems, manufacturers can store generic values instead of meaningful data.

Such values must not be treated as verified hardware identities solely on the basis of their output by dmidecode.

### Firmware specifications and physical labeling may differ.

For critical identification tasks, important data should be cross-checked with other sources if necessary.

For example:

Case label

Mainboard labeling,

Data carrier label

HWInfo,

LSHW

or manufacturer-specific diagnostic information.

### Serial numbers

Serial numbers are especially valuable for inventory management and support.

They are also potentially sensitive identifiers.

Therefore, a complete dmidecode report should not be published without review.

### System UUID

The same applies to:

**System UUID.**

It can make a system uniquely or permanently identifiable.

### Asset Tags

Asset tags can contain internal company identifiers.

They too should be checked before a report is passed on externally.

### Data protection

Unlike Inxi -z or LSHW -sanitize, dmidecode does not have a general sanitizing switch that automatically cleans up a full report for public distribution.

Therefore:

**Manually check dmidecode output before forwarding and reduce it if necessary.**

### Better to ask targeted questions

If only the BIOS version is needed, then:

dmidecode -s bios-version

more data-efficient than sharing a complete:

dmidecode

Reports.

### BIOS/Firmware diagnostics

A useful firmware overview can be achieved, for example, with:

dmidecode --type bios

be generated.

This allows BIOS-related DMI entries to be specifically examined.

### BIOS version is not a firmware integrity check.

dmidecode can display a reported firmware version.

However, it does not verify:

the cryptographic integrity of the firmware,

their safety

whether malicious code is present

or whether the firmware actually corresponds to the unmodified manufacturer's version.

### Mainboard diagnostics

For motherboard information:

dmidecode --type baseboard

This allows the board information reported by SMBIOS to be viewed in a targeted manner.

### System identification

A quick identification process might look like this:

dmidecode -s system-manufacturer

dmidecode -s system-product-name

dmidecode -s system-serial-number

### Please respect data protection

The third query returns a serial number, if one is available.

This should not appear unnecessarily in publicly shared screenshots or support posts.

### RAM inventory

For a detailed storage overview:

dmidecode --type memory

This allows the memory arrays and memory modules described by the firmware to be examined.

### Typical rescue case: RAM upgrade

dmidecode can provide clues about:

existing memory banks,

occupied or described slots,

installed module sizes

and capacities supported according to SMBIOS.

However, before actually purchasing hardware, this information should be compared with the manufacturer's documentation for the specific system.

## Reason

SMBIOS can:

faulty

incomplete

or outdated

be.

## Typical rescue scenario: unknown computer

**1. System manufacturer**

dmidecode -s system-manufacturer

**2nd model**

dmidecode -s system-product-name

**3. BIOS**

dmidecode --type bios

**4. Mainboard**

dmidecode --type baseboard

**5. Storage**

dmidecode --type memory

**6. Compare results with HWInfo, Inxi or LSHW**

## Typical rescue scenario: Preparing for a BIOS update

Before a firmware update, for example:

System manufacturer,

System model,

Mainboard model

and:

current BIOS version

to be determined.

The firmware package itself must then be selected exclusively based on the manufacturer's unique information for the specific system.

## Critical safety notice

**A seemingly appropriate BIOS designation from dmidecode alone does not justify a firmware update.**

Before writing firmware, the model, hardware variant, and manufacturer approval must be clearly verified.

## Typical rescue scenario: Document hardware before replacement

Before exchanging:

Mainboard,

R.A.M,

CPU

or complete system

Can a DMI dump be saved?

dmidecode --dump-bin system-before.bin

Additionally, a human-readable version can be archived.

## After hardware change

After an exchange, a new report can be generated and compared with the previous state.

It should be noted that firmware and manufacturer data do not necessarily reflect every physical change correctly.

## Typical rescue case: System report

For an internal report, the complete output can be written to a file:

dmidecode > dmidecode-report.txt

Before sharing information externally, sensitive data must be removed or the report reduced to the information actually needed.

## Typical rescue scenario: only BIOS version required

Instead of the full report:

dmidecode -s bios-version

This is clearer and uses less data.

### Typical rescue scenario: only the system model is needed.

dmidecode -s system-product-name

### Typical rescue case: Memory problem

A sensible diagnostic pathway is:

**1.**

dmidecode --type memory

Record the firmware description of the memory configuration.

**2.**

Cross-check with LSHW and Inxi.

**3.**

consider the actual memory available from the kernel.

**4.**

If RAM errors are suspected, perform a proper memory test.

### dmidecode and HWInfo

**HWInfo**

Detects hardware via various system and hardware sources.

**dmidecode**

It specifically decodes the SMBIOS/DMI table provided by the firmware.

### dmidecode and Inxi

**Inxi**

integrates a lot of system information into a readable overall report.

**dmidecode**

provides a significantly more detailed direct view of SMBIOS/DMI.

### dmidecode and LSHW

**LSHW**

builds a hierarchical hardware tree from multiple information sources.

**dmidecode**

focuses on the firmware tables.

LSHW can itself use DMI as one of its sources of information.

### Contradictions between tools

For example:

dmidecode,

LSHW

and:

HWInfo

If different data is provided, the DMI output should not automatically be considered authoritative.

Instead, the following must be investigated:

What source provides the respective information?

### Physical reality prior to firmware specification

For critical rescue decisions, the following applies:

**Actually identified or physically verified hardware takes precedence over an unconfirmed firmware description.**

### SMBIOS support from version 3.6

dmidecode 3.6 was released on April 24, 2024.

This version added support for:

SMBIOS 3.6.0

and:

SMBIOS 3.7.0.

This included new types of storage devices, processor upgrades, port types, slot features, and other memory areas.

### New options in 3.6

Version 3.6 also included:

--list-strings

and:

--list-types

a.

### Newer SMBIOS versions than dmidecode are supported.

If the firmware reports an SMBIOS version that is not yet supported by the dmidecode version being used, dmidecode will issue a corresponding message.

The documentation warns against considering the data provided as reliable.

### Rescue consequence

Such a warning must not be ignored.

The report should then be marked accordingly and compared with other hardware sources.

### Limitations

dmidecode reads SMBIOS/DMI data.

dmidecode does not perform a hardware stress test.

dmidecode does not check RAM for memory errors.

dmidecode does not check CPU stability.

dmidecode does not check the data carrier health.

dmidecode does not replace SMART.

dmidecode does not replace NVMe diagnostics.

dmidecode does not measure temperatures.

dmidecode does not physically verify manufacturer specifications.

dmidecode does not prove that an extended state described by SMBIOS is actually practically usable.

dmidecode does not check firmware integrity.

Firmware data may be incorrect or incomplete.

### Safety instructions

Do not treat DMI data as physically verified hardware data without verification.

Treat serial numbers, UUIDs, and asset tags as sensitive identifiers.

Do not publish full reports without review.

For external support cases, only provide the information that is actually needed.

Never select firmware updates based solely on a single dmidecode entry.

Do not confuse memory information with a RAM function test.

In case of conflicting information, use additional hardware tools for cross-checking.

Take warnings about unsupported SMBIOS versions seriously.

### Technical classification in JUMBO BLOCK Rescue

dmidecode is the fourth program in the hardware block:

**A.29 HWInfo**

broad hardware detection.

↓

**A.30 Inxi**

consolidated system report.

↓

**A.31 LSHW**

hierarchical hardware and bus structure.

↓

**A.32 DMI Decode**

Firmware description via SMBIOS/DMI.

↓

**A.33 Sensors**

Current readings from supported hardware sensors.

The four tools discussed so far consider the same system from different perspectives:

**HWInfo**

What does the Rescue environment detect?

**Inxi**

In summary, what is the current system configuration?

**LSHW**

How are devices and buses structurally assigned?

**dmidecode**

What information does the firmware provide about the system and hardware?

The combination of these perspectives is particularly valuable.

Voices:

Manufacturer,

System model,

Mainboard,

CPU

and storage information

By agreeing via several independent information channels, a significantly more reliable inventory is created than one based on a single source.

Therefore, the central security rule of DMI Decode is:

**SMBIOS/DMI is a valuable source of information – but not an independent physical verification of the hardware.**

# A.33 Sensors / lm-sensors

**Area:** Hardware
**Program:** lm-sensors
**Baseline 20261001:** lm-sensors 3.6.2
**Installed package:** `lm-sensors 1:3.6.2-2`
**Associated library:** `libsensors5 1:3.6.2-2`
**Configuration package:** `libsensors-config 1:3.6.2-2`
**Operation:** Command line
**Task:** Detect and display sensor values provided by the Linux hardware monitoring system

## Purpose

lm-sensors is a collection of tools for hardware monitoring under Linux.

The central command:

`sensors`

Displays the current measured values of the detected and system-supported sensor chips.

Depending on hardware, kernel drivers, and sensor support, this can include, for example:

CPU temperatures,

Motherboard temperatures,

additional temperature sensors,

Fan speeds,

tensions,

Current values

Performance values

and other metrics provided by hardware monitoring drivers.

Which values are actually available depends entirely on the specific system.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, lm-sensors serves primarily to:

to check current temperature values

to recognize unusual thermal conditions,

to check available fan speeds,

to view voltage values provided by the system,

To investigate hardware problems with possible temperature references,

to assess the thermal situation before longer rescue operations

and to combine sensor information with other hardware diagnostics.

## Central demarcation

lm-sensors is a:

**Hardware monitoring tool.**

This is not a full hardware test.

A sensor reading initially only answers:

**What measurement value is currently being provided by the sensor or driver in question?**

He does not answer automatically:

**Is all the hardware free of defects?**

## Three separate levels

Three levels should be distinguished when working with lm-sensors:

**1. Sensor hardware**

A physical sensor or monitoring chip must be present.

**2. Kernel support**

A suitable driver must recognize the sensor and provide its data.

**3. lm-sensors**

`sensors`reads the provided values and displays them in a readable format.

If one of these levels is missing, an expected measurement may be missing.

## Basic call

The most important appeal is:

```
sensors
```

It displays the current readings of all sensor chips found by libsensors.

## Typical issue

Depending on the system, output may contain categories such as:

Temperature,

Fan speed

Tension,

Performance

or other hardware-dependent values.

The actual output varies considerably between different computers.

### Do not expect a fixed sensor list.

There is no universal output that must be identical on every computer.

For example, a computer can:

CPU temperatures only,

CPU and motherboard values,

numerous voltage and fan values

or no values usable by lm-sensors at all

provide.

### Temperatures

Temperatures are displayed in degrees Celsius by default.

With:

```
sensors -f
```

or rather:

sensors -- Fahrenheit

Temperature values are displayed in Fahrenheit instead.

### Celsius for technical documentation

For JUMBO BLOCK Rescue, the standard Celsius output should generally be used.

This makes it easier:

Comparison,

documentation

and technical communication.

### Chip selection

The `sensors` can be restricted to specific sensor chips or chip patterns.

The basic scheme is:

sensors [OPTIONS] [CHIPS]

This means that, in complex systems, it is not necessarily necessary to consider the entire sensor output.

### Adapter information

By default, adapter information can also appear for sensor chips.

With:

```
sensors -A
```

or rather:

```
sensors --no-adapter
```

These adapter details will be hidden.

### To use

This can produce a more compact representation if only the measured values themselves are relevant.

### Raw data

With:

```
sensors -u
```

A raw representation is output.

According to the documentation, this mode is particularly suitable for:

Debugging,

Further processing by scripts

and:

Creation or investigation of a sensor configuration.

### Difference from normal output

The standard output is designed for human readers.

`-u`, on the other hand, shows the underlying raw names and values.

Therefore, for a normal Rescue quick check:

```
sensors
```

Usually easier to read.

For a detailed technical analysis, you can:

```
sensors -u
```

be helpful.

### JSON output

lm-sensors supports JSON output.

This allows sensor data to be further processed by machines.

The Baseline documentation provides corresponding JSON output options for this.

### Benefits of structured output

This can be helpful, for example, for:

Scripts,

automatic documentation,

Monitoring

or subsequent data processing.

For a manual Rescue diagnosis, the standard output is usually sufficient.

### version

With:

```
sensors -v
```

or rather:

```
sensors --version
```

The program version is displayed.

### Help

With:

```
sensors -h
```

or rather:

```
sensors --help
```

The available options will be displayed.

### Configuration files

libsensors uses system-wide configuration files.

This includes:

```
/etc/sensors3.conf
```

and as an older alternative:

```
/etc/sensors.conf
```

.

Additional configuration files can be found at:

```
/etc/sensors.d
```

lay.

### Meaning of the configuration

The configuration can affect, among other things:

designations,

calculations,

ignored values,

Limit values

and other display and sensor properties.

### Alternative configuration file

With:

sensors -c FILE

or rather:

sensors --config-file FILE

A specific configuration file can be used.

### Temporarily bypass configuration

The documentary mentions:

```
sensors -c /dev/null
```

as a way to temporarily disable the normal default configuration.

This can be helpful when troubleshooting.

### Rescue meaning

If a sensor value is due to a configuration:

renamed,

converted

ignored

or otherwise altered

If this appears, a comparison with the raw display or without standard configuration can help with diagnosis.

### Sensor limits

Sensor chips can provide different threshold values depending on the hardware.

This could include, for example:

maximum temperatures,

critical temperatures

or hysteresis limits.

### Limit value does not automatically constitute manufacturer approval.

A limit value displayed by `sensors`must not be interpreted as a general safe operating limit of the entire hardware without verification.

The value can:

from the sensor chip,

from the driver,

from the firmware

or:

from the libsensors configuration

originate.

### Plausibility check

Therefore, a displayed measurement value should always be evaluated within its technical context.

For example, an obviously unrealistic voltage or temperature would initially be a reason to:

Sensor assignment

Driver support

and:

configuration

to check.

### Incorrect or unconnected sensor inputs

Hardware monitoring chips can have inputs that are located on a specific motherboard:

not connected

wired differently

or used differently by the manufacturer

become.

A value output is therefore not physically meaningful simply because `sensors`can display it.

### Fan speeds

If the appropriate sensors and drivers are available, fan speeds can be displayed.

These are typically found in:

RPM

specified.

### Zero value for the fan

A displayed speed of zero can have various causes.

For example:

The fan is actually stopped.

The fan is switched off depending on the temperature.

Sensor cable is missing

wrong entrance,

Driver/configuration problem.

Therefore, a single zero value should not automatically be interpreted as a fan defect without further investigation.

### tensions

Supported hardware monitoring chips can provide voltage values.

The same applies here:

**Check plausibility.**

An implausible value can result from incorrect scaling or an unsuitable sensor configuration.

### Power and current values

Modern kernel hardware monitoring drivers may additionally, depending on the platform:

Perfomance,

Electricity

or other measured variables

provide.

Whether these appear in `sensors`depends on the respective hardware and driver support.

### CPU sensors

Many modern processors have internal temperature sensors.

If the corresponding kernel drivers are active, these values can appear via lm-sensors.

### Multiple CPU cores

Depending on the processor and driver:

Total values,

Package values

or:

individual core values

will be displayed.

Not every CPU provides the same structure.

### Sensor values are snapshots.

The command:

`sensors`

shows current values.

It is not a historical monitoring archive.

A single unremarkable measurement therefore does not prove that the system was also unremarkable under load or at an earlier time.

### Repeated observation

To diagnose a thermal problem, it can be useful to repeatedly examine the values over a period of time.

However, the following must be taken into account:

lm-sensors itself does not perform a stress test simply by reading the data.

### No artificial load required

For a rescue initial diagnosis, a potentially unstable system should not be unnecessarily stressed just to see temperature readings rise.

Especially in the case of:

unexplained crashes,

fan problems,

obvious overheating

or unstable power supply

Secure data recovery takes precedence over a stress test.

### sensors detect

The lm-sensors packet includes:

`sensors-detect`.

This program searches for hardware monitoring chips that are supported by libsensors or the lm-sensors tool collection.

### What `sensors-detect` investigates

The documentary lists the following in this order:

Sensors in CPUs, southbridges and memory controllers,

Sensors in super I/O chips,

Monitoring chips via ISA I/O ports

and:

Monitoring chips via SMBus or I²C.

### Interactive operation

sensors detect

guides the user interactively through the detection steps.

### Automatic operation

With:

```
sensors-detect --auto
```

The default answers to the questions will be used automatically.

### Important safety notice regarding `--auto`

**--auto does not mean "guaranteed safe".**

The program documentation explicitly warns that the internal logic can lead to potentially dangerous hardware probes, even in automatic mode.

### Fundamental risk of `sensors-detect`

, `sensors-detect` needs direct access to hardware.

Since the program cannot be certain which chip is actually present before detection, test accesses can be performed.

The developers explicitly point out that this can cause problems.

Possible consequences of the following are mentioned:

SMBus lockups

up to:

In rare cases, permanent hardware damage may occur.

### Manufacturer warning

The documentation therefore explicitly recommends:

`sensors-detect`on production servers

and:

to avoid forcing detection steps that would be skipped by default unless you know exactly what you are doing.

### Consequences for JUMBO BLOCK Rescue

`sensors-detect` is **not an automatic standard first step**.

The first thing you should always check is:

```
sensors
```

If meaningful values are already available, there is often no reason to perform additional hardware probes.

### Recommended Rescue Procedure

**1. First, just read.**

```
sensors
```

**2. Assess existing values**

Are CPU, motherboard, or other plausible sensors already being displayed?

**3. Only consider whether sensors-detect is necessary if sensor support is actually lacking.**

**4. Consider the risk and importance of the target system**

**5. No forced trial steps without a concrete technical reason**

### Production systems

Especially in the case of a critical server, a critical infrastructure environment, or other important hardware, the developers' advice is particularly relevant.

A rescue tool should not be run simply because it is available.

### Rescue principle

**Diagnostic access must be appropriate to the risk of the system.**

### Kernel modules

If `sensors-detect`detects a supported monitoring chip, a corresponding kernel driver may be required to make its values available via the Linux hardware monitoring system.

### Live system perspective

The following rules apply at JUMBO BLOCK Rescue:

Kernel version,

loaded driver

and:

Hardware support

the started Rescue environment.

A sensor that was available in the installed target operating system will therefore not necessarily appear identically in the rescue environment – and vice versa.

### Missing sensor value

If an expected value is missing, this could mean, for example:

Sensor not present

Sensor not supported by kernel,

Required driver not loaded,

Hardware inaccessible

Sensor ignored by configuration

or:

The platform provides the value in a different way.

### A missing value is not proof of a missing sensor.

Out of:

"lm-sensors does not display this value"

Therefore, it should not be automatically closed:

"This sensor does not physically exist."

## `sensors -s`

With:

`sensors -s`

or rather:

`sensors --set`

`set`instructions contained in the configuration file are evaluated.

Root privileges are required for this.

### Important distinction

`sensors -s` is not a pure read operation.

It can set values or limit parameters on supported hardware according to the configuration.

### Rescue Rule

For a purely diagnostic purpose:

`sensors -s`

**not required.**

It should not be performed routinely.

### Configuration instructions

`sensors.conf`supports, among other things:

`label`

`compute`

`ignore`

and:

`set`.

### `label`

This allows sensor labels to be customized.

### `compute`

This allows raw values to be mathematically converted.

### `ignore`

This allows certain sensor values to be hidden.

### `set`

This allows values or limit parameters to be set on supported hardware.

#### consequence

The displayed sensor label or a calculated value may be affected by the configuration.

This is another reason not to interpret sensor readings without context.

#### Temperature sensor types

The configuration supports different temperature sensor types with suitable chips.

The documentary mentions, among other things:

thermal diodes,

Thermistors,

AMD AMDSI

and:

Intel PECI.

Not every chip supports changing these sensor types.

#### Firmware configuration

The documentation indicates that the BIOS should theoretically configure the sensor type correctly, but this is not always the case in practice.

#### Hysteresis

Temperature limits may also have hysteresis values.

That means:

An alarm can be triggered at a high temperature and only reset at a slightly lower temperature.

This prevents the cooling reaction from constantly switching on and off immediately at the limit value.

### Sensor alarm

Depending on the hardware, sensor values can indicate an alarm status.

Such an alarm should be taken seriously and investigated in the context of the sensor in question.

### An alarm does not automatically mean total loss.

An alarm can be triggered by:

exceeded limit value,

faulty configuration,

implausible sensor input

or an actual hardware problem

point out.

The cause needs to be determined.

### `--bus-list`

With:

```
sensors --bus-list
```

Bus instructions suitable for sensor configuration can be generated.

This is particularly relevant when multiple chips with the same address are present on different buses.

### Bus numbers

The documentation indicates that bus numbers do not necessarily remain stable across restarts.

The bus configuration therefore allows referencing based on bus names instead of exclusively on numerical bus numbers.

### Typical rescue scenario: Computer shuts down unexpectedly

**1.**

```
sensors
```

carry out.

**2.**

Check CPU and, if applicable, motherboard temperatures.

**3.**

Check fan speeds, if available.

**4.**

Pay attention to unusual alarm messages.

**5.**

Compare values with manufacturer specifications and further hardware diagnostics.

**6.**

Do not create unnecessary additional load in case of obvious overheating.

### Typical rescue case: Fan constantly running at high speed

`sensors`

can help determine:

whether high temperatures are reported

and:

which fan speeds are available.

Next, a distinction must be made between possible causes, for example:

actual high temperature,

Cooling problem

Firmware control,

Driver problem

or incorrect sensor interpretation.

### Typical rescue scenario: fan is stopped.

One displayed:

`0 RPM`

Value alone is not sufficient for diagnosis.

Check additionally:

Is the fan physically rotating?

Is zero-RPM operation possible?

Is the correct sensor assigned?

Is the connection monitored by tachometer?

### Typical rescue scenario: unusually high CPU temperature

**1. Check value again**

**2. Pay attention to the package/core assignment**

**3. Check alarm or limit value information**

**4. Control physical cooling**

Fan,

Heat sink

Airflow.

**5. Avoid creating unnecessary stress**

**6. Back up important data if system stability is in question.**

### Typical Rescue case: obviously incorrect voltage

A value that seems technically implausible should not immediately be interpreted as a defective power supply.

First check:

Sensor label

Raw value with `sensors -u`,

Configuration,

driver

and:

Motherboard support.

For reliable electrical measurements, suitable external measuring equipment may be required.

### Typical rescue scenario: no sensors found

If `sensors`find no usable sensor values:

**1. Do not automatically assume a hardware defect**

**2. Determine the hardware model using HWInfo, Inxi, LSHW and dmidecode**

**3. Check kernel/driver support**

**4. Only then decide whether sensor-detect is necessary and justifiable.**

### Typical rescue scenario: thermal control before rescue

A prolonged ddrescue operation can keep computers and storage media running for an extended period of time.

A sensor check before starting may be useful:

```
sensors
```

This makes it possible to determine at least whether unusual system temperatures are already present before the process.

### Data carrier temperature

lm-sensors is not the primary tool for diagnosing the condition of hard drives or NVMe drives.

For data storage media:

**A.8 SMART Diagnosis**

and:

**A.9 NVMe Diagnosis**

decisive.

Depending on the device, these tools can also provide drive-specific temperature data.

### Sensor value and SMART temperature

CPU/motherboard sensors and data storage SMART/NVMe telemetry come from different sources.

They should not be confused with each other.

### lm-sensors and HWInfo

**HWInfo**

Identifies hardware.

**lm-sensors**

displays supported current monitoring values.

### lm-sensors and Inxi

**Inxi**

can include sensor information in its overall report.

**lm-sensors**

is the specialized tool for direct sensor output.

### lm-sensors and LSHW

**LSHW**

shows the hardware structure.

**lm-sensors**

Displays current measurements from supported monitoring hardware.

## lm-sensors and dmidecode

**dmidecode**

displays firmware-side SMBIOS/DMI descriptions.

**lm-sensors**

displays ongoing measurements available via kernel hardware monitoring.

This means the two tools are fundamentally different.

## Four different hardware perspectives

The hardware block of JUMBO BLOCK Rescue thus provides different perspectives:

**HWInfo**

Hardware detection.

**Inxi**

System overview.

**LSHW**

Hardware hierarchy.

**dmidecode**

Firmware description.

**lm-sensors**

current sensor values.

## Writing utilities

The lm-sensors packet includes `sensors`and `sensors-detect`, as well as other low-level tools.

This includes, for example:

`isaset`.

This program can directly modify ISA registers.

## Security warning regarding `isaset`

Such low-level writing tools are **not part of the normal sensor rescue workflow**.

Direct registry access with incorrect parameters can:

System crashes,

Data loss

or further damage

cause.

### `isadump`

`isadump`also accesses ISA registers.

Although it primarily reads registers, its documentation explicitly warns against working with arbitrary parameters in the ISA address space.

### JUMBO BLOCK Rescue Principle

Low-level sensor tools are not used experimentally.

For normal diagnosis, the following applies:

**sensors first.**

### No automatic write operations

Pure sensor diagnostics typically require neither:

`sensors -s`

still:

`isaset`

other direct register changes.

### Limitations

lm-sensors can only display sensors that:

available,

accessible

and:

supported by the running Linux system

become.

Not every computer provides the same sensors.

Not every sensor reading is automatically labelled correctly.

Not every displayed limit value is automatically a universally valid manufacturer's limit.

A single sensor reading is only a snapshot.

lm-sensors does not perform a CPU stress test.

lm-sensors does not perform a RAM test.

lm-sensors does not replace SMART.

lm-sensors does not replace NVMe diagnostics.

lm-sensors does not test power supplies with external measuring technology.

lm-sensors can help indicate physical cooling problems, but it does not fix them.

`sensors-detect`is not risk-free.

### Safety instructions

For initial diagnosis, only the following:

`sensors`

use.

`sensors-detect`if sensor detection is actually required.

On critical production systems, heed the developers' explicit warning regarding `sensors-detect`.

Do not force detection steps that are skipped by default without a specific technical reason.

"car" with "risk-free".

`sensors -s`as a normal read command.

low-level writing tools like `isaset`experimentally.

Verify implausible sensor readings first before inferring hardware defects from them.

Do not perform unnecessary stress tests if the device is clearly overheated.

In case of an unstable system, prioritize backing up important data first.

### Technical classification in JUMBO BLOCK Rescue

With lm-sensors, the hardware block of JUMBO BLOCK Rescue is complete:

**A.29 HWInfo**

What hardware does the system recognize?

↓

**A.30 Inxi**

What does the overall current system configuration look like?

↓

**A.31 LSHW**

How is the hardware structured hierarchically and via buses?

↓

**A.32 DMI Decode**

What information does the BIOS/UEFI provide about the system and hardware?

↓

**A.33 Sensors / lm-sensors**

What current measurement values do supported hardware sensors provide?

These tools complement each other.

A typical diagnostic approach for an unknown or unstable computer might be:

**1. Inxi**

Quick overview.

**2. HWInfo**

Targeted hardware detection.

**3. LSHW**

Device and bus assignment.

**4. dmidecode**

System, board, BIOS and memory information from SMBIOS.

**5. sensors**

Current temperature and monitoring values.

Follow the steps for storage problems:

**SMART Diagnostics**

or rather:

**NVMe diagnostics.**

In the case of a physically unstable storage medium, the following may then be necessary:

**GNU ddrescue.**

The central role of lm-sensors in JUMBO BLOCK Rescue is:

**To visualize the current thermal and sensory state of the running system – without deriving more from a single measurement than what the hardware and its sensors actually show.**

And the Rescue principle applies to sensor detection:

**Read existing values first. Only perform additional hardware probes if they are truly necessary and justifiable for the system in question.**

# A.34 Midnight Commander

**Area:** Files
**Program:** GNU Midnight Commander
**Baseline 20261001:** Midnight Commander 4.8.33
**Installed packages:** `mc 3:4.8.33-1+deb13u1`, `mc-data 3:4.8.33-1+deb13u1`
**Operation:** Text-based two-window interface in the terminal
**Task:** Navigate, view, copy, move, rename, search and edit files and directories, as well as execute shell commands from a common interface

## Purpose

GNU Midnight Commander – short:

**MC**

– is a text-based file manager or a visual shell for Unix-like systems.

The interface is predominantly occupied by two directory windows. This allows two directories to be viewed simultaneously and file operations to be performed between them.

Additionally, Midnight Commander integrates:

a shell command line,

an internal file viewer

and an internal text editor.

## Area of application in the rescue system

Within JUMBO BLOCK Rescue, Midnight Commander is particularly suitable for:

Navigating through mounted file systems,

Examining recovered files,

Copying important data to a backup medium,

Comparing source and target directories,

Renaming and organizing recovered files,

Displaying text and configuration files,

Editing configuration files,

Searching for files and directories,

Working with supported archives,

Executing shell commands

and managing files on locally or otherwise mounted file systems.

### Not data recovery software in the strict sense.

Midnight Commander does not reconstruct lost partition tables.

He does not perform file carving.

It does not read damaged data carriers sector by sector like GNU ddrescue.

It does not recover deleted files from file system metadata.

Its role typically begins where a file system is already accessible or files have already been recovered.

### Two-panel principle

The central workspace consists of two directory windows.

One panel is active at a time.

The other panel can, for example, display the target of a file operation.

This allows a typical rescue situation to be clearly depicted:

**left panel**

Source directory.

**right panel**

Security objective.

Or vice versa.

The Debian man page confirms that Midnight Commander displays two directories simultaneously and that copy and rename operations can, by default, suggest the directory of the other panel as the target. A confirmation prompt appears before the actual operation.

### Program start

Midnight Commander starts with:

```
mc
```

### Start with a directory

A specific directory can be specified directly:

```
mc /pfad/zum/verzeichnis
```

This directory will then appear in the active panel.

### Two starting directories

Two directories can also be specified:

```
mc /quelle /ziel
```

The first directory is displayed in the active panel.

The second directory appears in the other panel.

### Rescue benefits

This allows, for example, a working view to be set up immediately:

```
mc /mnt/quelle /mnt/sicherung
```

The prerequisite, of course, is that these directories actually exist and that the corresponding file systems have been correctly mounted.

### navigation

The arrow keys are used to navigate within a panel.

`Enter`

opens a selected directory.

At:

`..`

one switches to the parent directory.

### Switch between panels

With:

`Tab`

The screen switches between the two panels.

This allows for quick switching between source and destination.

### menu

With:

`F9`

The menu bar will be activated.

The Debian man page explicitly mentions F9 for displaying or activating the menu bar.

### Function keys

Midnight Commander displays the currently important function keys at the bottom of the screen.

The standard basic configuration includes, in particular:

`F1`– Help

`F3`– Display

`F4`– Edit

`F5`– Copy

`F6`– Move or Rename

`F7`– Create directory

`F8`– Delete

`F9`– Menu

`F10`– Exit Midnight Commander.

### Rescue Advantage

This representation makes many file operations possible without having to remember all the shell commands.

### Internal viewer

With:

`F3`

A selected file can be opened in the internal viewer.

The viewer allows you to view files without having to edit them with an editor.

### Rescue principle

If a file is only to be examined initially:

**Viewer before editor.**

This reduces the risk of unintentional changes.

### Start viewer directly

Midnight Commander also supports:

```
mc -v DATEI
```

or rather:

```
mc --view=DATEI
```

to directly start the internal viewer with a specific file.

### Typical rescue applications of the viewer

The viewer is suitable, for example, for:

Log files,

Configuration files,

Text files,

Scripts,

Lists

Reports

and other readable files.

### Internal Editor

With:

`F4`

The selected file can be opened in the configured editor.

Midnight Commander includes:

`mcedit`

its own internal text editor.

### Start the editor directly

The internal editor can also be started directly:

```
mcedit DATEI
```

or via Midnight Commander with:

```
mc -e DATEI
```

or rather:

```
mc --edit=DATEI.
```

### Editing means writing

As soon as a file is saved, its contents are changed.

Therefore, the following applies to rescue operations:

**Do not edit files on evidence-based, original, or damaged source media without a specific reason.**

### Better on a working copy

If a configuration file needs to be changed, the following should be clarified beforehand:

Is the original medium the right working environment?

Is there a fuse?

Can work be done on a copy?

Is the change reversible?

### Copy

`F5`

opens the copy function.

The other panel typically serves as a suggested target.

Before the copying process, the destination is confirmed or can be adjusted.

### Typical Rescue case

Left panel:

`/mnt/beschaedigtes-system/home/benutzer`

Right panel:

`/mnt/backup/rettung`

Then selected files or directories can be:

F5

be copied to the backup medium.

### Copying does not automatically mean data recovery.

If the source file system or storage medium is physically unstable, normal file-by-file copying is not necessarily the best first step.

In case of read errors or failing data carriers, the following should first be checked:

**A.3 GNU ddrescue**

which is the more suitable acquisition method.

### Why?

A file manager works at the file and directory level.

GNU ddrescue works at the block level and was specifically designed for problematic reading media.

### Move

F6

Used for moving or renaming.

### Key Rescue Difference

**Copy**

The source is always preserved.

**Move**

The spring should not be left in its original location after a successful operation.

Therefore, for rescue operations involving valuable original data, the following is usually required:

**Copy before move**

the safer strategy.

### Rename

F6 can also be used to rename files or directories.

This also changes the file system.

### Create directory

With:

F7

A new directory can be created.

This is useful, for example, on a backup medium to store recovered data in a structured manner.

### Delete

With:

F8

Selected files or directories can be deleted.

### Critical safety notice

**Firefighting is fundamentally a high-risk operation in rescue operations.**

Before confirming, the following must be clearly clarified:

Which panel is active?

Which storage device is mounted there?

Which file or directory is selected?

Is this a source or a target for backup?

### No wastebasket as a rescue strategy

With terminal file managers, it cannot be assumed that a deletion operation will safely end up in a recycle bin, as is the case with some desktop file managers.

Therefore, a deletion must be treated as a potentially permanent file system change.

### Select multiple files

Midnight Commander supports selecting multiple files.

This allows file operations to be applied to groups of files.

### Pattern-based selection

Files can also be selected or deselected based on patterns.

This is helpful, for example, if only certain file types need to be backed up.

### Rescue caution when selecting large numbers

Before:

Copy,

Move

or in particular:

Delete

With a large selection, the number of selected files should be checked again.

### File search

Midnight Commander has an integrated search function.

This allows you to search for files within a directory tree.

### Content search

The search functions can also be used to search for text content in files.

This can be helpful, for example, when searching for:

Configuration values,

Hostnames,

Error messages,

Log entries

or specific strings.

### Rescue benefits

On an accessible but unknown Linux system, this can be used, for example, to specifically search for:

Configuration files,

Logs,

documents

or project-specific files

are being sought.

### Search is not file carving

Midnight Commander searches within the accessible file system structure.

Deleted files that are no longer present in the directory tree are not automatically reconstructed.

Depending on the situation, the following are needed:

TestDisk,

PhotoRec,

Foremost

or:

Extundelete

responsible.

### Shell command line

An important component of Midnight Commander is the integrated shell command line.

Text that was entered there and used with:

`Enter`

Once confirmed, it will be executed as a shell command.

### Importance for Rescue

This allows file managers and the command line to be combined.

For example, you can first navigate visually to a directory and then execute a diagnostic command there.

### Caution

The shell command line has the rights of the started Midnight Commander process.

A destructive command entered there remains destructive.

Midnight Commander does not automatically make shell commands safer.

### Subshell

Midnight Commander supports an integrated or parallel subshell, provided this feature was enabled during the build.

Version 4.8.33 brought in particular improvements to subshell support for bash, ash/dash and fish, as well as initial support for ksh variants.

### Rescue meaning

This allows switching between file manager and shell work without losing the entire file context.

### Archives and virtual file system

Midnight Commander uses a VFS concept – a virtual file system.

This allows supported archives and other data sources to be partially displayed as if they were directories.

### Archive

Supported archive types can thus be examined directly from the file manager interface, provided that the necessary VFS support or external utilities are available.

### Do not confuse with a real mount

An archive structure opened in Midnight Commander is not automatically a file system that is regularly mounted in the Linux file system.

Other programs do not necessarily see this internal VFS view as a normal mount point.

### Do not unnecessarily modify the archive.

In rescue and analysis work, archives should initially only be viewed or extracted onto a working medium.

Changes to the original archive should be avoided if its original state is to be preserved.

### File types and actions

Depending on the file type and configuration, Midnight Commander can associate certain actions with files.

`Enter`on a file can, depending on:

File type,

Feasibility

and configuration

an action or an external program will be started.

### Rescue Rule

For unknown or potentially harmful files:

**Do not execute it carelessly by pressing Enter.**

First, let's examine:

`F3`

use.

### Executable files

If the selected file is executable, pressing Enter will execute it.

This is particularly relevant when investigating a potentially compromised system.

### Security rule for suspicious files

**Do not run suspicious programs, scripts, or binaries for analysis.**

The mere existence of a file is no reason to start it.

### Hidden files

Midnight Commander can also display hidden Unix files and directories.

This is important for rescue operations, as numerous relevant configurations in names begin with:

.

Examples include user configurations within home directories.

### File permissions and metadata

Midnight Commander can retrieve Unix file information such as:

Permissions,

Owner,

Group,

Size

and time information

to display or – depending on the operation and rights – to edit.

### Rescue meaning

During a recovery process, it can be important to restore not only file contents, but also:

Owner,

Permissions,

Timestamp

and symbolic links

to be taken into account.

### A copy is not automatically forensically identical.

A normal file copy using a file manager is not the same as a forensic block image.

Depending on:

file system,

Copying method,

Mounting options,

Permissions

and target system

Metadata can be handled differently.

### Forensic evidence preservation

If an unaltered, bit-perfect backup of the source medium is required, Midnight Commander is not the appropriate acquisition tool.

This requires a block image or a suitable forensic workflow.

### Read-only source

For rescue operations, it is often useful to select a file system to be examined:

**read-only**

to integrate.

Midnight Commander can then be used for navigation and reading, while write operations on this mount are blocked by the file system or the mount itself.

### Important limitation

Midnight Commander itself does not make a file system read-only.

Protection must be provided at the level of:

Mounts,

file systems,

Block device

or other suitable protective measures

to be realized.

### Read-only and backup target

A typical safe setup is as follows:

**Source medium**

embedded in read-only format.

**Target medium**

read-write functionality is included.

**Midnight Commander**

Left and right panels for source and destination.

### Check device identity first

Before any major copying operation, the following must be clearly clarified:

Which mount point belongs to the source medium?

Which one goes to the target medium?

Is there enough free storage space available on the target?

Is the goal actually describable?

### Don't blindly trust path names

A directory name like:

`/mnt/backup`

This does not prove that the desired backup medium is actually installed there.

The mount situation must be checked before critical operations.

### Free space

Before performing extensive copy operations, the free capacity of the destination should be checked.

A target medium that fills up during the recovery process can interrupt a copying operation and make it difficult to maintain an overview.

### Internal editor mcedit

The editor provided with Midnight Commander can be started independently of the file manager:

`mcedit`

Version 4.8.33 has its own program documentation for this.

### Syntax highlighting

mcedit supports syntax definitions.

According to the Debian documentation, system-wide syntax files are located at:

`/usr/share/mc/syntax/`.

User-specific editor information is located at:

`~/.local/share/mc/mcedit/`.

### configuration

The user-specific Midnight Commander configuration is typically located at:

`~/.config/mc/`.

The Debian documentation mentions, among other things:

`~/.config/mc/ini`

for user-specific configuration.

### Rescue significance of user configuration

Changes to the MC configuration fundamentally alter the working environment of Midnight Commander.

This does not automatically change the source files being examined.

This is distinct from file operations performed within MC on a mounted target system.

### Mouse support

Midnight Commander supports mouse operation in the terminal.

With:

```
-d
```

or rather:

```
--nomouse
```

Mouse support can be disabled.

### black and white representation

With:

```
mc -b
```

or rather:

```
mc --nocolor
```

A black and white representation is forced.

### Color representation

With:

```
mc -c
```

or rather:

```
mc --color
```

The color display can be forced.

### View version

With:

```
mc -V
```

or rather:

```
mc --version
```

The version number is displayed. The program documentation explicitly recommends including this information in error reports as well.

### Help

Midnight Commander has a built-in help function.

Within the program, she is referred to as:

`F1`

reachable.

### Typical rescue scenario: Backing up user files

**1. Mount the source medium in read-only mode if possible.**

**2. Mount the backup medium in read-write mode.**

**3. Check device and mount point assignment.**

**4. Start Midnight Commander.**

Example:

```
mc /mnt/quelle /mnt/backup
```

**5. Visually check the source and destination again.**

**6. Select the required files.**

**7. Copy using F5.**

**8. Check the copy result.**

**9. Leave the source unchanged.**

### Typical rescue scenario: Examine the configuration file

**1. Navigate to the target system.**

**2. Select the file.**

**3. Use F3.**

**4. Read the content.**

**5. Only use F4 if a change is explicitly necessary, and only in a suitable working environment.**

### Typical rescue scenario: Repair configuration

For example, if a damaged text configuration needs to be corrected:

**1. Secure the original.**

**2. Check whether the target system should actually be describable.**

**3. Edit the file on a working copy if necessary.**

**4. Make the change using mcedit.**

**5. Check the result.**

**6. Do not overwrite the original file without a backup.**

### Typical rescue scenario: sorting rescued files

PhotoRec or Foremost can generate large quantities of reconstructed files.

Midnight Commander can then:

Navigate,

Show,

Rename,

Sort

and copying

These results will help.

### Important limitation

Midnight Commander does not improve the quality of a PhotoRec or Foremost reconstruction.

It merely manages the results that already exist.

### Typical rescue scenario: Back up logs

A sensible approach might be:

**1. Mount the source system in read-only mode.**

**2. Open the log directory in one panel.**

**3. Open the backup directory in the other panel.**

**4. Copy relevant logs using F5.**

**5. Then analyze the copies.**

### Typical Rescue case: Remote files

For example, if a remote file system is connected to:

**A.28 SSHFS**

Once mounted into the local file system, Midnight Commander can use the mount point like any other accessible directory.

### Security limit

Midnight Commander adopts the properties of the underlying mount.

A read-only SSHFS mount remains read-only for normal file operations.

In contrast, a remote mount configured for read-write access can be modified through file operations.

### Midnight Commander and SSH

Midnight Commander can also be used in a terminal via an SSH connection.

The project explicitly states that the text-based interface can be used on a normal console, in graphical terminals, and via SSH or other remote shells.

### Midnight Commander and GNU ddrescue

The roles are fundamentally different:

**GNU ddrescue**

Block level and acquisition of problematic media.

**Midnight Commander**

File and directory level.

A typical sequence of events might therefore be:

**damaged data carrier**

↓

**ddrescue image**

↓

**Make the file system or reconstructed data accessible**

↓

**Midnight Commander**

Review files and organize them onto a target medium.

### Midnight Commander and TestDisk

**TestDisk**

reconstructs or examines partition and certain file system structures.

**Midnight Commander**

manages accessible files.

### Midnight Commander and PhotoRec/Foremost

**PhotoRec / Foremost**

Reconstruct files using File Carving.

**Midnight Commander**

It then helps with reviewing and organizing the results.

### Midnight Commander and Extundelete

**Extundelete**

Attempts to recover deleted files from ext file systems.

**Midnight Commander**

The recovered files are then managed at the file system level.

## Midnight Commander and GParted

**GParted**

It works with partitions and file system structures.

**Midnight Commander**

works with files and directories within accessible file systems.

## Midnight Commander and GNOME Disks

**GNOME Disks**

It manages and examines drives, partitions, and images.

**Midnight Commander**

is responsible for file operations within accessible directory structures.

## Midnight Commander and ClamAV

Files located or recovered with Midnight Commander can then be:

**ClamAV**

be examined.

The following still applies to potentially compromised files:

**do not execute.**

## Midnight Commander and Binwalk / Hexedit

**Binwalk**

analyzes binary data and embedded structures.

**Hexedit**

works at the byte level.

**Midnight Commander**

In contrast, it primarily serves for navigation and management of the files in question.

## Limitations

Midnight Commander is not a file carving tool.

Midnight Commander is not a partition recovery program.

Midnight Commander is not a block imager.

Midnight Commander is not a SMART diagnostic tool.

Midnight Commander does not check file system integrity.

Midnight Commander does not guarantee forensic metadata preservation.

Midnight Commander does not automatically make a source read-only.

Midnight Commander does not prevent user errors.

Midnight Commander makes shell commands not entirely safe.

Midnight Commander does not automatically detect if the source and destination have been mixed up.

## Safety instructions

Check the active panel before each write operation.

Clearly identify the source and destination.

When performing rescue operations, always prefer **F5 Copy** to **F6 Move** if the original data is to be preserved.

**F8 Delete** only after thorough verification.

Include original sources in read-only format whenever possible.

Do not start suspicious executable files with Enter.

Use F3 instead of F4 for the initial examination of a file.

Create a backup before editing important configurations.

Do not assume that a normal file copy constitutes a forensic image.

Do not repeatedly search large file trees with a file manager on physically unstable storage media if a ddrescue image is required first.

Check free storage space and mount points before large copy operations.

## Technical classification in JUMBO BLOCK Rescue

Midnight Commander forms a separate area within Block A:

**Files.**

Its role differs significantly from previous diagnostic and recovery tools.

A complete rescue workflow might look like this, for example:

**Identify hardware**

HWInfo / Inxi / LSHW / dmidecode

↓

**Assess the data carrier's condition**

SMART / NVMe

↓

**Secure unstable medium**

GNU ddrescue

↓

**Examine partition or file system structures**

TestDisk / Extundelete

↓

**Reconstruct files if necessary**

PhotoRec / Foremost

↓

**Manage and back up files**

**Midnight Commander**

This allows MC to take over the practical file and directory work within an already accessible rescue environment.

Its two-panel structure is particularly suitable for clearly comparing:

**source**

and:

**Security objective.**

But this very strength also brings responsibility:

An incorrect panel, a misidentified mount point, or confusing F5, F6, and F8 can alter or delete original data.

The central rescue principle for Midnight Commander is therefore:

**First clearly define the source and destination – then copy. Move, edit, or delete only if the change is explicitly intended.**

# A.35 QPDF

**Area:** Documents
**Program:** QPDF
**Baseline 20261001:** QPDF 12.2.0
**Installed package:** `qpdf 12.2.0-1`
**Associated Library:** `libqpdf30 12.2.0-1`
**Operation:** Command line
**Task:** Structurally check PDF files, rewrite them for repair, and selectively transform them

## Purpose

QPDF is a command-line tool for the technical processing of PDF files.

The JUMBO BLOCK Rescue System focuses on:

Review of problematic or recovered PDF files,

Detection of structural PDF errors,

Rewriting damaged PDF structures,

Examination and processing of encrypted PDFs,

Selection and compilation of individual pages

as well as:

Splitting larger PDF documents.

QPDF is neither a PDF viewer nor a traditional document editor. It edits the internal structure of a PDF file.

## Area of application in the rescue system

QPDF is typically used after a PDF file has already been backed up or reconstructed from a data carrier, file system or rescue image.

A typical sequence of events might be:

**Recover data from the storage device**

↓

**Restore file**

↓

**Technically check PDFs with QPDF**

↓

**Create a repaired working copy if necessary**

↓

**Visually check the result using a PDF reader.**

### Check PDF

The most important diagnostic command is:

```
qpdf --check DATEI.pdf
```

QPDF examines the internal PDF structure and reports any detected errors or warnings.

This is especially useful if a PDF:

cannot be opened

after a data recovery appears suspicious,

PDFs are processed differently by different PDF readers.

or:

Error messages are generated when opening.

### Correctly assessing warnings

A warning does not automatically mean that the document is unusable.

QPDF has recovery mechanisms and can reconstruct certain damaged PDF structures during import.

Conversely, a successful structural audit does not prove that every page is complete in terms of appearance and content.

Therefore:

**Technical inspection with QPDF and visual inspection with a PDF reader complement each other.**

### Rewrite damaged PDF

If QPDF can still read a damaged file, it can write it as a new PDF file.

Example:

```
qpdf problem.pdf recovered.pdf
```

The document structure interpreted by QPDF is then re-output.

The new file should then be checked again:

```
qpdf --check recovered.pdf
```

### Original preserved

The damaged original file must not be unnecessarily overwritten during a rescue operation.

It is recommended that:

```
problem.pdf
```

→ Unaltered original

`recovered.pdf`

→ Working or repair version.

The central principle is:

**Original retained – repair attempt written to a new file.**

### Limits of repair

QPDF can only process information that is still present in the recovered file or that can be reconstructed from its structure.

For example, if actual file areas are missing after an incomplete data recovery, QPDF cannot recreate this content.

QPDF therefore does not repair every damaged PDF.

### Check page number

With:

```
qpdf --show-npages DATEI.pdf
```

The number of pages can be displayed.

This is helpful, for example, after a recovery process, to compare the reconstructed file with the expected document size.

However, the page number alone does not prove the completeness of the page content.

### Select and compile pages

QPDF can be used with:

`--pages`

Select specific pages, rearrange them, and combine pages from multiple PDF files into a new document.

This can be useful in rescue operations if:

only certain pages are needed

Individual pages of a document are to be saved.

or:

usable pages from multiple files should be merged into a new PDF.

The result should generally be written to a new file.

### Split PDF

With:

`--split-pages`

Can a PDF be split into multiple files?

This can be helpful, for example, when a large document needs to be divided into smaller units or when a problem needs to be limited to specific page areas.

### Encrypted PDFs

QPDF can detect and display whether and how a PDF is encrypted.

These include functions for:

Checking the encryption status,

Display of encryption information

and:

Processing with existing access data

available.

### Decode

With:

`--decrypt`

With authorized access, an unencrypted working copy can be created.

The following should be noted:

The new file will then no longer have the previous PDF encryption protection.

It must therefore be stored securely.

### No password cracker

QPDF is not a tool for brute-force cracking unknown PDF passwords.

In the rescue system, it is only used with existing authorization and the required access data.

### Embedded files

PDF documents can contain file attachments.

QPDF can examine and extract such embedded files.

This can be relevant for document recovery if a PDF contains additional files.

### Safety rule for attachments

Extracted unknown files must not be opened or executed automatically.

If their origin is suspicious or unknown, they may also be accompanied by:

**ClamAV**

be examined.

### Digal signatures

Rewriting a PDF changes its file structure and usually also its hash value.

For digitally signed documents, this can have an impact on signature validation.

Therefore:

**Digally signed original PDFs are always preserved unchanged.**

Perform analysis or repair exclusively on a copy.

QPDF does not replace standalone cryptographic signature verification.

### Typical Rescue case: PDF cannot be opened

**1. Secure the original.**

**2. Check the PDF:**

```
qpdf --check problem.pdf
```

**3. Document warnings and errors.**

**4. If QPDF can read the file, create a new working copy:**

```
qpdf problem.pdf recovered.pdf
```

**5. Check the working copy again:**

```
qpdf --check recovered.pdf
```

**6. Visually check recovered.pdf with a PDF reader.**

**7. Keep the original unchanged.**

### Typical rescue scenario: PDF from PhotoRec or Foremost

A reconstructed PDF can first be used with:

```
qpdf --check DATEI.pdf
```

be examined.

Then you should:

the page number was checked

the document is visually checked

and, if structural problems arise, a newly written working copy may be created.

QPDF can only work with the data that PhotoRec or Foremost has actually recovered.

### Typical rescue scenario: saving individual pages

If only certain pages of a still processable document are needed:

Original received.

↓

Select the required pages using `--pages`.

↓

Create a new PDF.

↓

Check new files with `--check`.

↓

Visually check the result.

### Typical rescue scenario: encrypted PDF

Determine encryption status.

↓

Use existing login credentials.

↓

Create a decrypted working copy if necessary.

↓

Protect your working copy appropriately.

The encrypted original remains intact.

### QPDF and other rescue tools

**GNU ddrescue**

Saves data at the block level.

**TestDisk / Extundelete**

working on partition and file system structures and recovery.

**PhotoRec / Foremost**

Reconstruct files using File Carving.

**Midnight Commander**

manages recovered files and directories.

**PDF**

It then examines the internal structure of a PDF file.

This means that QPDF is typically at the end of the technical rescue chain.

## Limitations

QPDF is not a PDF viewer.

QPDF is not an OCR system.

QPDF is not a classic PDF content editor.

QPDF cannot regenerate missing file data.

QPDF does not repair every damaged PDF.

QPDF does not automatically check the visual completeness of all pages.

QPDF is not a password brute-force tool.

QPDF is not an antivirus program.

QPDF does not replace cryptographic signature verification.

A newly written PDF is not byte-identical to the original.

## Safety instructions

Back up the original files before any transformation.

check damaged PDFs using the `--check` option.

Write repair attempts to a new file.

Clearly separate the original and the working copy.

Digally signed originals preserved unchanged.

Protect decrypted working copies accordingly.

Do not automatically execute unknown embedded files.

After structural repairs, visually inspect the document as well.

A successful QPDF check should not be confused with a security check of the document.

## Technical classification in JUMBO BLOCK Rescue

QPDF covers the area:

**Documents**

and thus the conclusion of Block A.

A typical rescue route might be:

**Back up data storage devices**

↓

**Restore file system or files**

↓

**Manage recovered files with Midnight Commander**

↓

**Check problematic PDFs with QPDF**

↓

**Create a repaired working copy if necessary**

↓

**Visually check the result**

QPDF thus takes on a clearly defined task:

**After rescuing PDF documents, technically examine them and, as far as possible, consistently output their internal structure in a new working copy.**

The central Rescue principle is:

**Receive the original – check the PDF structure – perform the repair only on a working copy – then check the result technically and visually.**

# Block B

Technically, JUMBO BLOCK Rescue is based on MX Linux, but it is designed as a **cross-operating-system rescue system**.

The MX base provides numerous system and administration tools. This manual therefore covers only those MX tools that are specifically useful for rescue tasks, working with bootable media, or securely transferring and verifying system states and ISO images.

Purely administrative and convenience features of the MX base system are outside the scope of the detailed rescue documentation.

## Tool groups

**Six relevant MX tools:**

**B.1 – MX Boot Repair**
Repair of GRUB-based Linux boot environments and regeneration of the GRUB configuration and initramfs.

**B.2 – UEFI Manager**
Checking and management of UEFI boot entries and boot orders for diagnosing and correcting boot problems.

**B.3 – Chroot Rescue Scan**
Detection of installed Linux systems and provision of a chroot environment for targeted repair work on installations that no longer boot normally.

**B.4 – MX Live USB Maker**
Creation and duplication of bootable USB media from ISO images or existing live systems.

**B.5 – MX Snapshot**
Creation of bootable ISO images from a prepared system state. Selected rescued files or files to be examined can be transferred into such a snapshot in a controlled manner and backed up or transferred together with the rescue system.

**B.6 – ISO Signature Verifier**
Operating-system-independent integrity and authenticity checking of ISO images on the basis of suitable signature and checksum information.

The six tools thus complement the specialised programs from **Block A – JB-Rescue** without shifting the focus of the manual to the underlying MX Linux distribution.

**JUMBO BLOCK Rescue uses MX Linux as its technical foundation, but its scope is not limited to MX Linux or Linux systems.**

# Assignment of areas, programs and versions

| No. | Area | Program | Baseline version |
|---|---|---|---|
| **B.1** | Boot repair | MX Boot Repair | 26.06.1 or later |
| **B.2** | Boot / UEFI | UEFI Manager | 26.07.1 or later |
| **B.3** | Linux system rescue | Chroot Rescue Scan | 0.2.6+5 or later |
| **B.4** | Rescue medium | MX Live USB Maker | 26.04.1 or later |
| **B.5** | System image / ISO creation | MX Snapshot | 26.08 or later |
| **B.6** | Integrity / authenticity | ISO Signature Verifier | 2026.09.01 or later |

# B.1 MX Boot Repair

**Area:** Boot repair
**Program:** MX Boot Repair
**Baseline 20261001:** 26.06.1
**Operation:** graphical user interface
**Task:** repair of GRUB-based Linux boot environments

## Purpose

MX Boot Repair is a graphical tool for repairing certain boot problems of Linux systems.

In the JUMBO BLOCK Rescue system, the program is particularly useful for installed Linux systems that use **GRUB** as their boot loader.

The essential functions are:

- regenerating the GRUB configuration file,
- reinstalling the GRUB boot loader,
- regenerating damaged or missing initramfs images.

## Significance for JUMBO BLOCK Rescue

JUMBO BLOCK Rescue is not restricted to repairing MX Linux.

MX Boot Repair is therefore to be classified as a **specialised Linux/GRUB tool within the universal rescue system**.

It can be helpful if an installed Linux system no longer boots because of a damaged or faulty GRUB configuration.

By contrast, it is **not a universal repair tool for the boot loaders of all operating systems**.

## Repairing the GRUB configuration

The following function regenerates the GRUB configuration of the selected Linux system:

**Repair GRUB configuration file**

MX recommends this function as the first repair attempt for corresponding GRUB problems.

A possible fault pattern is, for example:

```
grub rescue>
```

The function requires GRUB itself to be basically installed in the intended location.

If the actual boot loader is damaged or no longer present, GRUB may first have to be reinstalled.

## Reinstalling the GRUB boot loader

MX Boot Repair can reinstall GRUB.

Depending on the boot method used, different targets are provided for this:

**ESP**

EFI System Partition on UEFI systems.

**MBR**

Master Boot Record on conventional BIOS/legacy systems.

**PBR**

Partition Boot Record, or boot sector of a partition.

After GRUB has been reinstalled, the MX documentation recommends then also having the GRUB configuration regenerated.

### UEFI systems

For a UEFI repair with MX Boot Repair, the rescue system itself must have been started in **UEFI mode**.

During the repair, the following in particular must be identified unambiguously:

- the correct EFI system partition,
- the correct Linux root partition

and, where applicable:

- a separate `/boot` partition.

The program's automatic preselection must not be adopted without checking for complex installations. The MX documentation expressly points this out.

### Legacy/MBR systems

With conventional BIOS/legacy installations, GRUB can be reinstalled in the MBR of the affected storage device.

Here too, the target device must be identified unambiguously.

For a legacy repair, JUMBO BLOCK Rescue must have been started in legacy/MBR mode accordingly.

## Regenerating initramfs

MX Boot Repair also offers:

**Regenerate initramfs images**

This allows missing or damaged initramfs images in the `/boot` area of a Linux system to be regenerated.

During Linux startup, the initramfs provides the early user environment in which, among other things, the actual root file system is found and prepared.

A damaged or missing initramfs can therefore prevent a Linux system from starting completely.

### Typical rescue workflow

For a GRUB-based Linux system that no longer starts:

**1. Start JUMBO BLOCK Rescue.**

**2. Identify storage devices and partitions unambiguously.**

**3. Check whether the target system uses UEFI or legacy/BIOS.**

**4. Determine the root, EFI and, if applicable, /boot partition unambiguously.**

**5. First repair the GRUB configuration if the fault pattern matches.**

**6. Reinstall GRUB if necessary.**

**7. Then regenerate the GRUB configuration again.**

**8. If the initramfs is damaged or missing, regenerate it.**

**9. Restart the system and check the boot process.**

### Dual-boot and multi-boot systems

Particular care is required with systems on which several operating systems are installed.

An EFI system partition can contain boot files of several operating systems.

In addition, only the UEFI boot order may have changed, although the actual boot loaders are still present.

The MX documentation expressly mentions the case in which Windows updates can place the Windows Boot Manager in front of an existing Linux entry in the UEFI boot order. In such a case, the UEFI configuration should be checked first before GRUB is reinstalled.

**B.2 – UEFI Manager** is available in JUMBO BLOCK Rescue for this task.

### Limitations

MX Boot Repair is geared to **Linux/GRUB-related repair tasks**.

In particular, it is:

not a general Windows boot repair tool,

not a macOS boot repair tool,

not a file system repair program,

not a partition recovery program

and:

not a data recovery program.

A Windows system on the same computer can be relevant for assessing the boot configuration. However, this does not mean that MX Boot Repair repairs the Windows boot loader itself.

## Safety notes

Boot loader repairs are writing interventions in the boot configuration of a storage device.

Before proceeding:

**Target storage device, root partition, EFI system partition and boot method must be identified unambiguously.**

With a physically unstable or damaged storage device, data backup or image creation takes priority.

With dual-boot and multi-boot systems, it must be borne in mind that several operating systems can use the same EFI system partition.

A merely incorrect UEFI boot order should not be “repaired” by an unnecessary reinstallation of the boot loader.

Automatically suggested target partitions must be checked before writing.

## Interaction with other rescue tools

### UEFI Manager

This checks and manages UEFI boot entries and the boot order.

### Chroot Rescue Scan

This enables access to installed Linux systems via a chroot environment if further repairs are required.

### GParted / GNOME Disks

These help to identify storage devices and partitions.

### SMART diagnostics / NVMe diagnostics

These check the condition of the underlying storage device.

### GNU ddrescue

This backs up an unstable or damaged storage device before further repair measures are carried out.

### TestDisk

This is responsible for damaged or lost partition structures and thus performs a different task from MX Boot Repair.

### Technical classification in JUMBO BLOCK Rescue

MX Boot Repair belongs to:

**Block B – MX tools → Boot repair**

Its significance within JUMBO BLOCK Rescue is clearly limited:

**It extends the universal rescue system with a graphical repair option for GRUB-based Linux installations.**

Other operating systems and other boot loaders may require their own repair procedures.

The central rescue principle is:

**First identify the boot method and target partitions unambiguously – only repair components that are actually damaged – with unstable storage devices, back up the data before any boot loader repair.**

Official MX documentation: MX Boot Repair

# B.2 UEFI Manager

**Area:** Boot / UEFI
**Program:** UEFI Manager
**Baseline 20261001:** 26.07.1
**Operation:** graphical user interface
**Task:** display and manage UEFI boot entries

## Purpose

UEFI Manager is a graphical tool for managing the boot entries stored in the UEFI firmware area of a computer.

The **Manage UEFI Entries** function is particularly useful for JUMBO BLOCK Rescue. It can be used to establish which boot options are present and in which order they are used.

## Area of use in the rescue system

UEFI Manager can help with a supposed boot problem before a boot loader is actually repaired or reinstalled.

An operating system may still be fully intact, including its boot loader, even if the computer selects the wrong UEFI entry.

The MX documentation on Boot Repair therefore expressly recommends checking the UEFI configuration first in the case of corresponding dual-boot problems.

## Prerequisite

UEFI Manager requires a UEFI system and must be run in an environment that has itself been started in **UEFI mode**.

As a way of checking this, the MX documentation names the presence of:

`/sys/firmware/efi/efivars`

An accessible EFI system partition must also be present.

## Displaying existing boot entries

UEFI Manager displays the existing UEFI boot entries in their current order.

Among other things, the following can be identified:

- active and inactive entries,
- the boot entry currently in use,
- the boot order,
- a one-time next boot entry, if set.

This first establishes whether the required boot entry is still present.

### Changing the boot order

Existing entries can be moved up or down within the boot order.

This is relevant, for example, if a different boot loader is suddenly started in preference after a change to the computer.

An incorrect boot order must be distinguished from an actually damaged boot loader.

### Boot Next

**Boot Next** can be used to select a specific UEFI entry for the next restart only.

The normal boot order then applies again.

This is particularly useful during rescue work:

An existing boot entry can be tested first **without immediately changing the permanent boot order**.

### Enabling and disabling entries

Existing UEFI entries can be enabled or disabled.

A disabled entry remains present but is not used as an active boot entry.

For diagnostic purposes, disabling is therefore basically less final than deleting an entry.

### Adding and removing entries

UEFI Manager can also create new boot entries for existing EFI files and remove existing entries.

Removing an entry is a writing change with potentially significant consequences.

Entries should therefore not be deleted merely because their function is initially unknown.

In multi-boot systems in particular, several entries can belong to different operating systems or recovery environments.

### Typical rescue scenario: wrong operating system starts

A computer has several operating systems or boot loaders but suddenly no longer starts via the expected entry.

Procedure:

**1. Start JUMBO BLOCK Rescue in UEFI mode.**

**2. Open UEFI Manager.**

**3. Check the existing boot entries and their order.**

**4. Establish whether the expected entry is still present and active.**

**5. If possible, test first with Boot Next.**

**6. Only after a successful check, change the permanent boot order if necessary.**

This can avoid unnecessarily reinstalling a working boot loader.

### Typical rescue scenario: dual boot

On a system with Windows and Linux, for example, several UEFI entries can be present.

The MX documentation points out that Windows updates can change the boot order and place the Windows Boot Manager in front of an existing Linux entry. With a corresponding fault pattern, the UEFI order should therefore be checked first.

This does not automatically mean that GRUB or another boot loader has been damaged.

### EFI Stub Installer

UEFI Manager additionally contains an **EFI Stub Installer** for Linux systems. This allows a Linux kernel to be started directly via a UEFI entry without an intervening boot loader such as GRUB.

For the general use of JUMBO BLOCK Rescue, this is a special function and not a regular rescue measure.

### Frugal EFI Stub Installer

The program also contains a special EFI stub installer for frugal MX/antiX installations.

This function concerns the MX/antiX infrastructure and is not central to the universal operating system rescue of JUMBO BLOCK Rescue.

It is therefore not covered further here.

### Limitations

UEFI Manager:

- does not repair file systems,
- does not restore partitions,
- does not repair the content of a damaged boot loader,
- does not replace a data backup,
- is intended for UEFI systems and not for conventional BIOS/legacy boot configurations.

The tool manages the **UEFI boot configuration**.

If, for example, GRUB itself is damaged, a tool such as **MX Boot Repair** is then required.

### Safety notes

Document the existing boot order before making changes.

Do not delete unknown entries hastily.

On multi-boot systems, pay particular attention to entries of other operating systems.

If possible, use **Boot Next** first before changing the permanent order.

An existing EFI file or boot entry does not automatically prove that the associated operating system is fully functional.

Deleting a UEFI entry is not the same as deleting the operating system, but it can prevent its regular startup.

### Technical classification in JUMBO BLOCK Rescue

UEFI Manager belongs to:

**Block B – MX tools → Boot / UEFI**

Unlike MX Boot Repair, the tool does not primarily repair the Linux boot loader.

Its central rescue task is:

**To examine existing UEFI boot entries and to identify and correct an incorrect boot selection or boot order.**

This makes UEFI Manager relevant for computers with different operating systems as well, because the UEFI firmware decides, above the operating relevant system, which configured EFI boot path is used.

The central rescue principle is:

**First check whether merely the wrong UEFI entry is being started – only then repair or reinstall a boot loader.**

Official MX documentation: UEFI Manager

# B.3 Chroot Rescue Scan

**Area:** Linux system rescue
**Program:** Chroot Rescue Scan
**Baseline 20261001:** 0.2.6+5
**Operation:** terminal / menu-driven selection
**Task:** detect installed Linux systems and open them in a chroot environment for repair work

## Purpose

Chroot Rescue Scan enables access to an installed Linux system even though it can no longer be started normally itself.

The tool searches existing partitions for Linux systems, mounts detected systems and then offers a choice of target system for a chroot session.

This allows repair commands to be run within the directory and system environment of the installed Linux installation.

## Significance for JUMBO BLOCK Rescue

Chroot Rescue Scan is one of the MX tools of direct relevance to rescue work.

It is not geared to repairing the running JUMBO BLOCK Rescue environment, but enables access to **other installed Linux systems**.

The MX documentation particularly emphasises that this can make access to an installed system possible even if its `initrd.img` is damaged. In addition, several installed Linux systems can be examined without having to reboot for each system.

## What is a chroot environment?

For a process and its child processes, `chroot` changes the perceived root directory `/`.

If, for example, the root partition of an installed Linux system is mounted and then used as a chroot, its directories such as the following appear in their normal positions again within the chroot:

`/etc`

`/usr`

`/var`

`/boot`

The same applies to other system directories.

This allows many administrative repair commands to be run as if work were being carried out within the installed system.

## Automatic search

`chroot-rescue-scan` searches the existing partitions for Linux systems.

By default, systems found are mounted under:

`/media`

The tool then provides a menu via which one of the Linux systems found can be selected for the chroot session.

### Several Linux installations

If there are several Linux systems on a computer, they can be detected and offered separately.

This is helpful, for example, with:

dual-boot systems,

multi-boot systems,

test installations

or:

several Linux installations on different storage devices.

Before a repair begins, it must be established unambiguously **which detected system is actually to be worked on**.

### Preparing the chroot environment

The underlying `chroot-rescue` mounts the system areas required for the target system, such as:

`/sys`

`/proc`

and:

`/dev`

Temporary file systems are provided for `/run` and `/tmp`. The program then changes to the selected root directory.

This gives the installed system access to important interfaces of the running rescue kernel during the repair.

### Shell in the target system

If no particular command is specified, Chroot Rescue first tries to use the following shell in the target system:

`/bin/bash`

and then:

```
/bin/sh
```

After a successful change, the administrator is in a shell whose root directory is the installed Linux system.

This is an important distinction:

**Commands within the chroot can change the installed target system.**

### Typical rescue tasks

Within a chroot environment, the following repair work can be carried out, for example – depending on the Linux relevant distribution and its tools:

- correcting configuration files,
- repairing damaged or missing packages,
- editing boot configurations,
- regenerating initramfs,
- correcting user or permission problems,
- examining incorrect entries in `/etc/fstab`,
- running distribution-specific maintenance commands.

Which commands are actually suitable depends on the **installed target system**.

Chroot Rescue Scan does not carry out these repairs automatically. It provides the environment in which they can be carried out.

### Network access

Chroot Rescue can try to provide network access for the target environment.

The tool family also has an option that suppresses this attempt.

Network access is required, for example, if packages from the repositories of the installed distribution are to be installed or repaired within the chroot.

However, a network connection should only be established if it is required for the specific repair.

### Damaged initrd

An essential rescue use case is an installed Linux system that can no longer start because of a damaged initrd or initramfs.

As Chroot Rescue Scan mounts the installed system from the JUMBO BLOCK Rescue system that is already running, the faulty target system does not have to boot successfully itself.

Within the chroot, suitable tools for regenerating the initramfs can then be run – provided that the relevant distribution supports this and the rest of the installation is sufficiently intact.

## Typical rescue workflow

**1. Start JUMBO BLOCK Rescue.**

**2. Check the condition of the affected storage device.**

If there are indications of a physically unstable storage device, back it up or create an image first.

**3. Start Chroot Rescue Scan.**

**4. Check the Linux systems found.**

**5. Select the correct target system unambiguously.**

**6. Open the chroot environment.**

**7. Diagnose faults within the installed system.**

**8. Carry out only the repair that is specifically required.**

**9. Leave the chroot session.**

**10. Unmount the mounted file systems cleanly.**

**11. Restart the target system and check the result.**

## Difference from MX Boot Repair

### MX Boot Repair

This specialises in certain repairs of GRUB and initramfs and provides a graphical interface for them.

### Chroot Rescue Scan

This, by contrast, provides a general administrative working environment within an installed Linux system.

This allows considerably more extensive repairs to be carried out.

This greater flexibility also brings greater responsibility: depending on the target system, far-reaching administrative options are available within the chroot.

## Not a second running operating system

A chroot environment is not a virtual machine and not a fully started second operating system.

The target system continues to use the **kernel of the running JUMBO BLOCK Rescue system**.

The chroot essentially changes the view of the root file system for the processes started within it.

Therefore, not all faults of an installed system can be realistically reproduced or diagnosed within a chroot.

Kernel-dependent problems in particular must be assessed accordingly.

### Limitations with regard to other operating systems

Chroot Rescue Scan is a tool for **Linux systems**.

It is not a corresponding repair procedure for:

Windows,

macOS

or other operating systems that do not provide a suitable Linux chroot environment.

As an overall system, JUMBO BLOCK Rescue remains universally usable; within this system, Chroot Rescue Scan merely provides a specialised Linux repair function.

### Limitations with damaged file systems

Chroot Rescue Scan does not replace file system or storage device recovery.

If the root partition is:

physically damaged,

not reliably readable,

badly corrupted

or:

no longer present at all,

the corresponding rescue tools must be used first.

A chroot should not be used as the first repair attempt on a physically failing storage device.

### Safety notes

Identify the target system unambiguously before entering the chroot.

With several Linux installations, never infer the correct system solely from the order in the selection menu.

With an unstable storage device, first create an image or backup.

Administrative commands run within the chroot can permanently change the target system.

Package management, boot loader commands, initramfs generation and changes under `/etc` affect the selected installation.

Do not use repair commands intended for other distributions without checking.

When finished, leave the chroot and unmount the mounted file systems cleanly.

### Technical classification in JUMBO BLOCK Rescue

Chroot Rescue Scan belongs to:

**Block B – MX tools → Linux system rescue**

It adds to JUMBO BLOCK Rescue direct administrative access to Linux installations that no longer start.

Its task is not automatic repair.

Its actual function is:

**Detect installed Linux system → mount it → prepare chroot environment → enable targeted repair within the target system.**

Within the Linux world, Chroot Rescue Scan is thus considerably more universal than many purely MX-specific administration tools.

The central rescue principle is:

**Identify the target system unambiguously – back up storage devices at risk first – understand the chroot merely as a working environment – carry out only targeted and traceable repairs within the target system.**

Official MX documentation: Chroot Rescue Scan

# B.4 MX Live USB Maker

**Area:** Rescue medium
**Program:** MX Live USB Maker
**Baseline 20261001:** 26.04.1
**Operation:** graphical user interface
**Task:** create bootable live USB media and clone existing live systems

## Purpose

MX Live USB Maker creates bootable USB media.

Possible sources include:

- an ISO file,
- an existing live USB medium,
- a live CD/DVD system,
- the currently running live system.

The tool therefore does not serve directly to repair a target system, but to **create and duplicate bootable rescue media**.

## Significance for JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, MX Live USB Maker can be used, for example, to:

- transfer an ISO to a USB storage device,
- clone an existing live USB system,
- transfer the currently running live system to another USB storage device.

The tool is thus particularly relevant for **providing another bootable rescue medium**.

## Basic workflow

Creating a live USB medium basically requires the following steps:

**1. Select the target USB storage device**

**2. Select the ISO or other source**

**3. Check the settings**

**4. Start the write process**

The most important step here is the unambiguous identification of the **target storage device**.

## Full-featured mode

An extended mode is available for systems from the MX/antiX family.

It supports the special functions of the MX/antiX live infrastructure, including functions that use writable live media and persistence.

These MX-specific functions do not need to be explored further for general operating system rescue with JUMBO BLOCK Rescue.

### Image mode

For ISO files outside the MX/antiX family, MX Live USB Maker uses **Image Mode**.

In this mode, the ISO is written to the target medium as an image using a dd-based procedure.

MX Live USB Maker carries out additional plausibility checks to reduce the risk of selecting the wrong target medium.

The tool can therefore basically also be used to write suitable live or installation ISOs of other operating systems or distributions.

### Cloning a live system

MX Live USB Maker can clone the following to a USB storage device:

**an existing live system**

or:

**the currently running live system**

For JUMBO BLOCK Rescue, this can be used, for example, to create another bootable working medium.

### Dry Run

The **Dry Run** option simulates the intended process without actually writing to the storage device.

This function is particularly helpful for checking the selected configuration first.

However, a dry run does not replace the manual check of the selected target device.

### Encrypted live USB

MX Live USB Maker also supports the creation of a live USB medium fully encrypted with **LUKS**.

This function concerns the protection of the live medium and is not a function for decrypting or rescuing a third-party storage device.

For general LUKS management, JUMBO BLOCK Rescue contains a separate tool, **Cryptsetup**.

### Advanced options

MX Live USB Maker has further options, for example for:

- GPT partitioning,
- updating existing live USB media,
- limiting the size of the area used by the live system,
- file system labelling,
- special boot and development options.

Many of these settings serve special MX/antiX or development requirements.

They are not part of the normal rescue workflow and are therefore not explored further in the JUMBO BLOCK Rescue manual.

### Typical use

Another bootable rescue medium is to be created from an existing ISO.

Procedure:

**1. Connect a suitable USB storage device.**

**2. Start MX Live USB Maker.**

**3. Identify the target device unambiguously on the basis of device, capacity and content.**

**4. Select the ISO.**

**5. Check the selected mode and settings.**

**6. If appropriate, use Dry Run first.**

**7. Start the write process.**

**8. Let the process complete fully.**

**9. Then check the medium created by means of a controlled boot attempt.**

### Limitations

MX Live USB Maker:

- does not repair an installed operating system,
- does not restore deleted files,
- does not repair partition tables,
- does not diagnose storage device defects,
- does not guarantee that any given ISO is bootable on any hardware.

Whether the resulting medium actually boots depends on the ISO, its boot support, and the target hardware.

### Safety notes

Writing an image to a USB storage device overwrites existing data or storage device structures of the selected target medium.

With the underlying dd method too, MX expressly points out that an incorrectly selected target medium can be completely overwritten.

Therefore:

**Always identify the target storage device unambiguously on the basis of device and capacity before starting.**

A storage device with data that is still needed must not be used as the target.

The program's automatic safety checks do not replace this check.

### Technical classification in JUMBO BLOCK Rescue

MX Live USB Maker belongs to:

**Block B – MX tools → Rescue medium**

The tool does not repair a target system. Rather, its task is to **provide or duplicate bootable live and rescue media**.

The following distinction is particularly important:

**Full-featured mode**

→ extended MX/antiX live functions.

**Image mode**

→ write ISO image to USB; also intended for non-MX/non-antiX ISOs.

The central safety principle is:

**Distinguish source and target unambiguously – check the target medium before writing – regard existing data on the target as at risk – then check the rescue medium created by means of a boot attempt.**

# B.5 MX Snapshot

**Area:** System image / ISO creation
**Program:** MX Snapshot
**Baseline 20261001:** 26.08
**Operation:** graphical user interface
**Task:** back up a prepared MX-based system state, including deliberately included files, as a bootable ISO image

## Purpose

MX Snapshot creates a bootable ISO image from a running installed MX-based system.

Within JUMBO BLOCK Rescue, this function can do more than simply reproduce the rescue system:

**Selected files of an examined or damaged system can first be backed up with the rescue tools, transferred into the snapshot content in a controlled manner and then backed up together with the rescue system in a bootable ISO image.**

MX Snapshot can thus also be part of a rescue workflow for the **orderly transfer of selected rescued data**.

## Area of use in the rescue system

One possible use case is a compromised or hacked system.

JUMBO BLOCK Rescue can be used independently of the affected operating system to identify required files and transfer them to a controlled working area.

Selected backed-up files can then deliberately become part of an ISO image created with MX Snapshot.

In this way, the following, for example, can be backed up together in a defined image:

- the bootable rescue system,
- selected rescued documents,
- logs,
- configuration files,
- examination results,
- technical documentation,
- other deliberately transferred files.

The files to be included must be selected for the specific rescue case.

### Special considerations for compromised systems

Files from a hacked or otherwise compromised system must basically be treated as **potentially untrustworthy**.

Including them in a snapshot ISO does not make these files secure and does not change their trust status.

In particular, executable files, scripts, macros, manipulated documents or other active content can still contain malicious code.

Therefore:

**Backing up is not the same as trusting.**

Within the rescue workflow, the transferred files should be clearly separated from the trustworthy system content and treated as material from the examined target system.

### System snapshot

For a system state that is to be passed on, MX Snapshot offers a system snapshot mode.

This adopts the installed system state and the intended applications without simply transferring all the personal content of a user account into the resulting system.

Any additional rescue files to be included in the snapshot must therefore be selected deliberately.

### Personal snapshot

Personal snapshot mode can take considerably more user content into account.

In a controlled rescue workflow, however, the goal is not to transfer as much data as possible.

What matters is:

**Only the files actually intended should become part of the image created.**

### Exclusions

MX Snapshot uses exclusion rules with which files and directories can be kept out of the ISO created.

These exclusions are particularly important for a rescue snapshot.

Before the ISO is created, the following must be checked:

- which rescued files are to be included,
- which working files are not needed,
- whether personal or confidential data is included,
- whether temporary files are to be excluded,

- whether potentially dangerous files are being deliberately carried along as examination material.

### Typical rescue workflow

A compromised system is not to be used productively any longer, but certain files must be backed up.

**1. Start JUMBO BLOCK Rescue independently of the affected system.**

**2. Check the condition of the storage devices.**

**3. Identify the required files and back them up to a controlled working area.**

**4. If required, examine files with the available analysis and security tools.**

**5. Maintain or document the origin and trust status of the transferred files.**

**6. Provide only the files actually needed for the snapshot.**

**7. Check the exclusion rules.**

**8. Start MX Snapshot and create the ISO.**

**9. Check the integrity and content of the ISO created.**

**10. Check that the ISO is bootable.**

This can produce a portable system state containing JUMBO BLOCK Rescue together with selected backup or examination material.

### No automatic decontamination

MX Snapshot does not carry out any security clean-up of the files included.

A file from a compromised system remains potentially compromised even within a snapshot ISO.

Such files should therefore not be opened or executed merely because they have been transferred successfully into a new ISO.

For an examination, the corresponding security and analysis tools from Block A, among others, are available within JUMBO BLOCK Rescue.

### Further use

MX Snapshot can of course also be used for:

- backing up a defined JUMBO BLOCK Rescue system state,
- reproducing a prepared system,
- creating a customised bootable ISO,
- passing on a defined system state.

These options complement the rescue use case of transferring selected files in a controlled manner.

### Limitations

MX Snapshot itself:

- does not rescue deleted files,
- does not repair damaged file systems,
- does not examine malware,
- does not decide on the trustworthiness of transferred files,
- does not clean compromised files.

The actual rescue and analysis are carried out with the JUMBO BLOCK Rescue tools intended for this purpose.

MX Snapshot can then serve to **combine the prepared system and data content in a bootable ISO image**.

### Safety notes

For data from a compromised system:

**Preserve the origin – do not change the trust status – do not execute files without checking.**

Only transfer files that are needed.

Treat executable content particularly critically.

Before creating the snapshot, check which data actually becomes part of the ISO.

Exclude personal, confidential and unneeded data.

Check the finished ISO for content, integrity and bootability.

### Technical classification in JUMBO BLOCK Rescue

MX Snapshot belongs to:

**Block B – MX tools → System image / ISO creation**

Its role in rescue operations can extend beyond simply reproducing the base system.

After appropriate preparation, it makes it possible to **back up and transfer a defined bootable rescue system state together with deliberately selected rescued files or files to be examined in an ISO image**.

This creates a clear division of responsibilities:

**JUMBO BLOCK Rescue tools**

→ identify, diagnose, rescue, copy and examine data.

**MX Snapshot**

→ can combine the subsequently prepared system and data content in a bootable ISO image.

The central principle is:

**Transfer rescued data in a controlled manner – preserve origin and trust status – include only files that are needed – do not confuse a snapshot with decontamination – then check the finished ISO.**

# B.6 ISO Signature Verifier

**Area:** Integrity and authenticity checking
**Program:** ISO Signature Verifier
**Baseline 20261001:** 2026.09.01
**Package:** verify-iso-sig
**Operation:** graphical user interface / command line
**Task:** verify signatures and checksums of ISO images

## Purpose

ISO Signature Verifier is used to check ISO images on the basis of the associated cryptographic signatures and checksum information.

Before an ISO is used, this makes it possible to check whether the file at hand matches the reference information signed or published by the publisher.

The tool combines two different checking principles:

- cryptographic verification of a signature,
- integrity checking of the ISO image on the basis of a checksum.

## Significance for JUMBO BLOCK Rescue

Within a rescue system, ISO files can be of particular security significance.

They can contain, for example:

- operating systems,
- installation media,
- live systems,
- diagnostic environments,
- rescue systems,
- firmware or maintenance environments.

Before such an ISO is booted, installed, written to a storage device, or otherwise used, its integrity should be verified and, where suitable signature information is available, its origin should be cryptographically authenticated.

Within JUMBO BLOCK Rescue, ISO Signature Verifier provides a specialised checking tool for this purpose.

## Operating-system-independent checking

ISO Signature Verifier does not check which operating system is contained in an ISO image.

The subject of the check is the **signature and checksum information** belonging to the ISO.

What matters is not the operating-system family, but whether the ISO has suitable signature or checksum information supported by the tool and whether the signing key can be reliably attributed to the expected publisher.

The underlying checking principle is thus **independent of the operating system**.

For JUMBO BLOCK Rescue:

**It is not the operating system in the ISO that determines whether it can be verified, but the signature and checksum procedure used and the availability of a trustworthy reference.**

### Checksum and signature

Checksum and cryptographic signature perform different tasks.

A **checksum** makes it possible to compare a file with a known reference value.

If, for example, a published SHA checksum matches the checksum calculated from the ISO at hand, the file checked corresponds to the data for which this reference value was generated.

However, a checksum alone does not automatically answer the question of **who published this reference value**.

This is where the cryptographic signature comes in.

A verifiable signature can cryptographically link the checksum or the ISO to a specific signature key.

This combines:

**integrity**

and:

**authenticity checking**.

### Different signature procedures

Publishers use different publication and signature procedures for ISO images.

Depending on the procedure, for example:

- the ISO can be signed directly,
- a checksum file can be signed,
- a single checksum belonging to the ISO can be signed,
- checksum and signature information can be published together.

ISO Signature Verifier evaluates the supported structures and carries out the associated checks.

### Basic workflow

Operation basically follows a simple scheme:

**1. Start ISO Signature Verifier.**

**2. Select the ISO to be checked or the associated signature information.**

**3. Start the verification.**

**4. Check the result.**

If the required accompanying files are present and the procedure used is supported, the ISO, checksum and signature can be compared with one another.

### Signature keys

A cryptographically correct signature is only meaningful if it has also been clarified **to whom the signature key used actually belongs**.

With an unknown key, its identity must therefore be checked.

The following are particularly important:

- key ID,
- fingerprint,
- stated identity.

The fingerprint of an unknown key should be checked via an independent and trustworthy source of the publisher concerned.

### A valid signature does not automatically mean a trustworthy publisher

An important distinction is:

**Cryptographically valid signature**

This initially means only that the signature is cryptographically valid for the verified data and the key used.

Whether that key actually belongs to the expected publisher is a separate question.

Unknown signature keys must therefore not be regarded as trustworthy merely because of a technically successful signature check.

### Typical rescue workflow

An external ISO is to be used within JUMBO BLOCK Rescue or transferred to a bootable medium.

**1. Obtain the ISO from a traceable source.**

**2. Also obtain the available signature and checksum information.**

**3. Start ISO Signature Verifier.**

**4. Check the ISO and the associated verification information.**

**5. Check the result.**

**6. With an unknown signature key, check its fingerprint independently.**

**7. Only use the ISO further after a successful check.**

**8. If required, then transfer the checked ISO to a suitable USB medium with MX Live USB Maker.**

This results in a clear workflow:

**Obtain the ISO → verify origin and integrity → only then use it.**

### Failed verification

If a verification fails, this can have various causes.

Possible causes are, for example:

- damaged file,
- incomplete file,
- modified ISO,
- wrong checksum file,
- wrong signature file,
- files that do not belong together,
- non-matching signature key.

The failure of the verification alone therefore does not automatically reveal **why** the check failed.

The ISO should not be used further until the cause has been investigated.

### Successful verification

Within the scope of the verification method used, a successful result confirms that the ISO matches the checked signature or checksum information.

To establish origin reliably, the signing key must also be reliably attributed to the expected publisher.

### No malware check

ISO Signature Verifier is not a virus or malware scanner.

A successful verification does not automatically mean that all contents of an ISO are safe or free of malware.

In particular, it does not mean that:

- no security vulnerabilities are contained,
- all software contained is trustworthy,
- the ISO works without errors on the hardware present,
- the ISO is actually bootable.

The tool answers a different question:

**Does the ISO at hand match the checked signature and checksum information?**

### Interaction with ClamAV

ISO Signature Verifier and ClamAV perform different security tasks.

**ISO Signature Verifier**

→ checks integrity and cryptographic authenticity on the basis of existing reference information.

**ClamAV**

→ examines accessible files for detectable malware.

These two checks serve different purposes and should not be confused.

### Interaction with MX Live USB Maker

ISO Signature Verifier and MX Live USB Maker also have clearly separate tasks.

**ISO Signature Verifier**

→ checks an ISO.

**MX Live USB Maker**

→ transfers an ISO to a USB medium.

A sensible workflow within JUMBO BLOCK Rescue is therefore:

**Verify the ISO → then transfer it to the target medium → test the medium created.**

### Own checksums

A new checksum can be calculated for any existing file at any time.

Such a checksum can later show whether that exact file has changed.

It does not, however, retrospectively prove the file’s origin.

A trustworthy reference is required for this.

### Limitations

ISO Signature Verifier can only carry out an authenticity check if suitable signature information is available.

If no signature information is present, an existing published checksum may still serve for integrity checking.

Without a trustworthy reference, however, a self-calculated checksum alone cannot determine which publisher an ISO comes from.

Furthermore, the tool replaces neither a security analysis of the ISO content nor a practical boot and function test.

### Safety notes

Obtain signature and checksum information from a trustworthy source of the publisher wherever possible.

Do not accept unknown signature keys without checking.

Check fingerprints of unknown keys independently.

Do not use an ISO whose verification has failed without further clarification.

Do not confuse a successful signature check with a complete security analysis.

Before productive use of a bootable ISO, additionally check that it actually boots and works.

### Technical classification in JUMBO BLOCK Rescue

ISO Signature Verifier belongs to:

**Block B – MX tools → Integrity and authenticity checking**

Its function is **not geared to a particular operating system family**.

What matters is the signature and checksum method used for the ISO.

ISO Signature Verifier therefore provides an important verification function within JUMBO BLOCK Rescue:

**An external ISO can be checked for conformity with trustworthy cryptographic reference information before it is used.**

Together with MX Live USB Maker, this provides a secure basic workflow:

**Obtain the ISO → verify signature and checksum → transfer the ISO to the rescue medium → test the medium in practice.**

The central principle is:

**Do not trust the file name or the supposed origin of an ISO – check the available cryptographic references – verify unknown keys – only then use the image.**

# Block C

In addition to the specialised rescue tools from **Block A – JB-Rescue** and the selected MX tools from **Block B**, JUMBO BLOCK Rescue provides various general working tools.

These programs do not perform a central rescue function, but they support practical work during a deployment. They include in particular formatting removable media, graphical work with files and archives, searching for files, editing text files, checking checksums, network configuration, and terminal and documentation tasks.

For this manual, only those accessory programs are described individually that are of specific practical use within a rescue environment. Other programs and desktop components of the base system are not the subject of the detailed documentation.

## Tool groups

**Nine relevant accessory tools:**

**C.1 – Format USB**
Formatting and preparation of USB sticks and other suitable removable media for further use.

**C.2 – Thunar File Manager**
Graphical management of files, directories, storage devices and mounted file systems.

**C.3 – Catfish File Search**
Targeted search for files within accessible directories and mounted storage devices.

**C.4 – Engrampa Archive Manager**
Graphical work with supported archive formats for opening, creating and extracting archives.

**C.5 – FeatherPad**
Lightweight text editor for displaying and editing text, configuration and log files.

**C.6 – GtkHash**
Calculation and comparison of cryptographic checksums for checking the integrity of files.

**C.7 – Advanced Network Configuration**
Graphical configuration and editing of network connections for communication within a rescue environment.

**C.8 – Xfce Terminal**
Terminal environment for command-line tools, diagnostic commands and administrative work.

**C.9 – Screenshot**
Creation of screen captures for documenting system states, error messages, analysis results and rescue work carried out.

The nine accessory tools thus complement the specialised functions from **Block A – JB-Rescue** and **Block B – MX tools** without forming the core of the rescue system themselves.

**Block C provides the general working tools that support examination, backup, editing and documentation in practice during a rescue deployment.**

## Assignment of areas, programs and versions

| No. | Area | Program | Baseline version |
|---|---|---|---|
| **C.1** | Storage devices | Format USB | 26.09.01 or later |
| **C.2** | Files | Thunar File Manager | 4.20.2 or later |
| **C.3** | Files | Catfish File Search | 4.20.1 or later |
| **C.4** | Archives | Engrampa Archive Manager | 1.26.2 or later |
| **C.5** | Text | FeatherPad | 1.6.2 or later |
| **C.6** | Integrity | GtkHash | 1.5 or later |
| **C.7** | Network | Advanced Network Configuration | 1.36.0 or later |
| **C.8** | System | Xfce Terminal | 1.1.4 or later |
| **C.9** | Documentation | Screenshot | 1.11.1 or later |

# C.1 – Format USB

**Area:** Storage devices
**Program:** Format USB
**Baseline 20261001:** 26.09.01 or later

## Purpose

**Format USB** is a graphical tool for formatting removable USB media.

The program allows a connected USB storage device, a file system and, optionally, a volume label to be selected. The selected device is repartitioned and given a new file system.

In JUMBO BLOCK Rescue, Format USB is primarily used to prepare USB media for receiving, transferring or handing over files.

## Use in the rescue system

During a rescue deployment, additional removable media are often required, for example for:

- rescued files,
- logs and examination results,
- exported configurations,
- screenshots and documentation,
- working copies,
- files for handover to another system.

Format USB provides a simple graphical way to prepare a suitable USB storage device for this purpose.

It must be distinguished from **MX Live USB Maker from Block B**:

**Format USB** formats a USB storage device for general use.

**MX Live USB Maker**, by contrast, creates a bootable medium from an ISO image or a supported live system.

## Functions

According to the official MX documentation, Format USB supports the following file systems:

- **FAT32/VFAT**
- **NTFS**
- **ext4**
- **exFAT**

In addition, a label can be assigned to the newly created file system.

The selected USB device is repartitioned as a whole. A partition is then created and formatted with the selected file system.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Connect the desired USB medium.
2. Start **Format USB**.
3. Identify the correct target device unambiguously.
4. Select the desired file system.
5. If required, assign a volume label.
6. Check the selection once more.
7. Carry out the formatting.
8. When finished, check whether the medium is mounted correctly and can be used.
9. Only then copy the files to be backed up or transferred to the medium.

The unambiguous identification of the target device is particularly important. If several USB storage devices are connected, the assignment should be checked carefully on the basis of the device information displayed.

## Choice of file system

The choice of file system depends on the intended target system and use case.

**FAT32/VFAT** offers very broad compatibility but has technical limitations, including with regard to the size of individual files.

**exFAT** is particularly suitable for exchanging larger files between different modern operating systems.

**NTFS** can be useful if the medium is to be used mainly with systems that support this file system.

**ext4** is a Linux file system and is particularly suitable for further use within corresponding Linux environments.

For rescue work, the file system should therefore be selected with both storage requirements and the intended target system in mind.

## Typical rescue scenarios

### Providing a storage device for rescued files

After a data recovery with TestDisk, PhotoRec, Foremost or other tools, a separate USB medium can be prepared to which the rescued data is then transferred.

### Handover of examination results

Logs, checksums, screenshots, configuration files or other examination results can be collected on a newly prepared USB storage device and then evaluated on another system.

### Data exchange between different operating systems

By selecting a suitable file system, a removable medium can be prepared for data exchange between JUMBO BLOCK Rescue and another operating system.

## Limitations and safety notes

Format USB is **not a data recovery tool**.

The program does not restore lost files, does not repair damaged file systems and does not create sector-by-sector backup images.

Particularly important:

**Formatting fundamentally changes the selected target medium. Existing partitions and the data stored on them are put at risk or removed as a result.**

A storage device from which data is still to be rescued must therefore not be prepared with Format USB.

Likewise, a damaged or unstable source storage device must not be selected as the target by mistake.

If there are doubts about the device assignment, the identity of the storage device should first be checked with the diagnostic tools from Block A.

For a potentially compromised system, also bear in mind:

**Formatting a USB medium does not automatically make files subsequently copied to it trustworthy.**

Rescued files from a compromised system can still contain malicious code, manipulated content or other security-relevant changes.

## Technical classification within JUMBO BLOCK Rescue

Format USB is not part of JUMBO BLOCK Rescue's core data-recovery functionality. It is a practical tool for preparing additional removable media.

The roles are clearly separated:

**Block A – JB-Rescue**
identifies, examines, backs up and rescues data and storage devices.

**Block B – MX Live USB Maker**
creates bootable USB media.

**C.1 – Format USB**
prepares general removable USB media for further use by repartitioning and formatting.

Format USB thus particularly supports the controlled receipt and transfer of data after a rescue or analysis measure.

**Key rule:**

**Identify the target medium unambiguously – regard existing data as at risk – select a suitable file system – only transfer the data to be backed up after successful formatting.**

# C.2 – Thunar File Manager

**Area:** Files
**Program:** Thunar File Manager
**Baseline 20261001:** 4.20.2 or later

## Purpose

**Thunar** is the graphical file manager of the Xfce desktop environment. It is used to display, navigate, copy, move, rename and manage files and directories.

In JUMBO BLOCK Rescue, Thunar provides the graphical interface for working with accessible file systems and removable media.

## Use in the rescue system

After a storage device has been identified and, where applicable, backed up, files often have to be reviewed, organised or transferred to another medium.

Thunar is particularly suitable for:

- navigating accessible directory structures,
- reviewing rescued files,
- copying files to backup media,
- transferring between mounted storage devices,
- creating working and handover directories,
- renaming and organising rescued files,
- opening files with suitable applications,
- accessing mounted removable media and file systems.

Thunar complements the specialised data-recovery tools in Block A by providing graphical file management.

## Functions

The basic functions include:

- displaying files and directories,
- navigating within directory structures,
- copying and moving,
- renaming,
- deleting,
- creating new directories,

- displaying file properties,
- opening files with associated applications,
- working with mounted storage devices and removable media.

The specific availability of individual functions depends on the access rights, the mounted file system and the extensions present in the rescue system.

## Operation and workflow

A typical rescue workflow is:

1. Identify the source storage device and assess its condition.
2. With unstable or damaged storage devices, first create an image or backup.
3. Mount a suitable storage device or the working copy.
4. Open Thunar.
5. Go to the desired source directory.
6. Select the files to be backed up.
7. Open a separate target medium or target directory.
8. Copy the files.
9. Let the copy process complete fully.
10. Then check the transferred data at the target.

For rescue work, files should basically be **copied rather than moved** as long as the source is to be preserved as an original or as evidence.

## Typical rescue scenarios

### Taking over rescued files

Files recovered with PhotoRec, Foremost or other data recovery tools can be reviewed and sorted with Thunar and then copied to a separate backup medium.

### Backing up accessible user data

If an operating system that no longer starts is still readable via JUMBO BLOCK Rescue, accessible documents and other required files can be copied graphically to a separate medium.

### Compiling examination results

Logs, screenshots, configuration files, checksums and other results of an examination can be collected in a clear directory structure and then handed over.

### Data exchange with removable media

A removable medium prepared, for example, with **C.1 – Format USB** can be opened in Thunar and used to receive the data to be transferred.

## Limitations and safety notes

Thunar is **not a data recovery program**.

The file manager does not reconstruct lost partitions, does not restore deleted files and does not create sector-by-sector images of damaged storage devices.

If Thunar cannot display a file or directory, this therefore does not automatically mean that the data concerned is no longer present or cannot be rescued.

With an unstable storage device, no attempt should be made to read large amounts of data repeatedly and directly with the file manager. In this case, a suitable backup image, for example with GNU ddrescue, is preferable first.

Particular care applies to writing operations:

**Moving, renaming, deleting and creating or changing files can change the state of the mounted file system.**

If a source is to be preserved unchanged, it should therefore be mounted read-only wherever possible, and writing should be done exclusively to a separate working or backup copy.

Files from a compromised system also remain potentially untrustworthy. Copying with Thunar does not change their security status.

## Technical classification within JUMBO BLOCK Rescue

Thunar provides graphical file management within the rescue environment.

It is important to distinguish Thunar from the tools in Block A:

**TestDisk, PhotoRec, GNU ddrescue, Foremost and Extundelete**
serve to back up or restore data and storage device contents.

**Midnight Commander**
provides terminal-based file management.

**C.2 – Thunar File Manager**
enables the graphical review, organisation and transfer of files and directories that are already accessible.

Thunar is thus a practical working tool for the further handling of the data obtained, particularly after a successful backup, mounting or data recovery.

**Key rule:**

**Preserve the original as unchanged as possible – with storage devices at risk, back up first – preferably copy rather than move rescued data – check target and source unambiguously before every file operation.**

# C.3 – Catfish File Search

**Area:** Files
**Program:** Catfish File Search
**Baseline 20261001:** 4.20.1 or later

## Purpose

**Catfish File Search** is a graphical tool for searching for files and directories.

In JUMBO BLOCK Rescue, Catfish supports targeted searches across accessible directory structures, mounted storage devices and previously recovered data.

The program thus complements the Thunar file manager with a convenient search function.

## Use in the rescue system

In rescue work, large or confusing amounts of data are often present. Documents, configuration files, logs or other relevant files that are being sought must then be located within extensive directory structures.

Catfish can be used, for example, to:

- find specific files by their name,
- narrow down file types or file name patterns,
- search specific directory areas in a targeted manner,
- search rescued data for relevant files,
- locate files of a system that no longer starts but is still readable,
- compile examination results for further processing.

## Functions

Catfish provides a graphical search interface and in particular enables searches for file and directory names.

Depending on the search criteria and the available system environment, search results can additionally be narrowed down on the basis of various criteria.

The key point for rescue work is:

**Catfish searches files and directories that can be reached via the currently accessible file system.**

It therefore operates at a different level from data-reconstruction tools.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Mount the affected storage device or a working copy.
2. Check whether the required directory structure is accessible.
3. Start Catfish.
4. Specify the directory area to be searched.
5. Enter the search term or desired search pattern.
6. Carry out the search.
7. Check the list of results.
8. If required, open, copy or transfer relevant files to a separate working directory with Thunar.

With large amounts of data, it is advisable to narrow down the search area as sensibly as possible.

## Typical rescue scenarios

### Searching for user documents

If an operating system no longer starts but its file system is still readable, Catfish can help to search for required documents within the existing user directories.

### Searching rescued data

After a data recovery, numerous files and directories can be present. Catfish helps to search specifically for particular file names or groups of files within the accessible result.

### Locating configuration and log files

When analysing a system, certain configuration, log or backup files may be required. If their exact storage locations are not known, a targeted search can make the further examination easier.

### Preparing a data handover

Before rescued data is handed over, Catfish can be used to locate certain files being sought within the working data. The actual transfer to a separate medium is then carried out, for example, with Thunar.

## Limitations and safety notes

Catfish is **not a data recovery tool**.

A file that does not appear in the search results is therefore not automatically lost for good.

In particular, Catfish cannot:

- reconstruct deleted files,
- restore lost partitions,

- repair damaged file system structures,
- reconstruct data blocks that are no longer accessible,
- create sector-by-sector storage device images.

The specialised tools from **Block A – JB-Rescue** are intended for such tasks.

The access rights of the running system and the actual mounting and readability of the examined file system also influence which data can be searched.

With a physically unstable or badly damaged storage device, repeated extensive searches should also not be carried out on the original. In this case, the basic rescue principle continues to apply:

**First back up or create an image, and then search on the working copy.**

## Technical classification within JUMBO BLOCK Rescue

Within Block C, Catfish has a clearly defined supporting role:

**Block A – JB-Rescue**
rescues or reconstructs data and makes damaged or lost data accessible again as far as possible.

**C.2 – Thunar File Manager**
enables graphical navigation, organisation and transfer of accessible files.

**C.3 – Catfish File Search**
supports the targeted locating of files within such accessible data.

Catfish is therefore particularly useful when data is already accessible or has been rescued successfully, but its specific storage location within a larger set of data still has to be determined.

**Key rule:**

**Choose the search area deliberately – leave originals as unchanged as possible – with unstable storage devices, back up first – a missing search result is no proof that a file can no longer be rescued.**

# C.4 – Engrampa Archive Manager

**Area:** Archives
**Program:** Engrampa Archive Manager
**Baseline 20261001:** 1.26.2 or later

## Purpose

**Engrampa Archive Manager** is a graphical tool for creating, opening and editing archive files.

In JUMBO BLOCK Rescue, Engrampa is primarily used to open existing or recovered archives, inspect their contents, extract selected files and package data for an orderly handover.

## Use in the rescue system

In rescue and analysis work, archive files can be both part of the examined data and an aid for the further handover of data.

Engrampa can be used, for example, to:

- open existing archive files,
- view the content of an archive,
- extract one or more files,
- combine files into a new archive,
- prepare rescued data in an orderly manner for further processing,
- store logs, configuration files or examination results together.

Engrampa complements Thunar’s file management with tools for compressed and packaged data.

## Functions

Engrampa provides a graphical interface for working with archive files.

The essential functions include:

- opening existing archives,
- displaying the archive content,
- extracting one or more files,
- extracting complete archives,
- creating new archives,
- adding files to supported archives,
- removing entries from supported archives,
- working with various archive and compression formats.

Which formats and functions are actually available depends not only on Engrampa itself but also on the archive and compression tools present in the system.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Locate the archive file on the examined storage device or within rescued data.

2. With original storage devices at risk, first create a backup or working copy.
3. Open the archive from the working copy wherever possible.
4. Check the content of the archive.
5. Select the required files.
6. Specify a separate target directory for the extraction.
7. Extract the files.
8. Then check the extracted files and, if necessary, examine them further with suitable tools.

If a new archive is to be created for handing over data, the files to be included should first be compiled and checked specifically.

## Typical rescue scenarios

### Opening a rescued archive

If, for example, an archive file has been restored with a data recovery tool, Engrampa can be used to check whether the archive can be opened and its content extracted.

A successfully restored archive file is not automatically complete or free of errors. Damaged or incomplete archive structures can result in only parts of the content being accessible.

### Extracting required files

If relevant documents, configurations or other files are located within an archive, they can be extracted to a separate working directory and then examined independently of the archive.

### Compiling examination results

Logs, screenshots, checksums, configuration files and other results of a rescue deployment can be combined in an archive if required.

This allows related files to be kept or transferred as a common set of data.

### Preparing a data handover

Rescued files can first be organised with Thunar and then combined into an archive with Engrampa, provided that this is expedient for the further handover.

## Limitations and safety notes

Engrampa is **not a data recovery tool**.

The program does not restore deleted files, does not reconstruct lost partitions and does not create sector-by-sector backup images.

Also note:

**An archive is not a security container.**

Packing files in an archive does not change their trust status. Files from a compromised or unknown system remain potentially untrustworthy even after archiving.

When opening third-party or rescued archives, it should therefore be borne in mind that their content can include, for example, executable files, scripts, manipulated documents or other potentially dangerous content.

Merely extracting such files does not make them safe.

With unknown data, extracted content should therefore not be executed thoughtlessly or opened with unnecessarily privileged applications.

Even a successful extraction does not prove that the files contained are unchanged in content, complete or trustworthy.

## Technical classification within JUMBO BLOCK Rescue

Within Block C, Engrampa provides graphical archive management.

The roles are clearly separated:

**Block A – JB-Rescue**
serves to back up, analyse and restore data.

**C.2 – Thunar File Manager**
manages accessible files and directories.

**C.3 – Catfish File Search**
searches specifically within accessible data.

**C.4 – Engrampa Archive Manager**
opens, extracts, creates and manages supported archive files.

Engrampa thus becomes relevant in particular after a successful backup or data recovery, when archived content is to be examined or extracted, or rescued data is to be combined for further processing.

**Key rule:**

**Preserve the original or working copy – open and extract archives in a controlled manner – do not execute unknown content thoughtlessly – never equate archiving with a security check or trustworthiness.**

# C.5 – FeatherPad

**Area:** Text
**Program:** FeatherPad
**Baseline 20261001:** 1.6.2 or later

## Purpose

**FeatherPad** is a lightweight graphical plain-text editor.

In JUMBO BLOCK Rescue, FeatherPad is primarily used to view and edit text-based files such as configuration files, logs, scripts, lists and simple notes.

It allows text files to be inspected directly and, where necessary, modified in a controlled manner within the rescue environment.

## Use in the rescue system

In diagnostic and repair work, text-based files often have to be examined.

FeatherPad can be used, for example, to:

- open and check configuration files,
- read log files,
- search for specific strings within a file,
- change individual configuration entries specifically,
- view scripts and other text-based files,
- record examination results or technical notes,
- check copied or rescued text files.

Thanks to its support for several tabs, several files can be opened at the same time and compared with one another or edited one after another.

## Functions

The FeatherPad functions relevant to rescue work include, among others:

- opening and saving text files,
- working with several files via tabs,
- search and replace,
- highlighting of found search terms,
- display of line numbers,
- jumping to a specific line,

- syntax highlighting for common programming languages,
- column selection,
- session management,
- auto-save,
- enlarging and reducing text,
- printing.

FeatherPad is deliberately a text editor and not a complete office or document processing system.

## Operation and workflow

A typical rescue workflow is:

1. Identify the affected file unambiguously.
2. Establish whether the file is merely to be examined or actually changed.
3. With important original files, first create a copy.
4. Open the file with FeatherPad.
5. Check the content or locate the required point using the search function.
6. Make the necessary changes as specifically as possible.
7. Save the changed file.
8. Then check the content and effects of the change.

If a configuration file of a system that no longer starts is to be edited, particular care must be taken to ensure that the file of the desired target system has actually been opened and not a file of the same name belonging to the running rescue system.

## Typical rescue scenarios

### Checking configuration files

Text-based system and application configurations can be opened and examined for incorrect or conspicuous entries.

### Targeted configuration change

Once the cause of a problem has been identified unambiguously, a corresponding text file can be edited specifically on a working copy or in the mounted target system.

The original file should be preserved before a change.

### Evaluating log files

Existing log files can be opened and searched for relevant terms, messages or entries using the search function.

### Examining rescued text files

Text, configuration or script files restored with the tools from Block A can be opened with FeatherPad and checked for usable content.

### Documenting a rescue deployment

FeatherPad can also be used to record simple technical notes, command sequences, file lists or examination results as text files.

## Limitations and safety notes

FeatherPad is **not a data recovery, analysis or security tool**.

The editor does not restore deleted files, does not repair file systems and does not assess whether the content of an opened file is correct, trustworthy or harmless.

Particular care is required when editing system and configuration files.

Even small changes can result in an operating system, a service or an application no longer working properly afterwards.

Therefore:

**Back up important original files before a change and only make targeted, traceable changes.**

With files from a compromised system, their origin must also be taken into account. Opening or saving a file with FeatherPad confirms neither its integrity nor its trustworthiness.

Automatic save functions require particular attention when working on original files, as unintentional changes can thereby be written permanently to the opened file.

## Technical classification within JUMBO BLOCK Rescue

Within Block C, FeatherPad provides editing for plain-text content.

The roles are clearly separated:

**C.2 – Thunar File Manager**
manages files and directories.

**C.3 – Catfish File Search**
finds files within accessible data.

**C.4 – Engrampa Archive Manager**
manages and extracts archive files.

**C.5 – FeatherPad**
displays text-based files and enables their targeted editing.

FeatherPad is thus a practical tool within the rescue environment, particularly for examining configurations and logs and for controlled manual changes to text-based files.

**Key rule:**

**Identify the file and target system unambiguously – back up important originals before changes – only make targeted and traceable changes – then check the saved result.**

# C.6 – GtkHash

**Area:** Integrity
**Program:** GtkHash
**Baseline 20261001:** 1.5 or later

## Purpose

**GtkHash** is a graphical tool for calculating and checking cryptographic hash values or checksums of files.

In JUMBO BLOCK Rescue, GtkHash is primarily used to compare files before and after transfer, verify known reference checksums and document the integrity of backed-up or recovered files.

## Use in the rescue system

In rescue work, files are often transferred from one storage device to another medium, to a working directory or to a backed-up set of data.

GtkHash can be used here, for example, to:

- calculate hash values of individual files,
- compare files with one another on the basis of their hash values,
- check a file against a known reference value,
- check existing checksum files,
- check that important files have been transferred unchanged,
- record hash values of rescued files for technical documentation.

GtkHash complements JUMBO BLOCK Rescue's file and transfer tools by providing file-integrity verification.

## Functions

GtkHash supports numerous hash procedures. These include, among others:

- **MD5**
- **SHA-1**
- **SHA-2**, including SHA-256 and SHA-512
- **SHA-3**
- **BLAKE2**

GtkHash also supports the checking of checksum files, parallel or multi-threaded hash calculation and HMAC procedures.

Which hash functions are available in the specific system can depend on the libraries and options used when the program was built.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Identify the file to be checked unambiguously.
2. Start GtkHash.
3. Select the file.
4. Select the desired hash procedure.
5. Have the hash value calculated.
6. Document the result or compare it with an existing reference value.
7. After a copy or transfer process, calculate the hash value of the target file again.
8. Compare the source and target values exactly.

If the hash values calculated with the same procedure are identical, there is a very strong technical basis for assuming that the compared file contents are identical.

Different hash values, by contrast, show that the compared data is not identical.

## Typical rescue scenarios

### Checking a file transfer

An important rescued file is copied from a working storage device to a separate backup medium.

Hash values can be calculated before and after the transfer and compared with one another. This makes it possible to check whether the file content changed during the copy process.

### Documenting rescued files

Hash values can be documented for particularly relevant files. Later checks then make it possible to establish whether the file content concerned has remained unchanged compared with this documented state.

### Checking against a published checksum

If a reference value obtained from a trustworthy source is available for a file, the value calculated with GtkHash can be compared with it.

This is useful, for example, for downloaded images, software packages or other files for which the publisher has published a checksum.

### Checking rescued data

After a successful data recovery, hash values can be generated for selected files, or the files can be checked against existing reference values.

A distinction must be made here between successful data recovery and proven integrity: without a trustworthy earlier reference value, only the current state of the rescued file can be documented.

## Limitations and safety notes

A hash value initially answers the question:

**Is the compared data identical, or has its content changed compared with a known state?**

It does not automatically answer the question:

**Is this file trustworthy or harmless?**

A compromised, manipulated or malware-infected file also has a clearly calculable hash value.

If the hash value of an already manipulated file is documented, a later match merely confirms that this file has not changed since the calculation.

A published checksum, too, is only as trustworthy as the source from which the reference value comes.

For proof of origin or authenticity, additional cryptographic procedures, for example digital signatures, may therefore be required.

For new integrity proofs, suitable modern hash procedures such as SHA-256 or SHA-512 should be preferred over older procedures such as MD5 or SHA-1.

## Technical classification within JUMBO BLOCK Rescue

Within Block C, GtkHash provides general file-level integrity checking.

There is an important distinction here from **B.6 – ISO Signature Verifier**:

**B.6 – ISO Signature Verifier**
is geared to the integrity and authenticity checking of ISO images on the basis of suitable signature and checksum information.

**C.6 – GtkHash**
is a general tool for calculating and comparing hash values of any files.

GtkHash can thus be used at many points in a rescue workflow, particularly before and after copy, backup or handover processes.

However, GtkHash replaces neither a malware analysis nor cryptographic proof of origin.

**Key rule:**

**Identify the file unambiguously – use a suitable hash procedure – compare the reference value and the calculated value exactly – understand a checksum match as proof of integrity, not automatically as proof of trustworthiness or security.**

# C.7 – Advanced Network Configuration

**Area:** Network
**Program:** Advanced Network Configuration
**Baseline 20261001:** 1.36.0 or later

## Purpose

**Advanced Network Configuration** (`nm-connection-editor`) is the graphical interface for managing NetworkManager network connections.

In JUMBO BLOCK Rescue, this tool is used to configure the rescue environment's network connection or modify existing connection profiles.

This provides the connectivity required for network diagnostics, access to network resources and other network-based rescue tasks.

## Use in the rescue system

A working network connection may be required during a rescue deployment, for example, for:

- access to other systems in the local network,
- use of SSH connections,
- mounting remote systems via SSHFS,
- network diagnostics with the tools from Block A,
- DNS and reachability tests,
- controlled access to required internet resources.

Advanced Network Configuration does not perform the actual diagnosis of these connections, but their **configuration**.

## Functions

The program can do the following with network connections stored by NetworkManager:

- create them,
- edit them,
- remove them.

The documented connection types include, among others:

- Ethernet,
- Wi-Fi,
- bridge,
- bond,

- VLAN.

Depending on the connection type, the associated configuration options are available.

For corresponding network connections, these include in particular settings for **IPv4 and IPv6**.

Creating, changing or removing connections requires a running NetworkManager.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Determine the existing network environment and the desired connection.
2. Open **Advanced Network Configuration**.
3. Select an existing connection profile or create a new profile.
4. Enter or check the required connection settings.
5. Save the changes.
6. Establish the desired network connection.
7. Then check the reachability of the required target system or network service with suitable tools.

With manual IP configuration, the IP address, netmask or prefix, gateway and, where applicable, DNS configuration in particular must match the existing network environment.

## Typical rescue scenarios

### Manual network configuration

If no suitable automatic network configuration is available, a corresponding connection profile can be set up for JUMBO BLOCK Rescue.

This then enables access to required network resources.

### Preparing an SSH connection

Before **A.27 – SSH** can be used to access a remote system, JUMBO BLOCK Rescue must have a working network connection to the target.

Advanced Network Configuration can provide the local network configuration required for this.

### Preparing SSHFS

If a remote file system is to be mounted with **A.28 – SSHFS**, the corresponding network connection to the SSH server must first exist.

### Network diagnostics

A correctly configured network interface may be required for tools such as Nmap, Wireshark, TCPDump, ARP Scan, MTR or Dig.

Advanced Network Configuration creates the local configuration basis for this. The actual analysis is then carried out with the specialised tools from Block A.

### Working in separate network segments

If JUMBO BLOCK Rescue has to be used in an environment with specified addresses, gateways or other network settings, corresponding connection profiles can be set up specifically.

## Limitations and safety notes

Advanced Network Configuration is **not a network diagnostic or security analysis tool**.

For example, the program does not independently detect:

- compromised network devices,
- unauthorised network services,
- manipulated network traffic,
- malware,
- attacks within the network.

The corresponding analysis and security tools from Block A are intended for this.

Incorrect network settings can also result in network resources no longer being reachable or data traffic being routed via an unintended connection.

Particular care is required with unknown or possibly compromised networks.

**A technically working network connection does not mean that the network used is trustworthy.**

If no network connection is required for a rescue task, there is also no technical need to connect the rescue system to a network.

## Technical classification within JUMBO BLOCK Rescue

Advanced Network Configuration provides the graphical configuration layer for the rescue environment’s network connections.

The roles are clearly separated:

**C.7 – Advanced Network Configuration**
configures the network connections of JUMBO BLOCK Rescue.

**A.17 to A.23 – network tools**
examine network structures, connections, data traffic, routing and name resolution.

**A.27 – SSH and A.28 – SSHFS**
use an existing network connection for remote access or for mounting remote file systems.

Advanced Network Configuration thus creates the technical basis for numerous network-based rescue tasks without performing their analysis function itself.

**Key rule:**

**Only set up network connections that are needed – check configuration values before saving – do not automatically treat unknown networks as trustworthy – then check the reachability actually required specifically.**

# C.8 – Xfce Terminal

**Area:** System
**Program:** Xfce Terminal
**Baseline 20261001:** 1.1.4 or later

## Purpose

**Xfce Terminal** is a graphical terminal emulator of the Xfce desktop environment.

In JUMBO BLOCK Rescue, Xfce Terminal provides a graphical environment for running shell commands, system utilities and terminal-based rescue tools.

The terminal itself does not perform data recovery, diagnostics or repairs. It provides the interface in which the relevant commands and programs are run.

## Use in the rescue system

A significant proportion of the technical rescue tools can be used via the command line.

Xfce Terminal enables, for example:

- running diagnostic commands,
- starting terminal-based rescue tools,
- working with storage device and file system commands,
- running network commands,
- using SSH and SSHFS,
- calculating and checking technical information,
- navigating within directory structures,
- controlled execution of administrative commands,
- documenting or transferring command output.

The terminal is therefore a central interface between the graphical rescue environment and the system’s command-line tools.

## Functions

The basic functions of Xfce Terminal include, among others:

- running a shell within a graphical window,
- several terminal tabs,
- several terminal windows,
- copying and pasting text,

- search function within the terminal output,
- customising the display,
- full-screen mode,
- scrollback for earlier terminal output.

Several tabs make it possible, for example, to keep different work steps or diagnostic tools open in parallel.

## Operation and workflow

A typical rescue workflow is:

1. Open Xfce Terminal.
2. Determine the current working environment or required directory.
3. Before a writing operation, identify source and target unambiguously.
4. Enter the desired command completely.
5. Check the command once more before running it.
6. Run the command.
7. Check the output and any error messages completely.
8. Only then carry out the next work step.

With more complex commands, particular attention should be paid to device names, paths, options and target specifications.

A syntactically valid command is not automatically the right command for the intended storage device or the desired target system.

## Typical rescue scenarios

### Starting command-line tools

Numerous programs from **Block A – JB-Rescue** can be started directly in the terminal and used with specific options.

These include, for example, tools for:

- storage device diagnostics,
- data recovery,
- network analysis,
- hardware diagnostics,
- encryption,
- remote access.

### Checking storage devices and file systems

Devices, partitions, mount points and file systems can be determined with suitable system commands before further rescue steps are taken.

This is particularly important before writing or destructive operations.

### Remote work

**A.27 – SSH** is typically used within a terminal.

**A.28 – SSHFS** can also be set up and checked via the command line.

### Evaluating technical information

Output from diagnostic programs can be read directly in the terminal and used for further work steps.

### Working with administrative rights

Certain rescue tasks require elevated permissions.

The terminal enables corresponding commands to be run within the system's intended rights management.

Elevated permissions should only be used if they are required for the specific work step.

## Limitations and safety notes

Xfce Terminal is **not a rescue tool in the narrower sense**.

The effect of an action is determined by the command run within the terminal.

The terminal can therefore be used both for purely reading diagnostics and for far-reaching changes to storage devices and systems.

Particular care is therefore required with commands that:

- delete or overwrite files,
- write directly to storage devices,
- change partition tables,
- create or change file systems,
- mount or unmount storage devices,
- change access rights,
- edit system configurations.

With administrative rights, incorrect commands can have far-reaching effects.

Commands from unknown sources should therefore not be adopted and run without checking.

Likewise, device names such as `/dev/sda`, `/dev/sdb` or `/dev/nvme...` should never be assumed solely on the basis of a supposedly known order. The actual assignment must be checked in the system concerned.

## Technical classification within JUMBO BLOCK Rescue

Xfce Terminal provides the rescue environment's general graphical command-line interface.

It is important to distinguish these roles:

**C.8 – Xfce Terminal**
provides the terminal environment.

**The shell**
interprets the commands entered.

**The respective rescue and system programs**
carry out the actual diagnostic, backup, analysis or repair functions.

Xfce Terminal is thus not a specialised rescue tool, but an important working interface for numerous technical functions of JUMBO BLOCK Rescue.

Particularly with writing storage device operations, the same basic principle applies as throughout the rescue system:

**Identify and diagnose first – preserve originals as unchanged as possible – with storage devices at risk, back up first – only run writing commands deliberately and after unambiguously checking source and target.**

**Key rule:**

**Understand the command – check source and target – use the required rights deliberately – only then run it – check the result and error messages.**

# C.9 – Screenshot

**Area:** Documentation
**Program:** Screenshot / Xfce4 Screenshooter
**Baseline 20261001:** 1.11.1 or later

## Purpose

**Screenshot** (`xfce4-screenshooter`) is the screen capture tool of the Xfce desktop environment.

In JUMBO BLOCK Rescue, Screenshot is used to visually document screen states, error messages, diagnostic results, program settings and other information displayed during rescue work.

This provides a visual record alongside logs, text files and technical verification results.

## Use in the rescue system

During a diagnosis or data recovery, information can appear on the screen that is relevant for later documentation or for the traceability of the workflow.

Screenshot can be used, for example, to:

- record error messages,
- document diagnostic results,
- capture program settings,
- save storage device or partition displays,
- record network and hardware information visually,
- document intermediate results of a rescue process,
- compare states before and after a measure.

Screenshots therefore complement the technical documentation of rescue work.

## Functions

Xfce4 Screenshooter enables screen captures of different areas.

These include in particular:

- capture of the entire screen,
- capture of the active window,
- capture of a selected screen area,
- delayed capture,
- saving the screen capture created.

Which type of capture is used depends on which information is to be documented.

For technical documentation, capturing a single window or a specifically selected area is often sufficient.

## Operation and workflow

A typical workflow in JUMBO BLOCK Rescue is:

1. Bring about the state to be documented on the screen.
2. Check which information is actually needed for the documentation.
3. Make sure that no unnecessary confidential or personal information is visible.
4. Start Screenshot.
5. Select the entire screen, a window or the desired area.
6. Create the capture.
7. Save the file in a suitable working or documentation directory.
8. Then check the capture for legibility and completeness.
9. If required, back it up together with logs, checksums or other examination results.

With several captures, unambiguous and traceable file naming is advisable.

## Typical rescue scenarios

### Documenting an error message

An error message displayed during system startup, a diagnosis or a program run can be recorded immediately as a screen capture.

The visible message is thus preserved even after the program has been closed.

### Documenting the state of a storage device

Displays from GParted, GNOME Disks or other graphical diagnostic tools can be saved as supplementary visual documentation.

### Documenting settings

Before important settings are changed, the initial state can be recorded as a screenshot.

After the change, a further capture can document the state then visible.

### Documenting analysis results

Graphically displayed results or relevant terminal output can additionally be saved as a screen capture if required.

With extensive terminal output or technical measured values, however, the actual text or log data should preferably be preserved.

### Handover of a rescue process

Screenshots can be part of technical handover documentation together with logs, checksums, rescued configuration files and other examination results.

## Limitations and safety notes

A screenshot only shows the **visible screen state at the time of capture**.

It does not automatically prove:

- the technical cause of a displayed error,
- the completeness of a diagnosis,
- the integrity of an examined system,
- the authenticity of the displayed data,
- the unchanged state of a storage device or file.

A screen capture therefore does not replace technical original data.

If, for example, log files, diagnostic output or configuration files are available, these should additionally or preferably be backed up in their original form.

For integrity checks, hash values can additionally be documented with **C.6 – GtkHash**.

Particular attention must also be paid to confidential information.

Screenshots can contain, for example:

- user names,
- file names and directory structures,
- IP addresses,
- system information,
- personal data,
- contents of open documents,
- other information not intended for passing on.

Before passing them on, it should therefore be checked which information is actually visible on the capture.

## Technical classification within JUMBO BLOCK Rescue

Within Block C, Screenshot provides visual documentation of the rescue environment.

The roles are clearly separated:

**Technical rescue and diagnostic tools**
produce the actual examination results.

**C.5 – FeatherPad**
can be used for text-based notes and documentation.

**C.6 – GtkHash**
enables integrity checking of files.

**C.9 – Screenshot**
records a visible screen state as an image file.

Screenshot is therefore a supplementary documentation tool. It is particularly helpful when a graphically displayed state or a message that is only visible temporarily is to be preserved.

However, a screenshot should always be understood as **supplementary visual documentation** and not as a substitute for the underlying technical data.

**Key rule:**

**Only capture relevant information – pay attention to confidential screen content – also preserve technical original data – use a screenshot as supplementary documentation and not as the sole technical proof.**

# Block D

In addition to the specialised rescue tools from **Block A – JB-Rescue**, the selected MX tools from **Block B** and the general working tools from **Block C – Accessories**, JUMBO BLOCK Rescue contains the complete **JUMBO BLOCK Engineering Suite**.

The Engineering Suite provides the JUMBO BLOCK programs intended for planning, calculation, project work and document work within the local rescue environment. These include in particular tools for hydrology, geometric planning, costs, structural design, temperature, inspection, project management and tendering, as well as supplementary programs for calculations, units, spreadsheets, texts, documents, PDF files, archives and file management.

JUMBO BLOCK Rescue therefore does not serve exclusively for the technical rescue of failed systems. The integrated Engineering Suite also makes it possible to continue essential work processes directly within the rescue environment, independently of a failed workstation.

The programs of the Engineering Suite are deliberately described concisely in this manual. The separate **JUMBO BLOCK manuals** are available for complete operation, all functions and program-specific technical notes.

## Program groups

**20 programs of the JUMBO BLOCK Engineering Suite:**

**D.1 – Hydro**
Hydrological preliminary planning for rainfall events, retention, storage volume and service water use.

**D.2 – Plan**
Geometric project planning, module count, areas, volumes and quantity determination.

**D.3 – Budget**
Project costing for materials, transport, installation and overall overview.

**D.4 – Static**
Structural design and FE calculation of load and installation situations.

**D.5 – Temp**
Thermal infrastructure balance for assessing heat storage, temperature development, site relief and thermal infrastructure.

**D.6 – Inspect**
Local maintenance and inspection log for JUMBO BLOCK installations with check catalogue, inspections, measured and limit values, photo documentation, defects and reports.

**D.7 – Project**
Project management for JUMBO BLOCK engineering projects and feasibility studies, including responsibilities, deadlines, partners and documents.

**D.8 – Tender**
Local tool for creating and editing GAEB DA XML BoQs of data phases X80 to X84.

**D.9 – Calc**
Universal engineering calculator for general, geometric, scientific and JUMBO BLOCK-specific calculations.

**D.10 – Units**
Offline unit converter for engineering, physics and technical applications.

**D.11 – Note**
Local note-taking program for notes, search, import and export, and links to web pages and documents.

**D.12 – Tables**
Offline spreadsheet with formula engine, multi-sheet support, cell formatting, charts and various export options.

**D.13 – Writer**
Resilient offline writer for creating and editing documents completely locally.

**D.14 – Reader**
Offline reader for e-books and technical documents in various file and document formats.

**D.15 – Text**
Offline editor for text and source code files with syntax highlighting, line numbers, and search and editing functions.

**D.16 – PDF Merge**
Local tool for merging several PDF files and for sorting and removing individual pages.

**D.17 – PDF Printer**
Local tool for the controlled printing of existing PDF files with preview and print settings.

**D.18 – Convert**
Local converter for converting PDF files into editable Word documents.

**D.19 – Archive**
Offline tool for creating and extracting supported archive formats.

**D.20 – Files**
Offline file manager for local folders, USB sticks and drives with file operations, preview and search functions.

Together, the 20 programs form the **engineering and working environment** available within JUMBO BLOCK Rescue.

While **D.1 to D.8** mainly cover the specialist engineering, installation and project processes, **D.9 to D.20** provide general calculation, document, file and working functions.

**Block D thus makes it possible, after JUMBO BLOCK Rescue has been started, not only to examine and rescue systems and data but also to continue essential JUMBO BLOCK engineering and document processes completely locally.**

## Assignment of areas, programs and versions

| No. | Area | Program | Baseline version |
|---|---|---|---|
| D.1 | Hydrology | Hydro | 9.3.1 or later |
| D.2 | Planning | Plan | 9.2.3 or later |
| D.3 | Costs | Budget | 9.2.2 or later |
| D.4 | Structural design | Static | 11.0.2 or later |
| D.5 | Temperature | Temp | 9.0.0 or later |
| D.6 | Inspection | Inspect | 1.0.0 or later |
| D.7 | Project management | Project | 1.1.2 or later |
| D.8 | Tendering | Tender | 4.0.0 or later |
| D.9 | Calculation | Calc | 2.0.0 or later |
| D.10 | Units | Units | 2.0.0 or later |
| D.11 | Notes | Note | 1.0.0 or later |
| D.12 | Spreadsheets | Tables | 2.0.0 or later |
| D.13 | Documents | Writer | 10.1.0 or later |
| D.14 | Documents | Reader | 1.0.4 or later |
| D.15 | Text / Code | Text | 1.0.0 or later |
| D.16 | PDF | PDF Merge | 1.0.0 or later |
| D.17 | PDF | PDF Printer | 1.2.1 or later |
| D.18 | Conversion | Convert | 0.3.2 or later |

| No. | Area | Program | Baseline version |
|---|---|---|---|
| D.19 | Archives | Archive | 1.1.1 or later |
| D.20 | Files | Files | 1.0.0 or later |

# D.1 – Hydro

**Area:** Hydrology
**Program:** JUMBO BLOCK Hydro
**Baseline 20261001:** 9.3.1 or later

## Purpose

**JUMBO BLOCK Hydro** is the hydrological estimation tool of the JUMBO BLOCK Engineering Suite.

The program provides an initial technical assessment of urban rainwater retention and links hydrological input data directly with the modular JUMBO BLOCK installation.

Its essential tasks include:

- surface analysis,
- design rainfall,
- storage verification according to DWA-A 117,
- determination of rainfall and storage volume,
- calculation of the required number of modules,
- heavy rain scenarios,
- urban street model,
- project and emergency planning.

## Role within the Engineering Suite

Hydro forms the **hydrological basis** of the Engineering Suite.

The required storage demand is determined from the project area, degree of sealing, runoff coefficient and site-specific rainfall data and transferred to a real JUMBO BLOCK module installation.

Hydro can be used on its own, but at the same time it is part of the shared data model of the Engineering Suite.

Further geometric and technical work on a project can then be continued in other programs of the suite.

## Essential functions

Hydro supports two independent hydrological assessments.

The simplified method determines the rainfall volume from the effective area, rainfall depth, site factor and runoff coefficient, and from this the required number of modules.

In addition, Hydro carries out a comparative **DWA-A 117 storage verification** over five duration stages. The throttled outflow and, where applicable, the infiltration capacity are taken into account, and the duration stage with the greatest required storage demand is determined as the governing stage.

The system dimensioning transfers the determined storage demand to JUMBO BLOCK modules. The net volume of the standard module is stored as **13,9 m³**.

Other functions include, among others:

- project summary,
- heavy rain simulation,
- urban street model,
- emergency planning,
- infrastructure readiness,
- Hydrological Report,
- German/English switching,
- contextual specialist help.

## Data and data exchange

Hydro works with the shared project file:

**project.dbm**

This project file is also used by other programs of the JUMBO BLOCK Engineering Suite.

On export, an updated **SHA-256 hash** is generated over the saved data set and saved together with the project data.

The file name follows the pattern:

**YYYYMMDD_HHMMSS_project.dbm**

On import, Hydro checks the stored hash and adopts the data relevant to Hydro from the shared project file.

This includes in particular:

- project data,
- surface analysis,
- design rainfall,
- urban street model,
- DWA-A 117 inputs,
- associated orientation values,

- available shared information on the planned number of modules.

A shared module count set by **Plan** is adopted in preference.

After import and export, a recalculation takes place so that results and assessments correspond to the current data.

## Local operation and significance for JUMBO BLOCK Rescue

Hydro runs entirely locally in the browser.

In particular, the following are not required for using the program itself:

- cloud connection,
- server,
- user account.

In addition, rainfall data is not assigned to locations automatically.

The specialist rainfall data – in particular the **KOSTRA-DWD-2020 rainfall depths** used – is deliberately entered manually as checked site-specific values.

Hydro can therefore be used for hydrological project work within JUMBO BLOCK Rescue even when the regular workstation or central IT services are not available.

An existing project can be taken over via its `project.dbm` and further edited within the local rescue environment.

New projects can also be started completely locally and then saved again as `project.dbm`.

## Data backup

Hydro does not use persistent browser-based caching for the current project state.

During work, the current state is held in the memory of the open page.

Unexported changes can therefore be lost when the page is closed or reloaded.

For work within JUMBO BLOCK Rescue, the following rule therefore applies:

**Regularly export project states as project.dbm and back them up to a suitable storage device.**

The SHA-256 hash within the project file serves the traceability of changes. A hash deviation is displayed on import.

However, the hash prevents neither the import nor a subsequent calculation.

## Reports

Via **Print / PDF**, Hydro generates a standalone **Hydrological Report**.

This contains, among other things:

- project information,
- design rainfall,
- surface analysis,
- DWA-A 117 storage verification,
- JUMBO BLOCK system dimensioning,
- heavy rain simulation,
- urban street model,
- summary and assessment,
- document information and hash values.

Before the report is generated, the current project state is completely recalculated.

## Limitations and notes

Hydro is a tool for hydrological **preliminary planning and technical orientation**.

The program does not replace approval planning or specialist planning required under water law.

In particular, site-specific KOSTRA grid values are not determined automatically. The rainfall depths used must be entered as checked specialist values.

Orientation values for throttled outflow and infiltration replace neither consultation with the competent authority or municipal drainage requirements nor necessary site-specific investigations.

The simplified method and the DWA-A 117 storage verification are different calculation approaches and can therefore result in different storage requirements. Both results must be considered according to their respective calculation basis.

## Technical classification in JUMBO BLOCK Rescue

Within Block D, Hydro provides the **hydrological engineering function**.

While Blocks A to C mainly serve the rescue, examination, backup and technical processing of systems and data, Hydro enables hydrological JUMBO BLOCK project work to be continued or resumed within the local rescue environment.

The shared `project.dbm` forms the link to other programs of the Engineering Suite.

**Hydro → hydrological basis and storage demand**

**Plan → geometric project planning**

**Static → technical and structural design**

**Budget → project costing**

**Temp → temperature and thermal infrastructure balance**

The specialist project chain thus remains available within JUMBO BLOCK Rescue as well.

## Further documentation

Complete operation, calculation principles, input fields, default values, formulas, reports and other technical notes are documented in the separate manual:

JUMBO BLOCK Hydro – User Manual · Version 9.3.1

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Hydro” at **jumboblock.org**.

# D.2 – Plan

**Area:** Planning
**Program:** JUMBO BLOCK Plan
**Baseline 20261001:** 9.2.3 or later

## Purpose

**JUMBO BLOCK Plan** is the geometric preliminary planning tool of the JUMBO BLOCK Engineering Suite.

The program translates a required retention volume – typically taken from **Hydro** – or specified installation dimensions into a specific modular JUMBO BLOCK installation layout.

In particular, the following are determined:

- module count,
- block raster,
- real installation dimensions,
- net storage volume,
- floor plates,
- side plates,
- cover plates,
- component volumes and weights,
- excavation volume,
- joint lengths of the SABA sealing.

## Role within the Engineering Suite

Plan forms the **geometric basis** of the Engineering Suite.

While Hydro determines the hydrologically required storage demand, Plan converts this demand into a specific installation geometry.

A typical workflow within the suite is:

**Hydro → Plan → Static → further project work**

The installation layout created in Plan can then be used in particular for structural design in **Static**, cost determination in **Budget** and further calculations in **Temp**.

## Essential functions

Plan has two basic planning modes.

### Planning via target volume

The required minimum number of JUMBO BLOCK modules is determined from a specified storage volume.

Plan then automatically suggests a standard raster that is as compact as possible.

The raster actually generated can contain an additional capacity reserve compared with the specified target volume.

### Planning via installation size

Alternatively, an installation can be developed from a specified length and width.

The raster is determined on the basis of the JUMBO BLOCK module raster of:

**2,5 × 2,5 × 2,5 m**

Complete 2,5 m raster steps are taken into account.

### Manual adjustment

Automatically determined values can be overridden specifically for special layouts.

This applies in particular to:

- block count,
- floor plates,
- side plates,
- cover plates.

As soon as such a manual override is active, the planning is treated as a **special layout**.

The thicknesses of the floor, side and cover plates can also be adjusted within the limits provided in the program.

### Installation layout

The calculated raster is displayed graphically.

This allows the module count, raster, footprint, storage volume and real installation dimensions to be checked immediately.

### SABA sealing

Plan takes into account the selected version of the SABA joint sealing and determines the required joint lengths from it.

Whether the cover plate joints are taken into account can be controlled separately.

## Data and data exchange

Like Hydro, Plan uses the shared project file:

**project.dbm**

A `project.dbm` exported from Hydro can be taken over in Plan.

If hydrological results are present, Plan adopts the relevant target volume from them. If a planned module count already exists, this can also be included in the further planning.

Plan can also read in a previously saved planning state of its own.

On export, the current planning data is written to the shared `project.dbm`. Existing data of other programs of the Engineering Suite is preserved.

The `project.dbm` thus serves as the shared project basis for passing on the planning state within the suite.

## Data integrity

Every `project.dbm` exported by Plan contains a **SHA-256 hash** over the saved data set.

This hash is checked automatically on import.

The planning report additionally contains separate hash values for:

- planning inputs,
- calculated results.

This makes the documented calculation state technically traceable.

## Local operation and significance for JUMBO BLOCK Rescue

Plan runs entirely locally in the browser.

In particular, the following are not required for using the program:

- cloud connection,
- server,
- user account.

Geometric project planning is thus available directly within JUMBO BLOCK Rescue.

An existing project can be taken over via its `project.dbm` and further edited. A new installation plan can also be started within the rescue environment.

This is particularly relevant if a regular engineering workstation is not available but project work must be continued with the existing project data.

## Data backup

Plan does not use persistent browser-based caching of the current project state.

Unexported changes can therefore be lost when the page is closed or reloaded.

For work within JUMBO BLOCK Rescue, the following rule therefore applies:

**Regularly export current planning states as project.dbm and back them up to a suitable storage device.**

This applies in particular before a planning is reset or the rescue environment is shut down.

## Reports

Via **Print / PDF**, Plan generates a standalone planning report.

This contains, among other things:

- project information,
- planning mode,
- installation layout,
- component information,
- joint information,
- theoretical reference values,
- input and result hash.

Before the report is generated, the current planning state is completely recalculated.

## Limitations and notes

Plan serves **technical and geometric preliminary planning**.

The program expressly does not replace project-specific structural design.

This is carried out in **Static**.

The theoretical reference values of the available plate concrete classes shown in Plan do not constitute structural verification either.

Costs are not calculated in Plan either. Within the Engineering Suite, the cost estimate is carried out with **Budget** on the basis of the installation layout created.

For sites or installations whose geometry deviates greatly from a compact standard raster, manual adjustments or planning via installation dimensions may be necessary.

## Technical classification in JUMBO BLOCK Rescue

Within Block D, Plan provides the **geometric engineering function**.

The basic division of tasks within the Engineering Suite is:

**Hydro → hydrological storage demand**

**Plan → module count, raster, installation geometry and quantities**

**Static → project-specific technical and structural design**

**Budget → project costing**

**Temp → temperature and thermal infrastructure balance**

Plan thus forms the link between the hydrologically determined demand and the specifically planned JUMBO BLOCK installation.

Within JUMBO BLOCK Rescue, this makes it possible to resume existing planning states from a `project.dbm` or to continue geometric project planning independently of a failed regular workstation.

## Further documentation

Complete operation, planning modes, input fields, raster determination, special layouts, SABA joint logic, calculation principles, reports and other technical notes are documented in the separate manual:

JUMBO BLOCK Plan – User Manual

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Plan” at **https://jumboblock.org/.**

# D.3 – Budget

**Area:** Costs
**Program:** JUMBO BLOCK Budget
**Baseline 20261001:** 9.2.2 or later

## Purpose

**JUMBO BLOCK Budget** is the tool for rough project costing within the JUMBO BLOCK Engineering Suite.

The program combines the previously determined installation and component quantities with the cost assumptions stored for the costing and produces a structured cost overview from them.

In particular, the following are taken into account:

- installation components,
- material quantities,
- component quantities and weights,
- transport,
- installation,
- other items relevant to costing,
- total costs of the project state under consideration.

Budget thus serves the economic assessment of a JUMBO BLOCK installation that has previously been determined technically or geometrically.

## Role within the Engineering Suite

Budget forms the **commercial costing level** of the Engineering Suite.

The required technical quantities are not re-planned independently in Budget but can be taken over from the upstream engineering programs.

Two different data states are possible here:

**Plan → Budget**
provides a costing basis on the basis of the geometric preliminary planning.

**Static → Budget**
enables more realistic costing on the basis of the components and quantities determined structurally or technically.

This keeps it traceable in Budget on which technical state of work the cost estimate concerned is based.

## Essential functions

Budget structures the project costs into individual line items that can be costed.

The costing combines technical quantities with the costs applied in each case.

The essential information of a line item includes in particular – where available for the line item concerned:

- description,
- size or dimensions,
- quantity,
- unit weight,
- line item weight,
- unit price,
- line item costs.

In addition, the relevant total weights and total costs are combined.

This makes it possible not only to estimate the monetary scope of the project but also to trace the order of magnitude of the components relevant to transport and project logistics.

## Data and data exchange

Budget is integrated into the shared data model of the JUMBO BLOCK Engineering Suite.

Project data from **Plan** or **Static** can be used as the technical starting basis.

The origin of the data is relevant from a technical point of view:

### Import from Plan

The planning state provides the geometrically determined installation and component quantities.

This variant is suitable for an early budget estimate before a more detailed structural design is available.

### Import from Static

If components and quantities determined technically or structurally are already available from Static, they can be used as the basis for a more realistic budget assessment.

This allows the costing to be updated as the technical state of work on the project progresses.

## Costing principle

Budget basically distinguishes between:

**technical quantities**
and

**commercial cost assumptions**.

The engineering programs provide the geometrically or technically determined quantities.

Budget assigns the price assumptions used for the costing to these quantities and calculates the line item and total costs from them.

A change in prices therefore does not change the technical installation planning. Conversely, changes to the module count, components or weights can change the costing basis.

This separation makes it possible to handle technical and economic changes separately and traceably.

## Local operation and significance for JUMBO BLOCK Rescue

Budget is part of the locally available JUMBO BLOCK Engineering Suite.

Project costing can therefore be continued or rebuilt within JUMBO BLOCK Rescue even when the regular engineering workstation or central IT services are not available.

Existing project data can serve as the basis for further costing.

This makes it possible, for example, to:

- resume an interrupted project costing,
- carry out a rough cost assessment of an existing planning state,
- update cost assumptions,
- create a new cost overview from rescued or existing project data,
- continue working locally during an IT failure.

Technical planning and costing thus remain linked within the rescue environment as well.

## Reports

Budget presents the costing in a structured cost overview.

The report documents the underlying project and costing state and summarises the relevant line items and the resulting costs.

The origin of the technical data is particularly important here, as costing based on a geometric preliminary planning from Plan has a different technical state of work from costing based on Static.

## Limitations and notes

Budget is a tool for **budget and cost estimation**.

The quality of the result depends directly on two foundations:

- the technical quantities taken over,
- the cost assumptions used.

Price changes, regional differences, delivery conditions, transport distances, project-specific boundary conditions or technical requirements changed at a later stage can therefore lead to actual project costs that differ.

Costing from **Plan** is based on an earlier technical state of work than costing from **Static**.

These two costing bases should therefore not be equated without taking their respective data basis into account.

Furthermore, Budget replaces neither geometric planning nor structural design.

These tasks remain with the engineering programs intended for them.

## Technical classification in JUMBO BLOCK Rescue

Within Block D, Budget provides the **economic engineering function**.

The basic division of tasks is:

**Hydro → hydrological storage demand**

**Plan → module count, raster, installation geometry and quantities**

**Static → technical and structural design**

**Budget → cost assessment of the technical project state concerned**

**Temp → temperature and thermal infrastructure balance**

Budget thus takes over the results of the technical project work and translates them into an economic view of the project.

Within JUMBO BLOCK Rescue, this means that after a failure not only can the technical planning be continued, but its financial order of magnitude can also be assessed and documented locally.

## Further documentation

Complete operation, costing structure, cost line items, imports, calculations, reports and other technical notes are documented in the separate manual:

JUMBO BLOCK Budget – User Manual · Version 9.2.2

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Budget” at **https://jumboblock.org/.**

# D.4 – Static

**Area:** Structural design
**Program:** JUMBO BLOCK Static
**Baseline 20261001:** 11.0.2 or later
**Status:** engineering service / information page

## Status within JUMBO BLOCK Rescue

**JUMBO BLOCK Static** occupies a special position within the Engineering Suite.

The application contained in JUMBO BLOCK Rescue is an **information and handover page for the JUMBO BLOCK engineering service**.

The actual structural design tool and the calculation models it contains are **internal engineering tools** and are not accessible to outside parties.

The Static page on JUMBO BLOCK Rescue therefore does not carry out any structural calculation itself.

It explains:

- the scope of the structural pre-design,
- the required input data,
- the procedure for handing over the project,
- the technical verifications considered,
- the reports subsequently provided.

The information page itself runs entirely locally in the browser and does not transmit any data.

## Purpose of the engineering service

JUMBO BLOCK Static serves the **structural pre-design, reinforcement planning and quantity determination for JUMBO BLOCK installations**.

From the previously created installation planning, JUMBO BLOCK Engineering develops a structural pre-design including reinforcement and quantities.

The basis is the project file:

**project.dbm**

The project file can come from **JUMBO BLOCK Plan or Budget**. The calculation itself is then carried out with the internal structural design tool of JUMBO BLOCK Engineering.

## Procedure

The procedure described on the information page consists of three steps:

### 1. Plan

The JUMBO BLOCK installation is first prepared in **Plan or Budget** and then exported as `project.dbm`.

### 2. Hand over project data

The `project.dbm` is handed over to JUMBO BLOCK Engineering together with the additional project information required for the structural pre-design.

### 3. Receive results

JUMBO BLOCK Engineering calculates the installation with the internal structural design tool.

The associated technical PDF reports are provided as the result.

## Scope of the engineering service

The installation is considered as an overall system in which modules, frame, plates and subsoil interact.

### Cover plate

For the cover plate, traffic loads from cars to heavy goods traffic, earth cover and additional loads, among others, are considered.

In particular, the information page names:

- load-bearing capacity,
- crack width,
- deflection,
- fatigue.

**SLW 60** and **F900**, among others, are named as load situations.

### Floor and side plates

For floor and side plates, the following, among others, are taken into account:

- earth pressure,
- water filling,
- groundwater.

The empty and filled states are considered separately; the less favourable state is governing.

### Overall system

Modules, surrounding frame and subsoil are considered together as a three-dimensional model of the entire installation.

### Stability

According to the information page, the structural pre-design includes:

- bearing capacity failure,
- overturning,
- sliding,
- safety against uplift,
- safety against flotation in groundwater.

### Durability and robustness

The following, among others, are considered:

- exposure class,
- concrete cover,
- concrete class,
- the installation's capacity for load redistribution in the event of local failure.

### Reinforcement and optimisation

The engineering service also includes the determination or optimisation of:

- plate thicknesses,
- standard reinforcement,
- frame reinforcement,
- material quantities,
- weights,
- order quantities.

These points describe the scope of the internal engineering service. They are **not publicly operable calculation functions of the information page contained on JUMBO BLOCK Rescue**.

## Required project data

The installation geometry is taken from the `project.dbm`.

Project-specific information is also required for the calculation.

In particular, the information page names:

- `project.dbm` from Plan or Budget with modules, raster and plate build-up,
- use of the surface,
- traffic load class, for example car, lorry or SLW 60,

- material and height of the earth cover above the cover plate,
- soil type or permissible ground pressure,
- existing geotechnical report,
- groundwater influence and groundwater level,
- mode of operation – retention with floor plate or infiltration without floor plate,
- ambient conditions for the exposure class, for example de-icing salt and frost,
- project details and contact persons for the reports.

According to the description on the information page, missing values are agreed together with the client.

## Results and reports

Three coordinated PDF reports are provided as the result of the engineering service.

### Structural Report

The Structural Report is the complete technical report with inputs, verifications and results.

It is intended in particular for designers and checking engineers.

### Production Printout

The Production Printout contains:

- reinforcement drawings for cover plates,
- reinforcement drawings for side plates,
- reinforcement drawings for floor plates,
- frame reinforcement,
- steel schedule with quantities and weights.

### Customer Report

The Customer Report summarises the results in an understandable form for building owners, municipalities and project communication.

The reports bear an **input checksum and a result checksum using SHA-256**. This allows each report to be assigned to the calculated project state.

## Limitations

The calculation is expressly classified as:

**structural pre-design and plausibility check of the engineering**

For use within JUMBO BLOCK Rescue, the following is also decisive:

**The integrated Static page does not carry out any structural calculations itself.**

It provides neither the internal calculation model nor a publicly operable interface for its calculations.

An existing `project.dbm` can be prepared and backed up within the rest of the Engineering Suite. For the actual structural pre-design, however, it must be handed over to JUMBO BLOCK Engineering together with the required additional project information.

## Significance within JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Static is therefore to be classified **not as a self-contained calculation program**, but as an interface between the project preparation possible locally and the internal JUMBO BLOCK engineering process.

The engineering chain is deliberately divided at this point:

**Hydro → local hydrological preliminary planning**

**Plan → local geometric project planning**

**Static → information and handover page for the internal engineering service**

The local rescue environment thus makes it possible to open existing project data, continue the planning and prepare or back up the `project.dbm` required for structural work.

The actual structural pre-design, by contrast, remains reserved for the internal JUMBO BLOCK engineering tool.

This distinction also prevents the availability of the Static information page within JUMBO BLOCK Rescue from being confused with the availability of the internal structural design tool.

## Further information

Further information on the scope of services and on the structural pre-design of JUMBO BLOCK installations:

**https://jumboblock.org/**

# D.5 – Temp

**Area:** Temperature
**Program:** JUMBO BLOCK Temp
**Baseline 20261001:** 9.0.0 or later
**Status:** engineering service / information page

## Status within JUMBO BLOCK Rescue

**JUMBO BLOCK Temp** occupies a special position within the Engineering Suite.

The application contained in JUMBO BLOCK Rescue is an **information and handover page for the JUMBO BLOCK engineering service**.

The actual tool for calculating the **Thermal Infrastructure Balance (TIB)** and the calculation models it contains are internal engineering tools and are not accessible to outside parties.

The Temp page on JUMBO BLOCK Rescue therefore does not carry out any thermal balancing itself.

It explains:

- the purpose and scope of the Thermal Infrastructure Balance,
- the procedure for handing over the project,
- the thermal quantities considered,
- the required project information,
- the results and reports subsequently provided.

The information page itself runs entirely locally in the browser and does not transmit any data.

## Purpose of the engineering service

JUMBO BLOCK Temp serves to produce a **Thermal Infrastructure Balance (TIB)** for JUMBO BLOCK installations.

The accessible, water-bearing infrastructure space below the surface is considered as a thermal system.

Under sealed surfaces, water and concrete form a thermal storage mass. The TIB considers this relationship with regard to heat storage, medium temperatures, site relief and the use of the structure as an infrastructure space.

The basis for the work is the project file:

**project.dbm**

The project file can come from **JUMBO BLOCK Plan or Budget**.

On this basis, JUMBO BLOCK Engineering produces the Thermal Infrastructure Balance of the installation.

## Procedure

The procedure described on the information page consists of three steps:

### 1. Plan

The JUMBO BLOCK installation is first prepared in **Plan or Budget** and then exported as `project.dbm`.

### 2. Hand over project data

The `project.dbm` is handed over to JUMBO BLOCK Engineering together with the additional information required.

### 3. Receive results

JUMBO BLOCK Engineering balances the installation with the internal Temp tool.

A **TIB report with key figures and charts as a PDF** is provided as the result.

## Scope of the engineering service

The Thermal Infrastructure Balance considers various thermally effective components and processes of the installation.

### Heat storage

The following are taken into account as thermal storage masses:

- water,
- module concrete,
- plates,
- backfill.

The assessment covers the water space and dry space as well as an optional flooding mode.

### Heat dissipation

The balance takes into account:

- evaporation,
- heat transfer to the ground,
- convection to the air.

### Heat inputs

The following, among others, can be taken into account as heat inputs:

- solar radiation via the surface,
- cables,
- waste heat pipes,
- utility lines,
- technical equipment in the infrastructure space.

### Medium temperatures and operation

Temperatures of air and water in the water space and dry space and their significance for operation and maintenance are considered.

### Site effect

The TIB considers the local relief of the surface and the surface temperature with and without JUMBO BLOCK.

### Site TIB

As a further parameter, the degree of thermal compensation in relation to the heat inputs of the site and its technical installations is considered.

These points describe the scope of the internal engineering service. They are **not publicly operable calculation functions of the information page contained on JUMBO BLOCK Rescue**.

## Required project data

The structure geometry is taken from the `project.dbm`.

Project-specific information is also required for the thermal balance.

In particular, the information page names:

- `project.dbm` from Plan or Budget with modules and plate build-up,
- division into water space and dry space,
- water filling,
- flooding mode, if applicable,
- sealed area of the entire site,
- known heat inputs of the site from buildings and technical installations,
- number, length and power loss of planned cables,
- planned waste heat and utility lines,
- technical equipment and its power,
- backfill or build-up above the cover plate,

- desired climate profile,
- desired assessment period.

According to the description on the information page, missing values are agreed together with the client.

## Results and reports

A TIB report is provided as the result of the engineering service.

### TIB report

The report contains a core balance with:

- thermal reserve and utilisation,
- medium temperatures,
- operation and maintenance,
- site effect,
- site TIB,
- charts,
- traffic-light assessment.

### Detailed balance

The detailed balance compiles the individual items of the heat balance for planning and checking.

### Climate profiles

The installation can be assessed for typical conditions ranging from winter days through the annual average to summer and hot days.

### Infrastructure scenarios

Both conventional rainwater retention and an infrastructure space with cables, lines and technical equipment, as well as corresponding combinations, can be considered.

The report bears an **input checksum and a result checksum using SHA-256**. This allows the report to be assigned to the calculated project state.

## Limitations

The TIB is expressly classified as:

**simplified technical preliminary estimate**

In particular, it does not replace:

- site-specific thermal simulation,
- electrical planning,
- sealing planning,
- fire protection review,
- KRITIS review,
- assessment under water law.

For use within JUMBO BLOCK Rescue, the following is also decisive:

**The integrated Temp page does not carry out any TIB calculation itself.**

It provides neither the internal calculation model nor a publicly operable interface for thermal balancing.

The required `project.dbm` can be prepared or backed up with the other applications of the Engineering Suite. The actual thermal balancing is then carried out by JUMBO BLOCK Engineering.

## Significance within JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Temp is therefore to be classified **not as a self-contained calculation program**, but as an interface between the project preparation possible locally and the internal thermal engineering process.

The engineering chain is deliberately divided at this point:

**Hydro → local hydrological preliminary planning**

**Plan → local geometric project planning**

**Temp → information and handover page for the internal Thermal Infrastructure Balance**

Within the rescue environment, existing project data can thus be backed up and the basis required for a TIB can be prepared.

The actual thermal balancing, by contrast, remains reserved for the internal JUMBO BLOCK engineering tool.

This distinction prevents the availability of the Temp information page within JUMBO BLOCK Rescue from being confused with the availability of the internal TIB calculation tool.

## Further information

Further information on the Thermal Infrastructure Balance and the JUMBO BLOCK engineering service:

**https://jumboblock.org/**

# D.6 – Inspect

**Area:** Inspection
**Program:** JUMBO BLOCK Inspect
**Baseline 20261001:** 1.0.0 or later

## Purpose

**JUMBO BLOCK Inspect** is the maintenance and inspection log of the JUMBO BLOCK Engineering Suite.

The program documents the operation of an installation over its entire service life. In particular, it records:

- existing installations and components,
- intended checks,
- check intervals and due dates,
- inspections carried out,
- measured values and check results,
- photo documentation,
- defects found,
- measures and defect remedies.

Inspect thus links the previously generated engineering data with the later operational documentation of the installation actually present.

## Role within the Engineering Suite

Within the Engineering Suite, Inspect handles **maintenance, inspection and life-cycle documentation**.

The `project.dbm` from the upstream engineering work can be read in as the starting basis.

From it, Inspect reads base data from **Hydro, Plan, Static and Temp** and can generate item suggestions for the asset tree on the basis of this information.

The project file is only read. The data of the upstream engineering programs is not changed by Inspect.

After the import, the operational documentation is kept in a separate file:

**inspect.dbm**

Engineering project data and the later maintenance and inspection history thus remain separate from each other.

# Essential functions

## Base data from project planning

When a `project.dbm` is read in, the following, among others, can be adopted:

- project name,
- location,
- client,
- designer,
- module count,
- grid,
- installation dimensions,
- net storage,
- slabs,
- sealing,
- design values from Hydro and DWA-A 117,
- traffic load,
- groundwater,
- concrete class,
- Temp infrastructure.

The SHA-256 of the `project.dbm` read in is recalculated. The result is displayed and saved together with the source.

## Asset tree

Inspect represents the real installation in a hierarchical asset tree.

Items can be nested as deeply as required, for example:

**Installation → Shaft → Ladder**

Items have a type, a designation, a quantity and – where necessary – a grid.

Additional components that are not part of the `project.dbm` can be added from the catalogue. These include, for example, shafts, inlets, pre-treatment and accessories.

Items can be added, replaced, marked as withdrawn or – provided that no associated logs or defects exist yet – deleted.

Grid information enables defects to be assigned to an exact location within the installation.

### Design values and limits

For individual check points, the following can be specified per item:

- design values,
- units,
- labels,
- minimum values,
- maximum values.

Measured values outside defined limits can automatically be assessed as **“not OK”**.

Changes to design values and limits are documented in the history with the date and the old and new value.

Completed logs, by contrast, keep the values valid at the time of the check concerned.

### Check catalogue

Inspect contains an extensible check catalogue.

The check points can use the following check types, among others:

- visual check,
- function test,
- cleaning/service,
- measurement,
- selection,
- free text,
- photo.

In addition, intervals, event checks, units, limits, selection options, safety notes and sources can be stored.

The intervals of the standard catalogue are suggestions. They do not replace manufacturer, operator or regulatory requirements and must be checked and, where necessary, adjusted for the installation concerned.

Custom types and check points can be added. Catalogues can also be exported and imported.

### Due dates

For check points with a specified interval, Inspect calculates the next due date.

Among other things, a distinction is made between:

- overdue,

- due within 30 days,
- first-time without a start date,
- OK.

The commissioning date or the last completed check can serve as the starting point.

In addition, Inspect supports event-related checks and checks after structural changes.

### Inspection reasons

Inspect distinguishes between different reasons for inspection.

The standard ones are:

- annual inspection,
- inspection after event,
- inspection after structural change,
- acceptance / initial inspection,
- warranty inspection,
- inspection after repair,
- special inspection.

Custom inspection reasons can be added and provided with additional mandatory fields.

### Carrying out an inspection

A new inspection is first created as a draft.

Depending on the selected inspection reason, the intended check points are compiled.

For the individual checks, the following, among others, can be recorded:

- **OK**
- **not OK**
- **n/a**
- measured values,
- free text,
- photos,
- defects.

For measured values, the stored design values and limits can be displayed directly at the check point.

### Photo documentation

Photos can be embedded directly in the inspection file.

Inspect reduces the images according to the selected settings. EXIF information such as capture time, GPS position, altitude, viewing direction and device can be preserved and appear in the report below the photo concerned.

When passing on a report, GPS coordinates can be hidden. The data is preserved in the actual Inspect file.

### Defect management

Defects can be recorded during an inspection or directly in the Defects area.

A defect can include, among other things:

- affected item,
- severity,
- description,
- deadline,
- grid cell,
- component,
- photos.

After processing, the measure, the person responsible, the completion date, the evidence and further photos can also be documented.

Completed defects can become part of a later **inspection after repair**.

## Data and data exchange

Inspect uses two clearly separated data levels.

### Input data

**project.dbm**

It serves to adopt project and engineering base data.

When a changed project file is read in again, changes can be detected and existing items or design values handled accordingly.

Items are not deleted automatically when changes occur. Items that no longer apply can be marked as withdrawn if desired.

### Inspection data

**inspect.dbm**

An Inspect file belongs to one installation and contains, among other things:

- master data,
- source,

- check catalogue,
- asset tree,
- inspections,
- defects,
- embedded media,
- settings.

In addition, the following, among others, are available:

- CSV export of defects,
- CSV export of due dates,
- catalogue export,
- catalogue import.

## Data integrity

Inspect uses several SHA-256-based checking mechanisms.

### File hash

The `inspect.dbm` has a SHA-256 file hash based on the same basic principle as the `project.dbm`.

### Chained inspection logs

When an inspection is completed, the log becomes unchangeable.

Inspect assigns a sequential number and stores the hash of the previous completed log. A new SHA-256 is then generated over the complete log.

This creates a **chained log history**.

### Photos

Each embedded image is stored under its own SHA-256 and checked again when the file is opened.

The integrity check can thus establish whether logs, images or metadata have been changed subsequently.

However, it cannot prove that information originally entered or, for example, an EXIF entry already present in the photo was correct in content.

## Local operation and significance for JUMBO BLOCK Rescue

Inspect runs completely offline in the browser and can be used directly from the USB stick via `file://`.

It requires:

- no user account,
- no cloud,
- no central server.

The inspection data remains locally on the system or storage device used.

Inspect is therefore particularly relevant for JUMBO BLOCK Rescue, because existing installation documents and inspection information can be continued locally even if the regular IT infrastructure fails.

An existing `project.dbm` can be read in and used as the basis for new or existing installation documentation.

The actual maintenance and inspection history is preserved in the separate `inspect.dbm`.

## Data backup

Inspect additionally keeps a local interim state in the browser.

However, this interim state does **not** replace saving the `inspect.dbm`.

Unsaved changes are displayed in the status bar.

As photos are embedded directly in the file, the size of the `inspect.dbm` can grow considerably as the number of inspections increases. Inspect issues a notice from a file size of about **50 MB**.

For reliable documentation, the current `inspect.dbm` should therefore be saved and backed up regularly.

## Reports

Inspect provides three A4 reports.

### Inspection report

The inspection report contains, among other things:

- installation and base data,
- inspection reason,
- scope and additional fields,
- results per item,
- design values and limits,
- photos,
- EXIF/GPS details,
- defects,

- hash verification,
- signature fields.

### Maintenance and inspection plan

The maintenance and inspection plan brings together the check points with:

- design values and limits,
- check intervals,
- last check,
- next check.

### Defect list

The defect list documents open and completed defects, including any photos.

The reports can be printed or saved as PDF via the browser's print dialog.

## Limitations and notes

Inspect is a tool for structured maintenance and inspection documentation.

The check intervals contained in the standard catalogue are expressly **suggestions**. They do not replace binding:

- manufacturer requirements,
- operator requirements,
- regulatory requirements.

These must be checked and adjusted for the installation concerned.

The SHA-256 checks prove whether saved logs, images and metadata have remained unchanged since they were completed or attached. They do not automatically confirm the correctness in content of the information originally recorded.

Photos in **HEIC format** may not be readable by the browser. The manual therefore recommends JPEG or prior export as JPEG.

## Technical classification in JUMBO BLOCK Rescue

Inspect extends the engineering chain beyond planning and calculation into the **operation and life cycle of the installation**.

The functional assignment is:

**Hydro / Plan / Static / Temp → technical project and installation basis**

**Inspect → installation structure, check planning, inspections, measured values, photo documentation, defects and maintenance history**

Existing engineering data can thus not only be rescued and processed further. Within the rescue environment, it can also serve as the starting point for the operational documentation of an installation.

The separate `inspect.dbm`, the integrity check and the chained completed inspection logs support traceable local documentation even without cloud or server infrastructure.

## Further documentation

JUMBO BLOCK INSPECT – Maintenance and inspection log · Manual · Version 1.0.0

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Inspect” at **https://jumboblock.org/.**

# D.7 – Project

**Area:** Project management
**Program:** JUMBO BLOCK Project
**Baseline 20261001:** 1.1.2 or later

## Purpose

**JUMBO BLOCK Project** is the project management tool of the JUMBO BLOCK Engineering Suite.

It serves to organise **engineering projects and feasibility studies** and structures a project in three levels:

- project,
- project sections,
- project steps.

Sections and steps are linked with responsibilities, dates and processing status.

The program thus combines the technical project work of the Engineering Suite with structured organisational project management.

## Role within the Engineering Suite

Project handles the **organisational project management level**.

While programs such as Hydro, Plan, Budget, Static and Temp represent specialist engineering tasks, Project manages the overall project workflow.

This includes in particular:

- project master data,
- project sections and project steps,
- responsibilities,
- executing persons or partners,
- status,
- dates,
- project partners,
- project-related documents and original files,
- status history.

Several people can work independently of one another with copies of a project. The different states can then be merged again via an **additive import**.

# Essential functions

## Project master data

For the overall project, the following, among others, can be recorded:

- project name,
- client / customer,
- location,
- project phase,
- overall responsible person,
- overall executor.

The following project phases are available:

- Acquisition,
- Planning,
- Permitting,
- Execution,
- Acceptance.

## Project sections and project steps

A project can be divided into any number of project sections.

Within a section, any number of individual project steps can in turn be created.

This makes it possible, for example, to separate planning, permitting procedures and execution from one another and then subdivide them into specific work steps.

Sections and steps can be expanded and collapsed for a better overview.

## Responsible and Executor

Every section and every project step has two roles:

**Responsible**
bears responsibility for the entry concerned.

**Executor**
carries out the actual work.

Both roles can be assigned to the same or to different persons or project partners.

If either of the two assignments is missing, Project marks the entry with:

**“⚠ unassigned”**

Open responsibilities thus remain immediately visible.

### Status and dates

Three statuses are available for sections and steps:

- Open,
- In progress,
- Done.

A date does not necessarily have to be specified at the start.

However, if an entry is to be set to **“Done”**, a date is required.

An optional start date can also be stored.

The **“Sequence”** field enables supplementary documentary information such as “after soil survey”. However, this information does not control the project workflow automatically.

### Dates overview

Project combines all sections and steps with a stored date in a chronological dates overview.

The following are displayed:

- date,
- title,
- type,
- status,
- responsible person,
- executor.

Overdue entries whose date has already passed and whose status is not yet “Done” are highlighted.

### Automatic timeline

From the stored start and end dates, Project automatically generates a **Gantt view**.

Sections form the higher-level rows, and the associated project steps are displayed below them.

The bars show the period from the start date to the specified date.

The status concerned is shown visually in the timeline. In addition, a vertical line marks the current date.

### Partners and roles

External project participants can be recorded in the **“Partners & roles”** area.

For each partner, the following can be stored:

- name,

- role,
- free note.

The partner list also serves as a reference for assigning responsible persons and executors.

## Document register and file attachments

Project has an integrated document register.

Project-related documents can be recorded with:

- designation,
- note,
- an optional original file.

An actual file can be attached to a document entry, for example:

- PDF,
- image,
- Office document,
- other project-related files.

Several files can be added at the same time. Project automatically creates corresponding document entries.

The files are **exported directly together with the project** and are thus part of the `.jbpr` project file.

Attached files are made available for download after an import. Opening them directly in the browser is deliberately not used, as the file type cannot always be reliably detected after import from the ZIP container.

File and document entry can be handled separately: an attached file can be detached without removing the document entry.

## Status log

Project keeps a continuous additive status log.

The following, among other things, are documented:

- creation of sections,
- creation of project steps,
- exports,
- imports,
- conflict resolutions,

- custom manual log entries.

The entries contain a time stamp and author.

Existing log entries are not subsequently changed or deleted. This creates a chronological project history.

The user name entered at the top of the program is used as the author for new log entries and other relevant changes.

## Data and data exchange

Project uses its own project file format:

**.jbpr**

On export, the entire project including the attached original files is saved in a ZIP container.

The file name follows the pattern:

**YYYYMMDD_HHMMSS_projectname.jbpr**

Internally, the `.jbpr` contains:

- `manifest.json` with the project data and the checksum,
- the `attachments/` directory with the attached original files.

The file can be passed on, for example, via:

- USB storage device,
- e-mail,
- shared file storage.

No online service is required for the processing itself.

## Additive import and merging

A special function of Project is the **additive merging of different project states**.

Via **“Import / Merge”**, a `.jbpr` or a plain JSON manifest file can be read in.

The existing local project state is not simply overwritten.

Instead, the two data states are compared with each other:

- entries present only in the incoming file are adopted,
- identical entries remain unchanged,
- entries edited differently are marked as a conflict.

The status log is also merged additively. Unique IDs prevent duplicate log entries.

Attached files from the imported project file are also adopted.

## Conflict resolution

If contradictory changes are detected, Project displays the local and the incoming version side by side.

Three options are available for each conflict:

**Keep my version**

The local version remains active.

**Adopt incoming version**

The imported version becomes the active state.

**Keep both (flag as possible duplicate)**

Both versions are initially preserved and can then be checked manually.

A discarded version is not completely removed from the project history. It receives the status **“discarded”** and remains traceable with author, time and reason.

The decision on a conflict is made by the person carrying out the import. Project deliberately does not automate a mutual confirmation process between project participants.

## Local operation and significance for JUMBO BLOCK Rescue

Project runs entirely locally in the browser.

It uses:

- no cloud,
- no central server,
- no automatic data transmission.

The current working state is continuously cached in the local browser storage `localStorage`.

The `.jbpr` file is used for permanent backup and for exchange between devices or project participants.

This architecture is particularly relevant for JUMBO BLOCK Rescue.

After a failure of the regular IT infrastructure, an existing `.jbpr` project can be opened and further edited locally. Project structure, dates, responsibilities, status log and embedded original files are available together in a single project file.

This means that not only technical engineering work can be continued, but also organisational project control.

# Data backup

The local browser storage is specific to the device and browser.

For reliable data backup, the project must therefore be exported regularly as `.jbpr`.

Before extensive merges with other project states, the manual expressly recommends a backup export of the current own state.

When a project is reset, Project first asks whether the current state is to be exported. Only then are the project data, sections, steps, partners, documents, attachments and status log cleared.

# Reports

Project provides two different print outputs.

## Project report

Via **“Print / PDF”**, a print-ready report is generated.

This contains:

- project master data,
- sections,
- project steps,
- chronological list of dates,
- partners,
- document register,
- status log,
- SHA-256 checksum.

The report can be printed via the browser or saved as PDF.

## Gantt report

Via **“Print Gantt / PDF”**, the timeline is generated as a standalone print-ready document in landscape format.

Depending on the project size, the paper format is automatically chosen between **DIN A4 and DIN A0 landscape**.

This is intended to ensure that even extensive project plans, including their bar labels, are displayed completely.

# Data integrity

Project continuously calculates a **SHA-256 checksum over the current project state**.

The checksum is:

- displayed in the program,
- included in exported project files,
- output in the print report.

If the content changes, the checksum changes too.

Identical checksums thus identify identical project states.

However, the SHA-256 checksum is expressly **no substitute for version control**. It serves as proof of plausibility and integrity for the data state concerned.

## Limitations and notes

Project has no automatic synchronisation between different devices.

The exchange deliberately takes place via export and import of the `.jbpr` files.

The local `localStorage` is bound to the browser and device. Before changing device or browser, the project must therefore be exported.

When merging two projects with the same file path for an attachment, only the metadata level is checked for conflicts, according to the manual. The actual file contents are not compared byte by byte.

Unclear conflicts between project participants must therefore be resolved organisationally.

Furthermore, Project is not intended exclusively for large projects. The structure can equally be used for small feasibility studies with only a few sections and project steps.

## Technical classification in JUMBO BLOCK Rescue

Project complements the specialist engineering programs with a **locally available project management and documentation level**.

The functional assignment is:

**Hydro / Plan / Budget → specialist project work**

**Static / Temp → internal engineering services with information and handover pages**

**Project → project structure, responsibilities, dates, partners, documents and project history**

The `.jbpr` combines project data and attached original documents in a transportable project file.

The additive import also enables a form of decentralised collaboration without central cloud infrastructure: participants can edit separate project copies and merge the states later in a controlled manner.

Within JUMBO BLOCK Rescue, Project thus supports both the **continuation of a project during an IT failure** and the controlled exchange of project states via local or otherwise available transfer routes.

## Further documentation

JUMBO BLOCK Project – User Manual · Version 1.1.2

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Project” at **https://jumboblock.org/.**

# D.8 – Tender

**Area:** Tendering
**Program:** JUMBO BLOCK Tender
**Baseline 20261001:** 4.0.0 or later

## Purpose

**JUMBO BLOCK Tender** is the local tool of the JUMBO BLOCK Engineering Suite for opening, checking, displaying, creating, editing, pricing and saving BoQs based on **GAEB DA XML**.

The following data phases are supported:

- X80,
- X81,
- X82,
- X83,
- X84.

Tender thus enables a BoQ to be processed from the neutral working basis through specification, cost estimate and invitation to tender up to bid submission.

## Role within the Engineering Suite

Within the Engineering Suite, Tender handles the **tendering and BoQ level**.

It complements the engineering and project management tools with the structured processing of GAEB DA XML data.

The supported data phases have different tasks:

**X80 – Universal BoQ data**
Neutral, price-free working and exchange phase.

**X81 – Specification**
BoQ from planning as the basis for cost estimate and tendering.

**X82 – Cost estimate**
Internal costing with prices of the designer or client.

**X83 – Invitation to tender**
BoQ for the bidder. Internal X82 cost estimates are not carried over.

**X84 – Bid submission**
Return of the bid with bidder prices and permissible additions.

X80 to X83 form the shared structural editing basis. X84 is protected as the bid submission and is not treated as a freely editable BoQ.

# Essential functions

## Opening GAEB X files

Tender can open the following files locally:

- `.x80`,
- `.x81`,
- `.x82`,
- `.x83`,
- `.x84`,
- `.xml`.

Files can be loaded via the file selection or by drag and drop.

When reading in, Tender checks, among other things:

- file extension,
- XML syntax,
- data phase,
- metadata,
- BoQ structure,
- items.

If a file cannot be processed properly, the loaded working copy is discarded and the cause is displayed in the status area.

## New bill of quantities

Via **“New BoQ”**, a new X80 working file can be created.

The following are recorded first:

- project name,
- BoQ name,
- currency,
- first BoQ section,
- first item.

For the first item, quantity, unit, short text and, optionally, a long text can be specified.

The X80 created is then immediately available as a price-free working file.

## BoQ structure

In data phases X80 to X83, the structure of the BoQ can be edited.

BoQ sections can be:

- added,
- edited,
- assigned to permissible parent sections,
- moved within their level,
- deleted.

The structure is based on the hierarchy defined by `BoQBkdn` or `BoQLevel`.

If a new subsection requires an additional hierarchy level, this can be expanded explicitly. Tender does not create intermediate levels silently.

After structural changes, the affected item number/`RNoPart` structures are rebuilt.

## Items

In X80 to X83, items can be added and edited.

The following can be edited:

- quantity,
- unit,
- short text,
- long text,
- assignment to the lowest-level BoQ section.

Items can also be moved within their section or deleted.

After changes, the item numbers are rebuilt according to the new structure or order.

Tender works with a working copy and checks structural changes before adopting them.

## Summary items and subitems

Tender supports the GAEB structure:

**SumDescr / SubDescr**

A broken-down item remains in the BoQ as a summary item. The associated subitems are listed beneath it and numbered with `SubDNo`.

Subitems can be:

- added,

- edited,
- deleted,
- reordered.

In addition, an existing following item can be adopted as a subitem of the immediately preceding item.

Subitems can have quantity, unit, short text, long text and – in price-capable working phases – their own unit price.

The total price of a subitem is calculated as:

**quantity × unit price**

The summary item does not receive an additional unit price of its own. Its total price corresponds to the sum of the total prices of its subitems.

### Search and navigation

Items can be filtered by:

- item number,
- description,
- text.

The BoQ structure can be used to navigate directly to a section.

The table view supports 50, 100 or 250 items per page.

**“Reset view”** removes search and structure filters without deleting the loaded file.

## Prices and totals

The currency is displayed in the header and can be changed in the editable working phases.

In X81, X82 and X83, unit prices can be entered directly in Tender as working values.

For normal items, the following applies:

**total price = quantity × unit price**

For summary items, the total price results from the total prices of the subitems.

From this, Tender calculates:

- item totals,
- section totals,

- BoQ total.

The meaning of the prices depends on the data phase concerned.

X81 prices can serve in particular as working values for generating an X82 cost estimate.

X82 contains internal cost estimates.

X83 prices can be used as working values for the later X84 bid submission.

When saving as X80, X81 or X83, price information is removed. Tender points this out before the export.

## Data phases and phase changes

Tender can generate X80, X81, X82 and X83 from the permissible source phases in each case.

### X80

X80 remains a price-free working and exchange state.

If a priced phase is saved as X80, existing prices are removed after an explicit warning.

### X82

When saving as X82, existing working or cost prices, including subitem prices, are saved.

### X82 → X83

When changing from X82 to X83, the internal cost estimates are not carried over.

Internal client or designer costs thus remain separate from the invitation to tender.

### X84

X84 is not used as a source phase for X81 to X83. However, an existing X84 can be saved back as an X80 working basis.

Files created are saved locally with a time stamp.

## X84 bid submission

Tender can generate an X84 from X80 or X83. An existing X84 can also be saved locally again.

For an X84 from X83, prices must be present. Missing prices can be filled with **0,00** after confirmation.

When generating an X84 directly from X80, Tender points out that X80 is price-free. After explicit confirmation, the prices are initially set to 0,00.

Bidder data is recorded before the export.

Mandatory fields are:

- company / name,
- street and number,
- postal code,
- city.

Optionally, country, contact person and e-mail, among others, can be specified.

This information is written exclusively to the locally generated X84.

Summary items and subitems retain their price-relevant data. The actual structure of the BoQ remains protected in X84 working mode.

## Hidden migration aid for D81–D84

In addition to the regularly visible X80 to X84 functions, Tender also recognises older **GAEB 90 files** with the extensions:

- D81,
- D82,
- D83,
- D84.

This function is deliberately implemented as a **migration aid** and not as a separate GAEB 90 editor.

The assignment is phase-equivalent:

**D81 → X81**

**D82 → X82**

**D83 → X83**

**D84 → X84**

The original D file is not changed. Tender creates a local X working copy.

As DA 1990 is line-oriented and GAEB DA XML is element-oriented, this is expressly not a mere change of file extension. Information can be changed, combined or not transferred during the conversion.

Tender therefore requires a subsequent technical check, in particular of:

- items,
- item numbers,
- structure,
- texts,
- quantities,

- units,
- prices, where applicable.

Records that cannot be assigned in the standard way are not interpreted automatically but are reported and not adopted.

## Local operation and significance for JUMBO BLOCK Rescue

Tender runs completely in the browser.

During editing, the project data remains in the browser's memory.

For project data, Tender uses:

- no server communication,
- no cookies,
- no `LocalStorage`,
- no `SessionStorage`.

Opening and saving GAEB files and PDF generation take place locally.

Tender can therefore be used within JUMBO BLOCK Rescue to open, check and further edit existing BoQs within the supported range of functions, even without available central tendering or cloud infrastructure.

## Data backup

As Tender does not store the project data permanently in the browser, the files created or edited are themselves the authoritative basis for further work.

With regard to the files created, the user is responsible for:

- filing,
- versioning,
- data backup,
- passing them on.

Before extensive changes to a BoQ structure, an **unchanged source file** should be kept.

This applies in particular to changes to the `BoQLevel` levels, as these affect the item number structure of the entire BoQ.

## Printing and PDF

Tender can generate a **DIN A4 PDF report** completely locally.

Optionally, the following can be included in the output:

- long texts,
- table of contents.

Depending on the available data, the report contains:

- project information,
- BoQ information,
- structure,
- items,
- subitems,
- unit prices,
- total prices,
- section totals,
- BoQ total.

For X84, the bidder details can also be output.

## Language and number format

The user interface supports German and English and adopts the language setting from the global menu of the JUMBO BLOCK Engineering Suite.

The language change affects, among other things:

- labels,
- dialogs,
- status messages,
- quantity formats,
- price formats,
- PDF output.

The numerical values used internally remain in the GAEB/XML-compatible data format. The language switch only changes the display and input logic.

## Limitations and notes

Tender 4.0.0 works within the implemented scope of **GAEB DA XML 3.3 / 2021-05**.

Not every optional schema element is part of the user interface.

The internal XML and structure check supports plausibility checking, but does not replace external validation against the authoritative XSD.

For formal data exchange, files created should therefore additionally be validated against the authoritative XSD or with a suitable independent checking tool.

Particular attention is also required when changing between data phases:

**X82 cost estimates and X83/X84 bidder prices are technically different data.**

The corresponding warnings during phase changes should therefore not be skipped.

Tender is developed independently. The name **GAEB** describes exclusively the supported data format and the published technical rules. No certification, commissioning, partnership or other official connection with GAEB is claimed.

## Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Tender complements the Engineering Suite with a locally available tool for **BoQs and GAEB data exchange**.

The functional assignment is:

**Engineering tools → technical project work**

**Project → organisational project management and project documentation**

**Tender → BoQ, tendering, cost estimate and bid data exchange**

During a failure of the regular IT infrastructure, Tender thus makes it possible not only to continue editing technical project documents, but also to open existing GAEB DA XML BoQs locally, edit them structurally, price them, transfer them between supported data phases and document them as PDF.

The completely local processing supports the data sovereignty of the rescue environment.

## Further documentation

JUMBO BLOCK Tender – User Manual · Version 4.0.0

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Tender” at **https://jumboblock.org/.**

# D.9 – Calc

**Area:** Calculation
**Program:** JUMBO BLOCK Calc
**Baseline 20261001:** 2.0.0 or later

## Purpose

**JUMBO BLOCK Calc** is the scientific web calculator of the JUMBO BLOCK Engineering Suite.

The program combines several general and JUMBO BLOCK-specific calculation tools in one local application:

- basic calculator,
- scientific functions,
- percentage calculation,
- geometry calculator,
- JUMBO BLOCK Area Fit,
- unit conversion,
- JUMBO BLOCK Module Calculator,
- calculation history,
- export and import of the calculation history,
- PDF report.

Calc thus serves as a quickly available auxiliary tool for technical calculations, without requiring an external calculation program or an online service.

## Role within the Engineering Suite

Calc is a **standalone calculation tool** of the Engineering Suite.

Unlike the project-specific engineering applications, Calc does not work with the shared `project.dbm`.

There is no direct data exchange with Hydro, Plan, Budget, Static or Temp.

Instead, Calc has its own file format for backing up and restoring its calculation history.

The functional separation is therefore clear:

**Hydro, Plan, Budget and the engineering services → project-specific specialist work**

**Calc → general technical auxiliary calculations**

# Essential functions

## Basic calculator

The basic calculator supports:

- addition,
- subtraction,
- multiplication,
- division,
- brackets,
- decimal numbers,
- sign change.

CE clears the current entry, while C removes the entire expression.

A backspace key allows deletion character by character.

Opening brackets automatically add a multiplication sign where necessary. Missing closing brackets are added automatically when a calculation is carried out.

Every calculation completed with = is added to the calculation history.

## Scientific functions

Calc additionally provides scientific calculation functions:

- π,
- square root,
- square,
- cube,
- general power,
- reciprocal,
- direct percentage conversion,
- scientific power-of-ten input,
- sine,
- cosine,
- tangent,
- arcsine,
- arccosine,

- arctangent.

The trigonometric functions work in **degrees**, not radians.

For invalid value ranges or mathematically undefined situations, Calc refuses the calculation.

### Memory functions

The conventional memory functions are available for intermediate results:

- `MC` – clear memory,
- `MR` – recall memory,
- `M+` – add current value,
- `M−` – subtract current value.

A displayed `M` indicates that a non-zero value is in the memory.

### Percentage calculation

The percentage key of the basic calculator works context-dependently.

With addition or subtraction, a following percentage value refers to the preceding base number.

Example:

**200 + 10 % = 220**

In other contexts, the value entered is divided by 100.

The scientific function `x%`, by contrast, always carries out a direct division by 100, regardless of the preceding operator.

## Geometry calculator

The geometry calculator provides five types of calculation:

- rectangle area,
- cuboid volume,
- circle area,
- cylinder volume,
- triangle area.

Depending on the type of calculation, only the required input fields are activated.

The following base units are available:

- mm,
- cm,

- m.

The result is output accordingly as an area or volume in the selected base unit.

All input values of a calculation must be in the same selected unit. There is no automatic conversion between individual input fields.

## JUMBO BLOCK Area Fit

For rectangular, circular and triangular areas, Calc can additionally determine how many square JUMBO BLOCK modules can be arranged completely within a specified area.

Calc uses the following basic module dimensions:

**2,5 × 2,5 m**

In addition, a **working clearance per edge** can be specified.

The default value is:

**0,50 m**

This surrounding clearance is subtracted from the usable area before the modules are arranged.

For rectangular areas, the number of modules is determined directly from the usable length and width.

For circular and triangular areas, Calc uses a raster-based search procedure with several possible raster offsets and selects the variant with the highest number of modules lying completely within the area.

The following are output as the result:

- module count,
- occupied footprint,
- unoccupied remaining area,
- net volume,
- gross volume,
- total weight.

The calculation is also added to the history.

## Unit conversion

Calc provides 14 fixed technical conversions:

- mm ↔ m,
- cm ↔ m,
- m² ↔ ha,

- l ↔ m³,
- kg ↔ t,
- kN ↔ N,
- MPa ↔ N/mm².

The result is output in the target unit concerned and automatically added to the calculation history.

## JUMBO BLOCK Module Calculator

From an entered module count, the Module Calculator immediately calculates the associated technical parameters of the standard module.

The following are stored:

**Net volume:** 13,9 m³ per module

**Gross volume:** 15,625 m³ per module

**Total weight:** 4,205 t per module

The results are updated while the value is being entered.

With =, the current calculation state is also added to the history.

## Calculation history

Completed calculations are saved in the calculation history with consecutive numbering.

Calculations from the following are recorded:

- basic calculator,
- geometry calculator,
- JUMBO BLOCK Area Fit,
- unit conversion,
- Module Calculator.

In addition to the original expression or inputs, the complete results are saved.

Calculations with several parameters show them as individual detail lines.

Individual numerical results from basic calculations or unit conversions can be transferred back into the basic calculator by clicking on them and used further.

**“Clear history”** removes the entire history immediately and without further confirmation.

## Data and data exchange

Calc does **not** use the shared `project.dbm` of the Engineering Suite.

Instead, the calculation history can be exported to a separate file.

The file name pattern is:

**YYYYMMDD_HHMMSS_calc.dbm**

Despite the file extension, this is expressly a **CALC-specific format** and not an excerpt of the shared `project.dbm`.

Within the file, the data format is identified by:

**"application": "CALC"**

On import, Calc checks whether a valid Calc file is present.

A successful import completely replaces the currently displayed calculation history with the content of the imported file.

According to the manual, **no SHA-256 hash check** takes place for this Calc file format.

## Local operation and significance for JUMBO BLOCK Rescue

Calc runs entirely locally in the browser as a single HTML file.

It requires:

- no cloud,
- no server,
- no user account.

A general technical calculator is thus available within JUMBO BLOCK Rescue even if the regular IT infrastructure fails.

This is particularly relevant for quick auxiliary calculations, unit conversions, geometric estimates and JUMBO BLOCK-related quantity estimates.

Calc deliberately does not replace the specialised engineering applications. It complements them with an independent local calculation tool.

## Data backup

The calculation history is not stored permanently in the browser.

It is held exclusively in the memory of the open page.

An unexported history is lost when:

- the page is reloaded,
- the page is closed.

If a calculation history is to be preserved, it must therefore be saved deliberately as a Calc file via **“Export”**.

Independently of this, the selected language is saved locally in the browser.

## PDF report

Calc can generate a standalone print-ready report from the current calculation history.

The PDF report contains:

- date and time of creation,
- complete numbered calculation history,
- expressions or inputs used,
- all result lines,
- copyright notice.

The report can be printed or saved as PDF via the browser's print dialog.

If the calculation history is empty, the report points this out accordingly.

## Language and operation

Calc supports German and English.

The language is switched via the global menu of the Engineering Suite.

Among other things, the following are adjusted:

- labels,
- decimal separators,
- PDF report.

The existing calculation history is not changed by a language change.

In addition to the on-screen keys, Calc also supports extensive keyboard operation with digits, arithmetic operators, brackets, percentage key, Enter, Backspace, Escape and Delete.

## Limitations and notes

Calc is an auxiliary calculator and no substitute for the project-specific engineering programs.

In particular, there is no data exchange with the shared `project.dbm`.

The JUMBO BLOCK Area Fit for circular and triangular areas uses a **raster-based approximation method** with a limited number of tested raster offsets.

For very large areas, the number of positions tested is reduced for reasons of computing time.

The result is thus an approximation for an axially aligned module arrangement and not an optimisation over arbitrary angles of rotation.

As the calculation history is not stored permanently, it should be exported before the application is closed or reloaded if it will be needed later.

The history is cleared immediately and without confirmation.

### Baseline version note

For the frozen program state, the manual points out a version discrepancy:

The program interface bears **version 2.0.0**. In the exported Calc file and in the PDF report, however, the underlying program state outputs **version 1.7.7**.

According to the manual, this discrepancy exists in the current source code and must be taken into account when comparing version information.

For the JUMBO BLOCK Rescue manual, the following applies in accordance with the frozen baseline:

**JUMBO BLOCK Calc 2.0.0 or later**

## Technical classification in JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Calc provides a **general local technical calculator**.

The functional distinction is:

**Hydro → hydrological preliminary dimensioning**

**Plan → geometric installation planning**

**Budget → cost estimation**

**Static / Temp → internal engineering services**

**Calc → general calculations, geometry, units and quick JUMBO BLOCK quantity estimates**

Calc can thus be used independently of an existing project file and is particularly suitable for spontaneous check, plausibility and auxiliary calculations during local work with JUMBO BLOCK Rescue.

## Further documentation

JUMBO BLOCK Calc – User Manual · Version 2.0.0

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Calc” at **https://jumboblock.org/.**

# D.10 – Units

**Area:** Units
**Program:** JUMBO BLOCK Units
**Baseline 20261001:** 2.0.0 or later

## Purpose

**JUMBO BLOCK Units** is the comprehensive unit converter of the JUMBO BLOCK Engineering Suite for engineering and physics.

The program comprises **26 categories in four groups**:

- Mechanics & Physics,
- Electrical,
- Thermodynamics,
- Other.

Unlike conventional converters, each with one source and one target unit, a value is entered in a starting unit. Units then **immediately displays all units of the selected category converted at the same time**.

This allows technical units to be compared and converted quickly without having to select individual target quantities one after another.

## Role within the Engineering Suite

Units is a **standalone auxiliary tool** of the Engineering Suite.

It does not exchange data with the shared `project.dbm` of the project-specific suite tools.

Instead, the calculation history is exported and imported in a separate, Units-specific file format.

The functional distinction is therefore:

**Hydro, Plan, Budget and the engineering services → project-specific specialist work**

**Calc → general technical calculations**

**Units → physical and technical unit conversion**

## Essential functions

### Categories

The 26 categories are divided into four groups.

### Mechanics & Physics

This group comprises 18 categories:

- Length,
- Area,
- Volume,
- Mass,
- Force,
- Pressure/Stress,
- Energy/Work,
- Power,
- Torque,
- Speed,
- Acceleration,
- Density,
- Angle,
- Time,
- Volumetric Flow,
- Frequency/Rotational Speed,
- Dynamic Viscosity,
- Kinematic Viscosity.

### Electrical

The electrical categories are:

- Electric Voltage,
- Electric Current,
- Electric Resistance,
- Electric Charge.

### Thermodynamics

The Thermodynamics group contains:

- Temperature.

### Other

The following are also available:

- Data Size,

- Illuminance,
- Fuel Consumption.

Depending on the physical quantity, the individual categories contain between 3 and 18 units.

## Performing a conversion

The conversion is determined by three inputs:

- category,
- numerical value,
- starting unit “From”.

Every change automatically triggers a recalculation. A separate calculate button is not required.

For each unit of the selected category, the results list shows:

- unit symbol,
- full name,
- converted value.

The selected starting unit is highlighted in the results list.

When the category is changed, the first matching unit is automatically selected as the new starting unit and the results list is recalculated.

## Number formatting

Units automatically adjusts the display of the results to the order of magnitude concerned.

Very small or very large values are displayed in scientific notation where necessary.

Depending on the order of magnitude, the program uses different numbers of decimal places.

The display is also language-dependent:

**German → decimal comma**

**English → decimal point**

The formatting affects the display and does not change the underlying conversion values.

## Special case: temperature

Temperature values cannot be converted into one another solely by means of a simple linear factor, as the temperature scales have different zero points.

Units therefore uses its own conversion formulas for:

- Celsius,
- Fahrenheit,

- Kelvin,
- Rankine.

This takes into account not only the scaling differences but also the respective zero-point offsets.

## Special case: fuel consumption

Fuel consumption also differs from an ordinary linear unit conversion.

The following are supported:

- litres/100 km,
- kilometres/litre,
- mpg (US),
- mpg (UK).

Some of these quantities are in a **reciprocal relationship** to one another.

Units therefore uses its own conversion formulas.

With a consumption of **0 l/100 km**, conversion into the reciprocal units is mathematically undefined and is displayed as an error.

## Calculation history

The conversion currently displayed can be added deliberately to the calculation history with:

**“Save to history”**

Unlike in Calc, this does **not happen automatically** with every calculation.

A history entry contains:

- category,
- entered value,
- starting unit,
- all converted units.

An existing history entry can be clicked. Units then restores the associated category, starting unit and original value in the main form and recalculates the results.

**“Clear history”** removes the entire calculation history immediately and without further confirmation.

## Data and data exchange

Units does not use the shared `project.dbm` of the Engineering Suite.

Instead, the calculation history can be saved in a separate file.

The file name pattern is:

**YYYYMMDD_HHMMSS_units.dbm**

The file is a standalone **UNITS format**.

Within the file, the file type is identified by:

**"application": "UNITS"**

According to the manual, **no SHA-256 hash check** takes place for this format.

## Import

A previously exported Units file can be imported again.

On import, Units checks each history entry individually.

In particular, the following must apply:

- the category must exist in the current program version,
- the starting unit must exist,
- the entered value must be a valid number,
- at least one converted result must be numerically valid.

Invalid individual entries are skipped on import.

If the file as a whole is not a valid Units file, nothing is adopted.

A successful import **completely replaces the currently displayed calculation history** with the valid content of the imported file.

## Local operation and significance for JUMBO BLOCK Rescue

Units runs entirely locally in the browser as a single HTML file.

It requires:

- no cloud,
- no server,
- no user account,
- no online connection for the conversion factors.

The physical conversion factors used are permanently stored in the tool.

An independent technical unit converter is thus available within JUMBO BLOCK Rescue even when regular engineering workstations or online tools cannot be reached.

Particularly in diagnostic, planning and technical rescue work, values from different systems of units can thus be compared with one another immediately.

## Data backup

The calculation history is not cached permanently in the browser.

It exists exclusively in the memory of the open page.

An unexported history is lost when:

- the page is reloaded,
- the page is closed.

If the history is to be preserved, it must therefore be saved deliberately as a Units file via **“Export”**.

Independently of this, the selected language is saved locally in the browser.

As **“Clear history”** takes effect without confirmation, a history that is needed should be exported beforehand.

## PDF report

Units can generate a standalone print-ready report from the current calculation history.

The PDF report contains:

- date and time of creation,
- complete numbered calculation history,
- category,
- starting value,
- all converted units,
- copyright notice.

Within the report, the browser's print dialog can be opened via **“Print / PDF”**.

If the calculation history is empty, the report contains a corresponding notice instead of the list of calculations.

## Language

Units supports German and English.

The language is switched via the global menu of the Engineering Suite.

Among other things, the following are adjusted:

- labels,

- category names,
- unit names,
- number formatting,
- PDF report.

The existing calculation history is preserved when the language is changed.

The language last selected is saved locally in the browser and used again the next time the program is opened.

## Limitations and notes

Units is a tool for **unit conversion** and no substitute for project-specific engineering calculations.

There is no data exchange with the shared `project.dbm`.

The conversion factors are permanently stored in the program. External live data is not required.

Because of their mathematical properties, special conversion formulas apply to temperature and fuel consumption.

A calculation result is only added to the history if **“Save to history”** is expressly selected. Merely displaying a conversion does not create a history entry.

As the calculation history is not stored permanently in the browser, a working state that is needed must be exported before closing or reloading.

## Technical classification in JUMBO BLOCK Rescue

Units adds a **completely local technical unit converter** to JUMBO BLOCK Rescue.

The functional assignment is:

**Calc → general technical calculations**

**Units → unit conversion for engineering and physics**

The 26 categories cover mechanical, physical, electrical and thermodynamic quantities as well as other technical units.

The principle of displaying all units of a category at the same time provides a quick overview, particularly with unknown or differently documented systems of units.

A general conversion tool that works entirely without external services is thus available even during an IT failure or in an isolated rescue environment.

## Further documentation

JUMBO BLOCK Units – User Manual · Version 2.0.0

Detailed operation and technical details: see the separate JUMBO BLOCK manual “Units” at **https://jumboblock.org/**.

# D.11 – Note

**Area:** Notes
**Program:** JUMBO BLOCK Note
**Baseline 20261001:** 1.0.0 or later

## Purpose

**JUMBO BLOCK Note** is the fast local note-taking application of the JUMBO BLOCK Engineering Suite.

It serves for the free documentation of:

- Thoughts,
- Interim results
- Work instructions,
- technical notes
- References to other programs in the Engineering Suite,
- external resources
- associated files.

Note is not tied to a specific technical field and can be used as a general note-taking and documentation tool.

## Role within the Engineering Suite

**free local documentation layer** to the technical engineering programs.

While Hydro, Plan, Budget and other applications handle structured specialist tasks, Note allows additional information to be recorded independently of these data models.

Note does not exchange data with the shared project.dbm.

References to other suite tools – for example, plan.html – are simply links. They do not create a technical data connection and do not import any data.

Note uses its own file format for backup and exchange.

## Key functions

### Create and edit notes

A blank note is automatically created the first time the file is opened.

Each note consists of:

- Title,

- Note text,
- Links and files.

The title can contain up to **200 characters**.

New notes are inserted at the top of the notes list.

The order is based on the creation date and is not changed by subsequent edits.

### Notes list

The notes list shows:

- Title,
- Date and time of last modification.

If no title was assigned, the following designation appears:

**"Untitled Note"**

The currently open note is visually highlighted.

### Search

The search field filters the notes list immediately.

The following will be searched jointly:

- Title,
- Note text.

The search does not distinguish between uppercase and lowercase letters.

This allows even larger local note collections to be quickly searched for terms.

### Save

Changes to titles, text, links and files are initially treated as unsaved changes.

Clicking **"Save"** permanently saves the current note to the local browser storage.

The time of the change is also updated.

Additionally, the current note can be:

**Ctrl + S**

or on macOS with:

**Cmd + S**

be saved.

## Automatic saving

Note also automatically saves unsaved changes:

- when switching to a different note,
- when creating a new note,
- when leaving or closing the page.

This ensures that changes to the normal workflow are largely saved automatically.

In exceptional situations such as a browser crash, unsaved changes may still be lost.

## Links and files

Each note has its own section called **"Links and Files"**.

Two types of links are supported.

### Left

External or relative links can be included.

Examples include:

- Web addresses,
- relative paths,
- References to other programs in the Engineering Suite.

A link can be given a name and its actual destination.

External links will open in a new browser tab when accessed.

### files

Local files can be directly linked to a note.

The selected file is **fully embedded in the note**.

Technically, this is done as a Base64-encoded data URL.

This ensures that the file remains part of the note even if the original source file is no longer available later.

Embedded files are also retained during export and subsequent import.

The maximum size is:

**10 MB per file**

Larger files will be rejected.

An embedded file can later be downloaded again under its original filename.

Links and files can be removed individually from a note. The change will be permanent upon the next save.

## Data and data exchange

Note: It does not use the common project.dbm of the Engineering Suite.

A separate NOTE format is available for backup and data exchange.

The filename follows the following pattern:

**YYYYMMDD_HHMMSS_note.dbm**

During export, **all existing notes are** saved together, not just the currently open note.

The file contains its own scheme:

**"schema": "JUMBO_BLOCK_NOTE"**

Even embedded files are fully included in this export file.

**, no SHA-256 hash check** is performed for the NOTE file format.

## import

A file previously exported with Note can be imported again.

The program checks whether a valid NOTE file is present.

If the required schema is missing or the file does not contain a valid notes array, the import will be aborted.

Import behavior is important:

**A successful import will replace the entire collection of notes previously stored in the browser.**

It's not simply a matter of adding individual notes.

The process cannot be reversed.

Therefore, before importing, an export of the existing state should be created if the existing notes are to be retained.

## Local working methods and significance for JUMBO BLOCK Rescue

Note runs entirely locally in the browser as a single HTML file.

It requires:

- no cloud
- no server
- No user account.

A unique feature compared to several other standalone tools in the Engineering Suite is the permanent local storage of notes in the browser via:

**localStorage**

Notes are retained even after closing the page or restarting the browser, provided the same browser is used on the same device and its local data is not deleted.

This is particularly useful for JUMBO BLOCK Rescue because local work notes can be taken immediately during a malfunction or recovery.

For example, one can:

- Processing statuses,
- open tasks,
- technical observations,
- File paths,
- Left,
- supplementary documents

Record locally along with the respective notes.

## Data backup

Storage in localStorage is permanent, but **browser- and device-specific**.

A change of:

- browsers,
- device

It does not automatically import existing notes.

Deleting browser data can also make saved notes inaccessible.

Private or incognito browser windows are also not suitable for permanent storage.

For reliable backup, the notes must therefore be regularly:

***_note.dbm**

to be exported.

Exporting is also the intended way to exchange notes between different devices or browsers.

Automatic synchronization does not take place.

## Print and PDF

Clicking **"Print"** automatically saves the current note.

Note then generates a standalone, print-ready report.

This contains:

- Title,

- full note text,
- linked links,
- embedded files with label or filename,
- Date of creation,
- Date of last modification,
- Print date and print time.

The browser print dialog is then opened via **"Print / PDF".**

This allows the note to be printed or saved as a PDF.

## Language

Note supports German and English.

The language can be switched via the global menu of the Engineering Suite.

This will include:

- Labels,
- Placeholder text,
- Print report

switched.

The stored content of the notes themselves is not changed by this.

One special feature of Note is the ability to switch languages across tabs:

If the language is changed in another simultaneously opened browser tab from the same source, Note automatically applies this change without the need to reload the page.

## Limits and Notes

Note is a free, local note-taking tool and is not part of the shared project.dbm data model.

Links to other programs in the Engineering Suite are purely references and do not transmit any data.

Permanent storage via localStorage is tied to the specific browser and device. Therefore, it does not replace external data backup.

An import always replaces the **entire existing notes collection**. Therefore, a backup should be exported before importing a different data set.

Since embedded files are stored completely within the notes and therefore also in the export file, many or larger attachments can significantly increase the size of the note.dbm.

The size limit is 10 MB per embedded file.

The notes list is sorted by creation order, not by the time of last edit. Therefore, the integrated search function can be helpful for older notes that have been edited later.

## Technical classification in JUMBO BLOCK Rescue

**general local note-taking and documentation tool** within JUMBO BLOCK Rescue.

The functional assignment is as follows:

**Project → structured project management and project documentation**

**Note → free notes, interim results, references and supplementary files**

Especially during a rescue operation, this allows information to be documented that does not directly belong in the structured data models of other engineering applications.

By combining local storage, manual export, and embedded files, notes can be kept independently of cloud and server services and, if required, saved or transferred as a standalone .dbm file.

## Further documentation

JUMBO BLOCK Note – User Manual · Version 1.0.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Note” at https://jumboblock.org/**.**

# D.12 – Tables

**Area:** Tables
**Program:** JUMBO BLOCK Tables
**Baseline 20261001:** 2.0.0 or later

## Purpose

**JUMBO BLOCK Tables** is the fully local spreadsheet application of the JUMBO BLOCK Engineering Suite.

The program provides a self-contained workbook with several worksheets and supports, among other things:

- Tables and lists
- Formulas and calculations,
- References between worksheets,
- Cell formatting,
- conditional formatting,
- Sorting, as well as search and replace,
- Bar, line and pie charts,
- Import and export of XLSX,
- Import and export of CSV,
- the native .jbtab format,
- SHA-256 integrity check,
- Print and PDF reports.

Tables is therefore not limited to a specific technical field, but is designed as a general table and evaluation tool.

## Role within the Engineering Suite

Tables complements the specialized engineering applications with a **universal local spreadsheet**.

It can be used for, for example:

- free calculations,
- technical tables,
- Quantity lists
- Evaluations,

- Overviews,
- Documentation lists,
- Diagrams

be used.

Tables does not exchange data with the shared project.dbm of Hydro, Plan, Budget, Static and Temp.

For its own data exchange, the program uses the following instead:

- .jbtab,
- XLSX
- CSV.

This ensures that Tables remains functionally independent of the project-specific data models of the Engineering Suite.

## Workbook and worksheets

On first launch, Tables provides a workbook with three blank worksheets:

- Table 1,
- Table 2,
- Table 3.

Each leaf initially possesses:

- 26 columns from A to Z,
- 100 lines.

Additional worksheets can be created, renamed, deleted, and rearranged by dragging.

Each worksheet manages its own:

- cells,
- Formatting,
- Column widths,
- Line heights,
- Diagrams,
- Rules for conditional formatting,
- Fixations.

Formulas can access cells in other worksheets of the same workbook simultaneously.

When a worksheet is renamed, Tables automatically adjusts existing formula references to that sheet.

## Formulas and calculations

Formulas begin with:

**=**

The following operators are supported, among others:

- Addition,
- Subtraction,
- Multiplication,
- Division,
- Power,
- Percent,
- Text concatenation,
- Comparison operators.

Cell references and cell ranges can be used directly in formulas.

Relative and absolute references are supported, for example:

**A1**

**$A$1**

**A$1**

**$A1**

When copying and filling, relative cell references are adjusted according to their new position.

Tables also recognizes **circular references**, even if they occur across multiple worksheets.

## Formula functions

Tables has its own formula engine with **34 functions**.

These include, among others:

- SUM / SUM,
- AVERAGE
- MIN,
- MAX,
- COUNT

- IF
- AND
- OR
- ROUNDS
- VLOOKUP
- INDEX,
- COMPARISON / MATCH,
- SUMIF
- COUNTIF
- IFERROR
- ABS,
- ROOT / SQRT,
- POTENCY / POWER,
- PI,
- LN,
- LOG,
- EXP,
- SIN,
- COS,
- TAN,
- ARCSIN / ASIN,
- ARCCOS / ACOS,
- ARCTAN / ATAN,
- RADIANS
- DEGREES
- TODAY

The functions can be used with their German or English names, regardless of the chosen interface language.

The trigonometric functions operate in **radians**. For calculations using degrees, corresponding conversion functions are available.

## Error detection

Tables uses seven defined error values:

- #VALUE!
- #DIV/0!
- #NAME?
- #RELATION!
- #Circle!
- #NV
- #NUMBER!

This flags, among other things, incorrect value types, division by zero, invalid cell references, unknown functions, circular references, and mathematically invalid operations.

## Editing and clipboard

Cells and cell regions can:

- selected,
- edited
- copied
- cut out
- inserted
- emptied

become.

When copying, values, formulas, and formatting are retained.

The copied cell values are also available as tab-separated text in the system's clipboard and can therefore be pasted into other programs.

Conversely, tabular text from other applications can be imported as values into tables.

With the following features:

- Fill down,
- Fill to the right

Contents and formulas from a starting row or column can be transferred to a larger area.

Relative cell references migrate accordingly.

## Undo and Redo

Tables supports up to **100** undo and redo steps.

This applies, among other things, to:

- Cell contents,
- Formatting,
- Insert,
- Fill out,
- Sorting
- Substitute,
- Rows and columns,
- Column widths and row heights,
- connected cells,
- conditional formatting,
- Fixations,
- Diagrams,
- worksheets,
- Imports,
- Reset.

The undo history exists only in the memory of the currently open page and starts again empty after a reload.

## Rows and columns

Rows and columns can be inserted and deleted.

Tables automatically updates affected formula references – both:

- within the same worksheet,
- as well as in other worksheets.

Also:

- Cell areas,
- Chart areas,
- Rules of conditional formatting,
- connected cells

will be adjusted accordingly.

References to actually deleted cells become:

**#RELATION!**

## Sort, search and replace

Selected table ranges can be sorted in ascending or descending order.

The entire rows, including their formatting, are reordered together.

The integrated search function operates within the active worksheet.

The search is for cell contents or, in the case of formulas, the formula text.

Hits can be replaced individually or completely.

Search and replace can optionally distinguish between uppercase and lowercase letters.

## Cell formatting

Tables supports, among other things:

- Fat,
- Italics,
- Underlined,
- Text color,
- background color,
- horizontal alignment,
- vertical orientation,
- Frame,
- connected cells.

The following number formats are available:

- Standard,
- Integer
- Decimal with two decimal places,
- Currency €,
- Percent,
- Date.

The formatting only changes the appearance. The actual cell value remains unchanged for calculations.

## Conditional formatting

Cells can be automatically colored with text and background colors depending on their content.

Supported conditions include, but are not limited to:

- larger than,
- less than,
- greater than or equal to
- less than or equal to
- even,
- unequal
- between,
- contains text
- is a mistake.

If multiple rules apply to the same cell, the last rule applied takes precedence.

Conditional formatting is displayed in the print report, but is not exported to an XLSX file.

## Freezing rows and columns

Rows above and columns to the left of a selected cell can be frozen.

This leaves, for example:

- Table headings,
- Label columns

Visible when scrolling.

The settings are saved separately for each worksheet.

It is part of the native .jbtab format, but is not exported to XLSX and is not included in the print report.

## Diagrams

Tables supports three chart types:

- Bar chart
- Line chart,
- Pie chart.

Charts are generated from a selected data range.

They remain connected to their data range and are updated accordingly when the underlying values change.

Diagrams can:

- delay,
- changed in size,
- switched between the supported chart types,
- deleted

become.

When inserting or deleting rows and columns, the underlying chart areas are also updated.

Diagrams are saved and output in the native .jbtab format and in the print report.

When exporting to XLSX, the charts themselves are not transferred; the underlying values are retained.

# Data and data exchange

## Native .jbtab format

The native and lossless storage format for tables is:

**.jbtab**

The filename follows the following pattern:

**YYYYMMDD_HHMMSS_table.jbtab**

The file is a ZIP container with an internal:

**workbook.json**

The complete workbook is saved, including, among other things:

- all worksheets,
- Values,
- Formulas,
- Formatting,
- Diagrams,
- Rules of conditional formatting,
- Fixations.

Additionally, the file contains information about:

- Tool,
- Version,

- Export time,
- SHA-256 checksum.

A successful .jbtab import completely replaces the currently open workbook.

However, the import can be undone using the undo function, as long as the page remains open.

### XLSX

Tables can export XLSX files.

The following will be transmitted:

- all worksheets,
- Cell values,
- Formulas,
- Column widths,
- connected cells,
- Cell formatting including date format.

The formulas are translated into a notation expected by Excel during export.

This allows programs like Microsoft Excel, LibreOffice Calc and similar programs to recalculate the formulas themselves.

When importing XLSX files, the included worksheets are **appended to the existing workbook**.

Supported by:

- .xlsx,
- .xls,
- .xlsm.

If a sheet name already exists, the imported worksheet receives an addition such as:

**“(2)”**

Internal references in the imported file are redirected to the renamed sheets accordingly.

Cannot be transferred via XLSX:

- Diagrams,
- conditional formatting,
- Fixations.

### CSV

CSV import and CSV export only affect the active worksheet.

CSV only transmits values and not:

- Formulas,
- Formatting,
- Diagrams.

The export uses the semicolon as a separator.

During import, Tables automatically recognizes semicolons or commas and creates a new worksheet from the data.

CSV is therefore a simple exchange format, but **not a complete backup format for a Tables workbook**.

## Data integrity

When exporting a .jbtab file, Tables calculates a **SHA-256 hash of the program version and sheet data of the entire workbook**.

Since the program version is also factored into the calculation, the same table content generates a different hash under a different program version.

The checksum is recalculated during import.

Tables distinguishes, among other things:

**“✓ Hash valid”**
The file has remained unchanged since the export.

**“– No hash available”**
The file does not contain a corresponding check value.

**“⚠ Hash discrepancy”**
The content or version no longer matches the state calculated during export.

A file with a hash discrepancy will still be loaded. Additionally, a warning will appear.

After a change to the workbook, a previously checked hash is no longer valid for the current state of the work.

Tables indicates this accordingly.

The SHA-256 key is used in Tables to verify that the workbook has not been modified. No data exchange with the project.dbm file takes place as a result.

## Local working methods and significance for JUMBO BLOCK Rescue

Tables runs entirely locally in the browser as a single HTML file.

It requires:

- no cloud
- no server
- No user account.

Changes are automatically saved to the local browser memory approximately half a second after the last input.

The current state is also saved when closing or reloading.

The next time you open it in the same browser, this work state will be automatically restored.

This means that a general spreadsheet application is available within JUMBO BLOCK Rescue, without the need for an external office suite or cloud service for basic spreadsheet, calculation and evaluation tasks.

## Data backup

Automatic browser saving is enabled:

- the device used
- the browser used

bound.

Therefore, it is not intended as the sole permanent data backup.

For complete and shareable backup, the workbook should be regularly:

**.jbtab**

to be exported.

This format is the native lossless backup format for Tables.

XLSX is suitable for exchanging data with other spreadsheet programs, but it does not transfer all Tables functions.

CSV contains only values from the active worksheet and is therefore not suitable as a complete backup format.

## Print and PDF

Tables generates a formatted report via **"Print / PDF".**

The report contains:

- the selected print area
- the table with actual column widths,
- conditional formatting,
- Diagrams from the worksheet,

- Tool version,
- Time of creation,
- SHA-256 checksum of the workbook.

The workbook is automatically recalculated before the report is generated.

The following paper sizes are available:

- A4
- A3
- A2.

The output can be found in:

- Landscape format
- Portrait format

take place.

Landscape format is the default setting.

Additionally, it can be specified whether guidelines for the table are displayed in the report.

If the workbook has remained unchanged since the last .jbtab export, the report's checksum corresponds to the hash of the exported file. This allows the file and the printout to be associated with the same data state.

## Language

Tables supports German and English via the global menu of the Engineering Suite.

This includes switching between:

- Labels,
- Notes,
- Reports,
- Toolbars,
- Diagram labels,
- Function names in the function menu,
- Print report.

The actual content of the workbook is not changed by switching languages.

German and English function names remain valid regardless of the interface language.

When dealing with decimal numbers within formulas, it is important to note that the notation with a decimal point works in both language versions when switching between languages.

## Limits and Notes

Tables is a standalone local spreadsheet application and does not exchange data with the shared project.dbm.

Different functionalities apply to data backup:

**.jbtab → full native storage format**

**XLSX → Exchange format with formulas and cell formatting, but without charts, conditional formatting and freezes**

**CSV → only values from the active worksheet**

The undo/redo history is only retained while the page is open.

When cutting and then pasting, formulas in other locations that reference the cut cells are not adjusted.

The fill function does not automatically generate number sequences, but rather transfers content or formulas from the first row or column.

Search and replace only works in the active worksheet.

, the field **"Your Name (for log)"** is currently only a notes field. It is not included in the export, the print report, or the SHA-256 checksum.

## Technical classification in JUMBO BLOCK Rescue

Tables extends JUMBO BLOCK Rescue with a **fully local spreadsheet application featuring its own formula engine and native file backup**.

The functional assignment is as follows:

**Calc → quick individual and auxiliary calculations**

**Units → Unit conversion**

**Tables → structured tables, formulas, lists, evaluations and charts**

Thanks to XLSX and CSV support, Tables can also serve as a local exchange tool for existing table data.

the native .jbtab format retains the complete Tables work state, including program-specific functions and SHA-256 integrity information.

This allows tables, technical calculations, lists and evaluations to be created, continued, backed up and documented as PDFs entirely locally during an IT outage.

## Further documentation

JUMBO BLOCK Tables – User Manual · Version 2.0.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Tables” at https://jumboblock.org/**.**

# D.13 – Writer

**Area:** Documents
**Program:** JUMBO BLOCK Writer
**Baseline 20261001:** 10.1.0 or later

## Purpose

**JUMBO BLOCK Writer** is the fully local and offline word processor of the JUMBO BLOCK Engineering Suite.

The program runs as a single HTML file in the browser and requires:

- no server
- no cloud
- no user account
- no network access.

Writer allows you to create and edit formatted documents including tables, images, headings, table of contents, automatic fields and page layout.

Documents can be, among other things, as:

- Writer- .dbm,
- Word file .docx,
- PDF

will be issued.

Word files can now be imported again. Additionally, limited ODT import is available.

This means that Writer provides a standalone local text processor within JUMBO BLOCK Rescue.

## Role within the Engineering Suite

Writer handles **general document creation and text processing** within the Engineering Suite.

It is not bound to the common project.dbm data model of the project-specific engineering applications.

Writer is suitable for, for example:

- technical documentation,
- Reports,
- Business letters,

- File notes,
- Protocols,
- Free project documents.

The functional separation is as follows:

**Project → structured project management and project documentation**

**Note → free notes and additional information**

**Tables → Tables and Reports**

**Writer → formatted documents and word processing**

## Document structure

A Writer document consists of four independently editable areas:

### letterhead

The letterhead appears exclusively on the first page.

For example, it can be used for:

- Sender details,
- Logo,
- Address

be used.

### Main text

The main text forms the continuous body of the document and is automatically distributed across the required pages.

### Footfield

The footer appears exclusively on the last page.

It is suitable, for example, for:

- Greetings,
- Signature areas.

### Side foot

The footer is repeated on every page and can contain information such as company details or dates.

Below this area, Writer additionally generates a separate check line in the PDF with SHA-256 checksum and page number.

The letterhead, footer, and page footer can be shown or hidden. The main text always remains active.

## Text formatting

Writer provides an extensive formatting toolbar.

Support is provided to, among others:

- Fat,
- Italics,
- Underlined,
- Strikethrough,
- Superscript,
- Subscript,
- Text color,
- background color,
- Removing formatting,
- left-aligned
- centered,
- right-aligned
- Justified text
- Enumerations,
- Numbering,
- Left,
- horizontal dividing lines,
- Page breaks.

The following paragraph formats are available:

- Standard,
- Heading 1,
- Heading 2,
- Heading 3.

The font size can be set from **6 to 68 pt**.

**0.8 to 3.0** as well as a default setting are available for the line height.

## fonts

Writer includes twelve locally supplied fonts.

### Sans-serif

- Liberation Sans,
- DejaVu Sans,
- Noto Sans,
- Ubuntu,
- Cantarell.

### Serifs

- Liberation Serif,
- DejaVu Serif,
- Noto Serif,
- Nimbus Roman.

### Monospace

- Liberation Mono,
- DejaVu Sans Mono,
- Noto Sans Mono,
- Ubuntu Mono.

Since the fonts are provided locally, no online connection is required for their use.

## Headings and table of contents

Headings are defined via paragraph formats:

- Heading 1,
- Heading 2,
- Heading 3

defined.

Writer can automatically generate a table of contents from this in the main text.

The table of contents takes the headings into account according to their hierarchical level and contains:

- Heading text,
- Indentation by level,

- dotted line,
- Page number.

New, changed, or removed headings are automatically taken into account.

During PDF export, the page numbers are inserted based on the actual PDF page layout.

Headings from the letterhead, footer and page footer are not included in the table of contents.

## Automatic fields

Writer supports automatic fields for:

- Date,
- Time,
- Page number,
- Total number of pages.

Date and time are updated when the document is loaded.

The page count and total page count receive their final values during output.

In Word export, page numbers are written as Word fields and then calculated by Word itself.

## tables

Writer can insert tables directly into the document.

The number of rows and columns is specified during the creation process.

Then, among other things:

- Lines inserted above or below
- Lines deleted
- Columns inserted on the left or right
- Columns deleted
- Cell contents vertically aligned,
- Cell Limitations shown or hidden
- Complete tables deleted

become.

Column widths can be changed directly by dragging.

Borderless tables can be used, for example, for positioning content in the letterhead.

## Pictures

Image files can be embedded directly into a Writer document.

The image data is embedded as Base64 data within the document. No external image references are created.

Images can:

- proportionally scaled,
- delay,
- centered,
- without text wrap,
- flowed around the left side,
- flowed around the right

be displayed.

Additionally, the vertical alignment to the text line can be set.

## Links and files

Below the actual document, Writer has a separate area:

**"Links and files"**

An entry consists of:

- Designation,
- Goal.

The target can be, for example, a web address or a relative file path.

**"Link file"** function uses the name of a local file as its starting point.

Unlike images, a file linked in this way is **not embedded in the Writer document**. Only the link is saved.

Relative file paths can also be used when using a USB stick.

## Page settings

Writer supports:

**Paper sizes**

- A4
- Letter.

**alignment**

- Portrait format
- Landscape format.

Additionally, the four side edges can be individually adjusted in millimeters.

The default setting is:

- 10 mm at the top,
- left 25 mm,
- 10 mm on the right
- 10 mm at the bottom.

The page layout is calculated based on these settings.

## Templates

Writer includes pre-made document templates.

### business letter

The business letter template is based on DIN 5008 and includes, among other things:

- Company name,
- Return information,
- Address field,
- Information block,
- Sign,
- Contact person,
- Date field,
- Reference,
- Salutation,
- Greeting,
- Signature,
- Company information can be found in the footer.

The exact dimensions of DIN 5008 – for example, for window rebates – must be checked separately before being used accordingly.

### File note

The template includes, among other things, a header table for:

- Date,
- Writer,
- File number,
- Reference,

- Distribution

as well as a prepared outline.

protocol

The protocol template includes, among other things:

- Occasion,
- Date and time,
- Location,
- Participants,
- Minutes taking,
- Agenda,
- Agenda items,
- Action plan including responsibility and deadline.

The templates are generated in the currently selected interface language.

## Search and Replace

Writer has a document-wide search and replace function.

The displayed areas will be searched:

- letterhead,
- Main text,
- Footfield
- Side foot.

Automatic fields and the table of contents are skipped.

The search can optionally distinguish between uppercase and lowercase letters.

Hits can be replaced individually or completely.

## Word and character count

Writer continuously displays:

- Words,
- Characters with spaces,
- Characters without spaces

of the main text.

If text is selected, the word count of the selection is also displayed.

The table of contents is not taken into account in the numbering.

## Data and data exchange

.dbm format for the complete backup of its own document state.

The **"Export Document" function** saves, among other things:

- letterhead,
- Main text,
- Footfield
- side foot,
- Formatting,
- embedded images,
- Table of contents,
- Page settings,
- Activation status of document areas,
- Links and files
- Program version,
- SHA-256 test value.

The Writer .dbm file is a readable JSON format.

Writer therefore uses its own document format and not the common project.dbm of the project-specific engineering applications.

## Import and security sanitization

Writer can fully re-import its own .dbm documents.

During import, the content is additionally cleaned up.

Only the following are retained:

- elements,
- attributes,
- formatting,

that Writer itself supports or generates.

Among other things, the following will be removed:

- scripts,
- event attributes,

- external embedded content,
- external images,
- unsafe links.

If content was removed during cleanup, Writer displays this in the status bar.

This prevents an imported document from incorporating active or unsupported content into Writer without checking it.

## Word export

Writer can save documents as:

**.docx**

export.

Among other things, the following will be transmitted:

- Text formatting,
- Fonts and font sizes,
- Paragraph formatting,
- Headings as Word styles,
- Enumerations,
- Numbering,
- Tables,
- Pictures,
- Left,
- dividing lines,
- Page breaks,
- letterhead,
- side foot,
- Footfield
- automatic fields,
- Table of contents,
- Paper size,
- alignment,
- Page margins.

The SHA-256 checksum is not included in the Word file because a document subsequently edited in Word would no longer correspond to the verified Writer data.

## Word Import

Writer can import .docx files from programs such as Microsoft Word or LibreOffice Writer.

Among other things, the following will be taken over:

- essential text formatting,
- Headings,
- Enumerations and numbering,
- Tables,
- embedded PNG, JPEG and GIF images,
- Web links,
- Page breaks,
- Separation lines,
- Header,
- Footer,
- automatic fields,
- Document title,
- Paper size,
- alignment,
- Page margins.

For documents originally exported from Writer, Writer can restore the footer placed at the end of the main text as a separate document area.

The following are not covered, among other things:

- floating text fields and shapes,
- Footnotes,
- Endnotes,
- Comments,
- Formulas,
- embedded objects,
- linked, non-embedded images,
- multi-column sentence,

- Additional header and footer variations.

Content that was not imported will be reported in the status bar after the import process.

Active or unsafe content will not be included.

## ODT Import

Writer can also import .odt files.

However, only the **pure paragraph text is** transferred to the main text.

Not to be included:

- Formatting,
- Pictures,
- additional Writer document areas,
- Test values.

ODT import is therefore explicitly more restricted than DOCX import.

## Local working methods and significance for JUMBO BLOCK Rescue

Writer runs entirely locally as a single HTML file.

All necessary libraries are embedded in the application.

This is how Writer works:

- without network access,
- directly from a USB stick,
- during a failure of central IT infrastructure.

Changes are automatically saved to the local browser memory approximately half a second after the last input.

When reopened in the same browser, the last saved state will be reloaded.

This means that JUMBO BLOCK Rescue can be used not only for diagnostic, rescue and engineering tasks, but also as a local workstation for creating and maintaining extensive documents.

## Data backup

Automatic browser storage is tied to the specific browser and device used.

For permanent and transferable backup, the document should therefore be exported as a Writer .dbm file.

**"Clear screen"** permanently removes the document version stored in the browser after a security prompt.

Already exported:

- .dbm,
- .docx,
- PDF files

will not be changed by this.

## PDF export

Writer creates the PDF file entirely locally in the browser.

A separate print dialog is not required for this.

During export, Writer distributes the main text across the required pages according to the set page size and margins.

The following apply:

- Letterhead only on the first page,
- Footer only on the last page,
- Footer on each side,
- Manual page breaks are taken into account.
- Long lists and tables are spread across multiple pages.

Page number fields and page numbers in the table of contents receive their actual values during PDF generation.

The pages are included as images in the PDF during export.

Therefore, the text in the generated PDF is **neither searchable nor copyable**.

## Data integrity

Writer uses **SHA-256** to verify the integrity of its own documents.

When exporting a .dbm file, an SHA-256 checksum is generated for the entire document content, including the program version.

The same fingerprint is generated during PDF export.

A check line appears at the bottom of each PDF page:

**left → full SHA-256 value**

**right → side x / y**

This allows a PDF printout to be uniquely assigned to an unchanged Writer .dbm data set.

When importing a Writer .dbm file, the program distinguishes between:

- **valid hash**
- **No hash available**
- **Hash deviation**

The SHA-256 check is used to detect subsequent changes.

It does not explicitly replace **a qualified electronic signature (QES) according to eIDAS**.

## View and operation

Writer can adjust the document view to the available window width.

The zoom can be changed in 10 percent increments.

**100%** represents the actual paper size.

Additionally, a full-screen mode is available.

Keyboard shortcuts exist for important editing functions, including:

- Ctrl + Z → Undo,
- Ctrl + Y → Repeat,
- Ctrl + F → Search.

## Language

Writer supports German and English.

The switch is made via the global menu of the Engineering Suite.

Among other things, the following will be switched:

- Labels,
- Notes,
- Placeholder
- Context tools,
- Status messages,
- Test value reports,
- Table of contents,
- Page break labels.

The actual document content is not automatically translated.

Templates are generated in the language that is set when the respective template is loaded.

## Limits and Notes

Writer is a local word processor with its own document model.

The screen display shows the letterhead, footer, and page footer only once each. The actual repetition or page assignment is only visible in the exported file.

Page number fields only receive their final value during PDF or Word export.

The SHA-256 verification line appears on every PDF page generated by Writer and cannot be disabled.

Imported files are intentionally reduced to the content supported by Writer. Unsupported or potentially unsafe components are removed.

When exporting to Word, images are placed without text wrapping.

Footnotes are not currently supported.

Word import does not include text boxes, footnotes, comments, formulas, or embedded objects.

ODT import is limited to pure paragraph text.

The SHA-256 integrity check verifies the integrity of a writer's data state, but does not constitute an electronic signature.

## Technical classification in JUMBO BLOCK Rescue

Writer extends JUMBO BLOCK Rescue with **fully local, formatted word processing and document creation**.

The functional assignment is as follows:

**Note → quick free notes**

**Tables → Tables, formulas and evaluations**

**Writer → extensive formatted documents, reports, letters and minutes**

Local browser storage and the proprietary .dbm format allow documents to be created and backed up independently of a central office infrastructure.

The DOCX import and export function also allows for exchange with common word processors.

Furthermore, the SHA-256 link between Writer .dbm and PDF output creates a traceable association between the editable document state and the output PDF.

This means that JUMBO BLOCK Rescue can also be used as an independent local workstation for essential documentation and text processing tasks in the event of a failure of the regular IT infrastructure.

# Further documentation

JUMBO BLOCK Writer – User Manual · Version 10.1.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Writer” at https://jumboblock.org/.

# D.14 – Reader

**Area:** Documents
**Program:** JUMBO BLOCK Reader
**Baseline 20261001:** 1.0.4 or later

## Purpose

**JUMBO BLOCK Reader** is the local document and e-book reader of the JUMBO BLOCK Engineering Suite.

It is designed for both technical documentation and publications as well as general e-books.

Supported by:

- EPUB,
- PDF,
- Markdown
- TXT,
- HTML/HTM,
- JSON,
- XML.

Reader runs entirely locally in the browser. Documents are not uploaded to a server, and no user account is required to view the documents.

The application is for:

- Desktop,
- Tablet
- Smartphone

designed.

## Role within the Engineering Suite

Reader takes over the **local document and reading layer within the Engineering Suite**.

The functional distinction is as follows:

**Writer → Create and edit documents**

**Reader → Read and navigate existing documents and e-books**

Reader is therefore not a word processing program and does not modify the opened documents.

Instead, the program provides a unified local reading interface for different document and text formats.

It is particularly suitable for:

- manuals,
- technical documentation,
- PDF documents,
- E-books,
- Markdown documents,
- Text files,
- local HTML documents,
- structured JSON and XML files.

## Supported file formats

### EPUB

EPUB files are displayed as ebooks.

Support is provided to, among others:

- genuine EPUB table of contents,
- Chapter,
- internal links,
- Pictures,
- Bookmark,
- Text highlights.

### PDF

PDF documents are displayed page by page.

Be available:

- Page navigation,
- Zoom,
- Text search.

### Markdown

Supported by:

- .md,

- .markdown.

Markdown documents are displayed in a formatted reading view that takes into account headings, highlights, and simple Markdown structures.

text

TXT files are displayed as continuous scalable text.

HTML/HTM

Local HTML documents are displayed in a **clean reading view**.

JSON and XML

JSON and XML files are represented as structured text.

This function is particularly useful for technical data and exchange files.

## Open document

Above:

**“ Open”**

The operating system's file selection dialog will be opened.

After loading, Reader displays the file name in the header.

Status and reading progress are displayed at the bottom.

Reader stores information locally:

- the last opened file,
- document-related settings,
- Reading positions.

On the next start, Reader will attempt to reopen the last read file and continue from the saved position.

Whether the browser can permanently restore the local file depends on its local storage and security rules.

## Document-related settings

For each document, the following are stored separately, among other things:

- Reading position,
- Font size or PDF zoom,
- Reading width,
- Bookmark,
- Markings.

This allows different documents to be used, each with its own reading settings.

## navigation

The buttons:

◀ / ▶

They are used for navigation.

Their function depends on the type of document that is open.

### EPUB

Switch to the previous or next chapter.

### PDF

Switch to the previous or next page.

### Text formats

Scroll back or forward approximately one screen height.

## Table of contents and sidebar

Above:

☰

The sidebar will open.

### EPUB

**"Contents"** section displays the table of contents stored in the book.

Supported by:

- EPUB 3 navigation documents ( nav.xhtml ),
- EPUB-2-NCX table of contents.

Nested chapters and subchapters are preserved.

If a proper table of contents is missing, Reader first tries to use headings from the chapters.

Only if this is also not possible will a simple chapter list be used.

### PDF

In PDF format, the sidebar displays a page list.

Selecting a page will take you directly to that page.

## Internal and external links

Internal links within an EPUB are resolved within the reader.

This includes, for example:

- Tables of contents within a book page,
- Footnotes,
- Cross-references.

External web links are treated separately.

An internet connection may be required to access them.

## Depiction

Changes to EPUB and text formats:

**A− / A+**

the font size.

The adjustable range extends from:

**8px to 72px**

In PDFs, the same buttons change the zoom instead.

## Reading width

For text documents, the maximum reading width can be selected.

Be available:

- 720,
- 900,
- 1100,
- 1400.

This allows the line length to be adjusted to different screen sizes.

## Light and dark representation

The button:

‹

switches between light and dark display.

This allows the reading view to be adjusted to ambient light and personal preferences.

## Focus mode

Above:

⛶

The focus mode is activated.

This will include:

- head area,
- Suite menu,
- status bar

hidden.

Reader thus uses a larger portion of the available browser space for the actual document.

The focus mode works independently of the browser's fullscreen support.

## Full screen mode

Above:

▣

Reader requests true browser full-screen mode.

If supported by the browser, this may also:

- Browser tabs,
- Address bar

will be hidden.

Actual availability depends on the browser and device used.

## Search

The search field searches the currently open document.

The search is conducted via:

⌕

or the Enter key was pressed.

### PDF

Reader searches the extractable text page by page and jumps to the first page found.

### EPUB and text formats

The search operates within the currently displayed chapter or document content.

A match found will be highlighted via the browser's search function.

### Limits of PDF search

Image-based or scanned PDF files without an embedded text layer cannot be searched like normal text PDFs.

**Reader does not perform OCR text recognition.**

## bookmark

Above:



A bookmark will be set at the current position.

The saved position depends on the document type.

### PDF

The current page will be saved.

### EPUB

The following will be saved:

- Chapter,
- relative reading position.

### Text formats

The relative scroll position is saved.

Existing bookmarks are displayed in the **"Bookmarks" tab** and can be accessed again by clicking on them.

Individual bookmarks can be removed.

## Markings

Above:



Previously selected text can be highlighted in EPUB and text views.

The markings are stored locally on a document-specific basis.

When the chapter in question is displayed again, Reader attempts to restore the highlights.

The sidebar lists existing highlights with a text excerpt.

In the case of multiple identical text passages, the restoration can target a different or the first matching text passage.

Markings should therefore be classified as **personal reading aids and not as revision-proof annotations**.

# Keyboard operation

Reader can be operated largely without a mouse.

As long as the input focus is not in an input field, the following keys, among others, are available:

**← / Image↑**
back

**→ / Page Down / Spacebar**
Next

**Pos1**
to the beginning

**End**
to the end of the document, or to the last EPUB chapter or the last PDF page.

**+ / −**
Change font size or PDF zoom

**F2**
Open file

**F3**
Focus Search

**F4**
opens or closes the sidebar

**F9**
bookmark

**F10**
focus mode

**F11**
Fullscreen mode

**ESC**
to close focus mode or the mobile sidebar and, if necessary, exit browser full-screen mode.

Function keys may already be assigned to other functions depending on the browser or operating system.

# Touch operation

On touch devices, Reader supports horizontal swiping gestures.

A clear swiping motion:

**to the left → continue**

**right → back**

The interface adapts to narrow displays.

The sidebar is displayed there as a drop-down panel.

## Data and data exchange

Reader does not have its own document format for editing the opened files.

The original documents remain the primary data sources.

Reader only stores document-related reading data such as:

- Position,
- Display settings,
- Bookmark,
- Markings

locally in the browser.

This fundamentally distinguishes Reader from Writer, Tables, Note or Project, which create their own working or exchange files.

## Local storage

Reader operates according to an **offline-first principle**.

Document processing takes place locally on the device being used.

The following are among the items stored in the local browser storage:

- Reading positions,
- Display settings,
- Bookmark,
- Markings.

The last opened file is also stored locally – provided this is allowed by the browser – via:

**IndexedDB**

reserved for later reopening.

Deleting:

- Browser data,
- Website data,
- local storage

can remove this information and, if necessary, also the stored last read file.

## Data protection

Reader itself does not upload document content to a server.

The actual document processing takes place locally.

However, a network connection may be required if an open document contains an **external web link** and this link is accessed by the user.

The local nature of the document display must therefore be distinguished from external links within the read content.

## Significance for JUMBO BLOCK Rescue

**local universal document reader** within JUMBO BLOCK Rescue.

This allows, for example, the following during a rescue operation:

- technical manuals,
- Maintenance documentation,
- PDF reports,
- E-books,
- Markdown documentation,
- Text files,
- local HTML documents,
- JSON data,
- XML exchange files

can be opened directly within the Rescue environment.

JSON and XML, in particular, not only allow the reading of classic documents, but also the visual inspection of structured technical files.

Since document processing takes place locally, no cloud or server infrastructure is required for the actual reading.

## Limits and Notes

The display of an EPUB also depends on the quality and structure of the respective EPUB file.

Therefore, e-books with faulty or unusually structured content can lead to limitations.

PDF text search only works with PDF files that have a readable text layer.

The reader **does not have an OCR function**.

External links within EPUB or HTML documents may require an internet connection.

True browser full-screen mode depends on the browser and mobile device. Reader focus mode is available regardless.

Browser cleanup or private sessions may prevent the persistent storage of:

- Reading positions,
- Bookmark,
- Markings,
- last read file

prevent or delete existing data.

Very large PDF or EPUB documents, as well as documents with extensive embedded images, can cause longer loading times depending on the device and available memory.

## Technical classification in JUMBO BLOCK Rescue

Reader complements the local document tools of JUMBO BLOCK Rescue with a **pure reading and navigation application**.

The functional assignment is as follows:

**Writer → Create and edit documents**

**Reader → Read documents, technical documents and e-books**

Reader is therefore particularly suitable for situations where existing documentation must be immediately available during an IT outage, without a full office or cloud infrastructure being available.

Support for EPUB, PDF, Markdown, TXT, HTML, JSON and XML covers both classic documents and technical text and data formats.

Bookmarks, highlights, saved reading positions, as well as focus and full-screen modes further support the use of extensive documentation directly within the Rescue environment.

## Further documentation

JUMBO BLOCK Reader – User Manual · Version 1.0.4

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Reader” at https://jumboblock.org/.

# D.15 – Text

**Area:** Text / Code
**Program:** JUMBO BLOCK Text
**Baseline 20261001:** 1.0.0 or later

## Purpose

**JUMBO BLOCK Text** is the fully local text and code editor of the JUMBO BLOCK Engineering Suite.

The program is designed for editing **plain text files and source code** and is therefore deliberately separated from the formatted text processor JUMBO BLOCK Writer.

Text is particularly suitable for:

- simple text files,
- Source code,
- Configuration files,
- structured text formats,
- technical notes,
- Protocol and log files,
- HTML, CSS and JavaScript files,
- JSON and XML files.

The editing process is entirely local within the browser.

## Role within the Engineering Suite

Text takes on the function of a **general local plain text and code editor**.

The functional distinction is as follows:

**Writer → formatted documents and word processing**

**Note → free notes**

**Reader → Read and navigate documents**

**Text → Open, edit, and save plain text files and source code**

Text does not work with the shared project.dbm of the project-specific engineering applications.

The currently opened file remains the actual working file.

# Key functions

## Open files

Local text files can be opened via the file selection and loaded into the editor.

Text works with various text-based file types and recognizes supported formats based on the file or its extension.

Once opened, the full text content is immediately available for editing.

## New file

In addition to opening existing files, a new empty text file can be created.

This allows you to do this even without an existing source file:

- Notes,
- configurations,
- Source code,
- structured data,
- simple documentation

create.

## Edit

The editor works as a classic plain-text editor.

Content is edited as characters and lines. Unlike Writer, there are no:

- Font formatting,
- Paragraph formats,
- Page layouts,
- embedded images.

This ensures that the actual text content remains immediately visible and editable.

This is particularly important for source code, configuration files, and structured text formats.

# Syntax highlighting

Text supports syntax highlighting for various text-based formats and programming languages.

This allows, for example:

- Keywords
- strings,
- Comments,

- Pay,
- structural elements

They can be visually distinguished from one another.

Syntax highlighting is solely for display purposes in the editor.

It does not change the content of the saved file.

This makes Text suitable for local testing and editing of source and configuration files within the rescue environment.

## Line numbers

The editor provides line numbers.

This allows certain positions to be held in:

- Source code,
- Configuration files,
- Log files,
- technical text files

be clearly referenced.

This makes it easier, in particular, to find errors and to agree on specific text passages.

## Search and Replace

Text has an integrated search function.

This allows terms within the opened file to be found specifically.

Additionally, a replace function is available, which allows existing text passages to be changed.

Especially with extensive:

- Configuration files,
- Source texts,
- JSON/XML files,
- Protocols

This allows certain entries to be located and processed more quickly.

## Undo and Redo

Processing steps can be undone and repeated.

This allows accidental changes to be corrected during ongoing editing, as long as the corresponding editing state is still available in the open session.

## Data and file formats

Text works directly with text-based files.

It does not use its own project format or a shared project.dbm.

This distinguishes text from applications such as:

- Project,
- Note,
- Tables,
- Writer,

who possess their own structured working files or data models.

The text deliberately remains focused on the content of the text file being edited.

This allows, in particular, the editing of files that can also be used with other text editors.

## Local working methods and significance for JUMBO BLOCK Rescue

Text runs entirely locally in the browser.

No cloud or server services are required for the actual text editing.

This means that JUMBO BLOCK Rescue provides a general text and code editor that can be used independently of an installed development environment or an online application.

In the Rescue context, this is particularly relevant for processing or checking:

- Configuration files,
- scripts,
- HTML files,
- CSS files,
- JavaScript files,
- JSON files,
- XML files,
- Log files,
- other plain text files.

Text thus complements both the technical rescue tools and the engineering suite with an immediately available local editing option for text-based data.

## Data backup

Since Text works with the actual text files, an edited version must be deliberately saved as a file.

Before changes to important:

- System files,
- Configuration files,
- Source texts,
- Project files in readable text format

As a general rule, an unaltered backup copy of the initial state should be created.

This is particularly important within a rescue environment.

A small syntactic change to configuration or program files can have significant effects on the associated system.

## Language

Text is integrated into the German/English language user interface of the JUMBO BLOCK Engineering Suite.

The language switch affects the user interface.

The content of the opened text file is not changed or automatically translated.

This allows files in other languages or programming languages to be edited independently of the selected interface language.

## Limits and Notes

Text is a **plain text and code editor**.

It is not a formatted word processor.

For documents containing:

- Page layout,
- Font formatting,
- Table layout,
- embedded images,
- letterhead,
- Table of contents

JUMBO BLOCK Writer is planned.

Text is also not a complete integrated development environment.

Syntax highlighting improves the readability of source code, but does not perform any technical analysis of the program code.

Special care is required when making changes to system and configuration files.

Successfully saving a file does not automatically mean that its contents are:

- syntactically correct
- technically correct
- compatible with the target system

is.

Therefore, the original data state should be backed up before any changes are made to important files.

## Technical classification in JUMBO BLOCK Rescue

Text adds a **general local editor for plain text and code files to JUMBO BLOCK Rescue**.

The functional assignment is as follows:

**Note → quick free notes**

**Writer → formatted documents**

**Reader → Read documents**

**Text → Edit text, configuration and source code files**

This means that, in addition to its diagnostic, data recovery, engineering and document tools, JUMBO BLOCK Rescue also offers an immediately available way to directly examine and, if necessary, edit text-based files.

This function is particularly relevant for system rescue and recovery because numerous technical details and configurations are available in plain text format.

The completely local workflow enables this processing even when no regular development, office, or cloud infrastructure is available.

## Further documentation

JUMBO BLOCK Text – User Manual · Version 1.0.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Text” on https://jumboblock.org/**.**

# D.16 – PDF Merge

**Area:** PDF
**Program:** JUMBO BLOCK PDF Merge
**Baseline 20261001:** 1.0.0 or later

## Purpose

**JUMBO BLOCK PDF Merge** is a deliberately small and simple tool within the JUMBO BLOCK Engineering Suite for:

- Merging multiple PDF files into a new PDF file,
- Changing the page order,
- Removing individual pages from a PDF file.

Both core functions – **merging and removing pages** – are based on a common, ordered page list.

If multiple PDF files are loaded, their pages will appear together in this list. If only one PDF file is loaded and individual pages are subsequently removed, a reduced version of the original file will be created when saving.

The processing is done entirely locally.

## Role within the Engineering Suite

PDF Merge performs a clearly defined task within the Engineering Suite:

**Select existing PDF pages, rearrange them and merge them into a new PDF file.**

The program does not edit the actual content of a PDF page.

The functional distinction is as follows:

**Writer → Create documents and output them as PDFs**

**Reader → Read existing PDF documents**

**PDF Merge → Select, sort, and merge existing PDF pages**

**PDF Printer → Print documents or supported content as PDFs**

PDF Merge does not work with the shared project.dbm of the project-specific engineering applications.

## Add PDFs

PDF files can be added in two ways.

### File selection

Above:

**“ Add PDF(s)”**

The operating system's file selection dialog will open.

Multiple selections are possible.

### Drag-and-drop

PDF files can be dragged directly onto the designated storage area.

Clicking on this area also opens the file selection.

The storage area can also be operated via keyboard:

- Tab key for focusing,
- Press Enter or spacebar to open the file selection.

## Loading the PDF files

Each selected file will be loaded individually.

Meanwhile, PDF Merge displays the respective file name in the status.

Is a file:

- No valid PDF
- encrypted

She will be rejected.

Other valid files selected at the same time will be processed independently.

This means that a single unsuitable file will not cause the entire loading process to be aborted.

## Shared page list

Each successfully loaded PDF page appears as a separate line in the page list.

The following will be displayed:

- Source file name,
- original page number,
- Total number of pages in the source file.

The presentation follows, for example, the following scheme:

**"File name – Page X of Y"**

The pages of different source files are distinguished by color on the left edge.

PDF Merge uses a palette of six colors for this. If more than six PDF files are loaded, this color palette is repeated.

Above the page list, a summary is also displayed showing how many pages were loaded from how many files.

The crucial point is:

**The order of this list corresponds exactly to the later page order of the generated PDF file.**

## Arrange pages

Several options are available for each page.

### Move one position

With:

▲

The page will be moved up one position.

With:

▼

It will be moved down one position.

The function that is not possible at the top or bottom of the list is deactivated.

### Free movement

A page can also be dragged directly to a different position using the mouse.

While dragging, a line indicates whether the page will be inserted above or below the respective target line.

This allows even more extensive page sequences to be rebuilt immediately.

## Remove pages

Above:

×

The respective page will be removed from the worklist.

Only the entry in PDF Merge will be removed.

**The original PDF file will not be changed.**

In this way, for example, individual unnecessary pages can be removed from a PDF by subsequently saving the remaining pages as a new PDF file.

## Save

When adding pages for the first time, PDF Merge automatically generates a suggested filename according to the following pattern:

**yyyymmdd_hhmmss_pdfmerge.pdf**

Example:

**20260811_093000_pdfmerge.pdf**

The file name can be freely changed before saving.

Missing file extension:

**.pdf**

It will be added automatically.

Above:

**“ Save”**

PDF Merge creates a new PDF file from the current page list.

The pages are copied exactly in the order shown.

If the page list is empty, no file will be created.

## Storage location and browser differences

The behavior during saving depends partly on the browser used.

### Chrome and Edge

In Chrome and Edge, the native operating system dialog can be used:

**"Save as"**

be opened.

There you can:

- Target folder,
- File name

be elected.

The filename previously entered in PDF Merge is already suggested.

If the dialog is canceled, PDF Merge will not create a file.

### Firefox and Safari

Firefox and Safari generally do not offer this free folder selection for the respective browser storage method.

PDF Merge indicates this in these browsers.

The generated file is then saved under the specified filename in the browser's default download folder.

This limitation arises from the respective browser and not from the PDF processing of PDF Merge.

## Data integrity

After each successful save, PDF Merge calculates the following:

**SHA-256 checksum**

the newly generated PDF file.

**16 characters** of the SHA-256 value are displayed in the status bar.

This allows you to later check whether an existing file exactly matches the data state that PDF Merge created at the time of saving.

The hash refers to the generated PDF file itself.

## Reset

Above:

**“♻ Reset”**

The current work status can be completely cleared.

This will include:

- Page list,
- loaded source files in memory,
- File name field

reset.

If files or pages have already been loaded, a security prompt will appear beforehand.

Only after their confirmation will the current work status be discarded.

The original PDF files on the data carrier are not changed as a result.

## Local working methods and significance for JUMBO BLOCK Rescue

PDF Merge runs entirely locally in the browser.

There are:

- no cloud processing,
- no upload to a server,
- No user account.

All PDF files are processed on the device on which PDF Merge is run.

The new PDF file is also generated locally.

This allows, for example, within JUMBO BLOCK Rescue:

- Saved PDF documents merged,
- Reports arranged in the desired order
- Individual unwanted pages removed
- several technical documents combined into one common file

This can be done without having to transfer the documents to an external service.

## Meeting-based working methods

PDF Merge does **not save the current state of work between sessions**.

Each new session begins with an empty page list.

After:

- Reloading the page,
- Closing the browser tab

The loaded files and the established page order are no longer present.

Therefore, any required result must be saved as a new PDF file before ending the session.

PDF Merge does not have its own project or session file format for later resuming a merge that has already been started.

## Language

PDF Merge supports German and English.

The switchover is done via the shared menu of the JUMBO BLOCK Engineering Suite.

This will include:

- Labels,
- Status messages,
- Explanatory texts

switched accordingly.

The downloaded PDF content itself is of course not changed as a result.

## Limits and Notes

PDF Merge is intentionally limited to **arranging and selecting existing PDF pages**.

### No page preview

The pages are displayed as labeled lines.

No thumbnails of the individual PDF pages are generated.

This keeps the program deliberately small and eliminates the need for an additional extensive library for image preview.

### No editing of page content

PDF Merge cannot modify content within a page.

In particular, no functions are available for:

- Turning pages,
- Inserting watermarks,
- Editing text,
- Editing images or other page content.

Only the following will be changed:

- Selection of pages,
- Order of the pages.

### No encrypted PDF files

Password-protected or encrypted PDF files will be rejected upon loading.

### No continuous session

Loaded files and page order are not saved between sessions.

### Browser-dependent storage location

The ability to freely choose the destination folder depends on the browser used.

In Firefox and Safari, the file is saved via the browser's standard download mechanism.

## Technical classification in JUMBO BLOCK Rescue

PDF Merge complements JUMBO BLOCK Rescue with a **small, fully local tool for structurally reorganizing existing PDF documents**.

The functional assignment is as follows:

**Reader → Read PDF files**

**Writer → Create documents and output PDFs**

**PDF Merge → Merge, sort, and remove PDF pages**

**PDF Printer → Convert content to a PDF output**

Local processing is particularly relevant in rescue operations because recovered or confidential documents do not need to be transferred to an external online service for simple PDF merging.

The original files remain unchanged. A new PDF file is only created from the current page list when saving.

The subsequently calculated SHA-256 value also allows verification that the generated output file remains unchanged.

## Further documentation

JUMBO BLOCK PDF Merge – User Manual · Version 1.0.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “PDF Merge” at https://jumboblock.org/.

# D.17 – PDF Printer

**Area:** PDF
**Program:** JUMBO BLOCK PDF Printer
**Baseline 20261001:** 1.2.1 or later

## Purpose

**JUMBO BLOCK PDF Printer** is a browser-based, locally operating tool for:

- Advertisement,
- Test,
- Print preparation
- direct output

Existing PDF documents via the connected system printer.

The PDF file is processed entirely locally in the browser. Documents are not transferred to an external server.

In addition to normal page output, PDF Printer particularly supports:

- different paper sizes and orientations,
- multiple PDF pages on one printed sheet,
- Brochure printing,
- different rendering qualities,
- an interactive page preview.

PDF Printer thus serves as a specialized local printing interface for PDF documents.

## Role within the Engineering Suite

PDF Printer handles the **local print preparation of existing PDF files within the Engineering Suite**.

The functional distinction is as follows:

**Writer → Create documents and output them as PDFs**

**Reader → Read PDF documents**

**PDF Merge → Merge, sort, and remove existing PDF pages**

**PDF Printer → Check existing PDF documents, prepare them for printing and output them via the system printer**

PDF Printer does not change the actual content of the source document.

It does not work with the shared project.dbm of the project-specific engineering applications.

## Open PDF

A PDF file can be opened via the local file selection.

Alternatively, PDF Printer supports:

**Drag & Drop**

directly into the preview area.

The processing then takes place within the browser.

## Interactive preview

Once opened, the PDF is displayed in an interactive preview.

A sidebar contains thumbnails of each PDF page.

This allows the document to be visually checked and navigated in a targeted manner before the actual printing.

The active side can be selected via:

- the buttons ◀ / ▶,
- a click on the corresponding thumbnail.

The preview feature clearly distinguishes PDF Printer from the deliberately simplified PDF Merge, which does not generate page thumbnails.

## Zoom and page resizing

Several zoom functions are available for document preview.

### Zoom in and out

Above:

**+**

or rather:

**–**

The display can be enlarged or reduced.

### 100%

The function:

**100%**

resets the display to the corresponding zoom level.

### Width

**"Width"**

The PDF page is adjusted to fit the available width of the preview area.

### Page

**"Page"**

displays the complete PDF page within the available preview area.

### Continuous zoom

Additionally, you can use:

**Ctrl + mouse wheel**

The preview can be zoomed continuously.

## Paper format

According to the manual, the following paper sizes are available for printing:

- A4
- A3
- Letter.

This allows the target paper to be set independently of the original PDF page format for print preparation.

## Page orientation

The following options are available for the selected target document:

- Portrait format
- Landscape format

be elected.

Paper format and orientation together form the basis for subsequent print preparation.

## Print quality

PDF Printer provides three quality levels for rendering.

### Draft

Fast rendering at lower resolution.

This setting is particularly suitable for quick test prints.

### Good

Standard rendering for general documents.

### High

Maximum rendering quality.

This setting is especially useful for:

- Plans,
- Pictures,
- Presentations

planned.

The choice of quality thus influences the preparation of the PDF pages for printing.

## Pages per sheet – N-up

PDF Printer can arrange multiple PDF pages on a single printed sheet.

Supported by:

- 1 page per sheet,
- 2 pages per sheet,
- 4 pages per sheet.

This so-called **N-up output** allows, for example:

- space-saving control printouts,
- compact documentation,
- Reduced-size multi-page output.

The pages are arranged on the target paper according to the selected setting.

## Brochure printing

One special feature of PDF Printer is:

**“Print as a brochure”**

In booklet mode, the program automatically rearranges the pages so that:

- double-sided printing
- Fold,
- Staples

a finished booklet can be created.

The page order required for brochure printing is generated automatically.

When brochure mode is activated, the normal N-up selection is deactivated.

# Duplex note

PDF Printer supports preparation for double-sided printing by providing appropriate instructions.

Especially when printing brochures, the program provides recommendations for the required duplex setting.

One example is:

**Binding on the short edge**

for corresponding booklet output.

The actual duplex function is performed by the connected printer or its system print dialog.

PDF Printer prepares the page layout and indicates the appropriate print settings.

# Print output

The final output is via the connected **system printer**.

The options actually provided by the respective printer can be used via the system's or browser's print dialog.

Depending on:

- operating system,
- browser,
- printer driver,
- printer model

Other settings are included.

PDF Printer itself focuses on the prior preparation and arrangement of the PDF pages.

# Keyboard operation

PDF Printer supports multiple keyboard shortcuts.

## Page navigation

**Left arrow**
previous page

**Right arrow**
next page

## zoom

**+** or **=**
zoom in

**-**
zoom out

**Ctrl + mouse wheel**
stepless zoom in the preview area

Adjust to width

**F**

adjusts the display to the available width.

100%

**0**

Resets the zoom to 100%.

Press

**Ctrl + P**

opens the print dialog immediately.

This allows PDF Printer to be operated largely via the keyboard, even for frequently recurring checking and printing tasks.

## Data and data exchange

PDF Printer does not use its own project or document format.

The currently opened PDF file remains the original document.

The program serves the purpose of:

- Advertisement,
- Test,
- Print preparation,
- output

this file.

There is no data exchange with the shared project.dbm of the Engineering Suite.

The underlying manual also does not document a custom PDF printer project format.

## Local working methods and significance for JUMBO BLOCK Rescue

PDF Printer processes the opened PDF documents entirely locally in the browser.

The document is not uploaded or transferred to an external server.

This allows, for example, within JUMBO BLOCK Rescue:

- technical reports,
- rescued PDF documents,
- Plans,
- manuals,
- Project documents,
- Documentation

locally checked and prepared for an existing printer.

Especially for confidential documents or documents rescued from a damaged system, this eliminates the need to use an external online PDF service for print preparation.

## Importance of the system printer

PDF Printer is not a separate printer driver.

The final output is done via the print function of the system used.

Therefore, the actual possible physical output also depends on whether the following is possible within the rescue environment:

- the printer is recognized
- a suitable printer driver or appropriate system support is available,
- The desired duplex or paper function is supported by the printer.

PDF Printer handles the browser-side document display and print preparation; the physical print output then takes place via the existing system printing environment.

## Data protection

Local processing is a key feature of PDF Printer.

The opened PDF file is not transferred to an external server for document display.

This also allows confidential documents to be prepared for printing within the local Rescue environment.

This distinguishes PDF Printer in particular from web-based printing or PDF services, which first have to upload documents to an external server.

## Technical basis

JUMBO BLOCK uses PDF Printer for PDF display:

**PDF.js**

the Mozilla Foundation.

PDF.js is located under the following:

**Apache License 2.0**

Their own:

- HTML components,
- CSS components,
- JavaScript program logic

The JUMBO BLOCK PDF Printer is from the ZANNI GROUP.

## Limits and Notes

PDF Printer is a tool for **viewing and preparing existing PDF documents for printing**.

It is not a PDF content editor.

The actual page content is not edited.

**JUMBO BLOCK PDF Merge** is designed for merging, removing, or rearranging individual PDF pages.

The actual print output depends on the existing system printing environment.

In particular, functions such as:

- Duplex printing
- supported paper sizes,
- Print resolution,
- Printer trays,
- Edge behavior

This may depend on the printer used and its system support.

The booklet mode prepares the page order. For a correct physical booklet, the appropriate duplex, folding, and, if applicable, binding requirements of the respective printing system must also be observed.

The functions documented in the underlying manual refer to version 1.2.0. The frozen JUMBO BLOCK Rescue Baseline 20261001 contains version 1.2.1. Changes between these two versions are not assumed here unless they are verified by the available program documentation.

## Technical classification in JUMBO BLOCK Rescue

PDF Printer adds a **fully local PDF preview and specialized print preparation to JUMBO BLOCK Rescue**.

The functional assignment is as follows:

**Reader → Read PDF documents**

**PDF Merge → Merge, sort, and remove PDF pages**

**PDF Printer → Prepare and print PDF documents for physical output**

With:

- Page preview,
- Thumbnails,
- Zoom,
- A4/A3/Letter,
- Portrait and landscape format,
- three rendering qualities,
- 1/2/4 pages per sheet,
- Brochure printing

PDF Printer covers the essential functions for local print preparation documented in the manual.

This allows existing PDF documents to be checked within the rescue environment and prepared for an available system printer even during an IT outage, without transferring the documents to external services.

## Further documentation

JUMBO BLOCK PDF Printer – User Manual · Available Documentation Version 1.2.0

JUMBO BLOCK Rescue Baseline 20261001: PDF Printer 1.2.1 or later

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “PDF Printer” at https://jumboblock.org/.

# D.18 – Convert

**Area:** Conversion
**Program:** JUMBO BLOCK Convert
**Baseline 20261001:** 0.3.2 or later

## Purpose

**JUMBO BLOCK Convert** is the local conversion tool of the JUMBO BLOCK Engineering Suite for the following purpose:

**PDF → Word (.docx)**

Many programs in the Engineering Suite deliberately generate their reports exclusively as PDFs. Convert fills the gap when the content of such a PDF document needs to be **further edited in Microsoft Word or a compatible word processor**.

The goal of the conversion is explicitly **not to create a pixel-perfect, optical 1:1 copy** of the original PDF document.

The priority instead is:

**A Word document that is as editable as possible and has a cleanly structured flow of text.**

Minor visual deviations from the PDF template are deliberately accepted.

## Role within the Engineering Suite

Convert forms a bridge between the PDF outputs of the Engineering Suite and subsequent editable Word processing.

The functional distinction is as follows:

**Writer → Create and edit documents**

**Reader → Read existing documents**

**PDF Merge → Merge and rearrange PDF pages**

**PDF Printer → Prepare PDF documents for printing**

**Convert → Transfer existing PDF content into an editable Word document**

Convert does not work with the shared project.dbm of the project-specific engineering applications.

A separate project or work format is also not provided for.

The input file is an existing PDF file; the result is a new .docx file.

## operation

The operation is deliberately reduced to three steps.

### 1. Select PDF

Above:

**“Select PDF”**

The operating system's local file selection will open.

Only PDF files are accepted.

### 2. Convert

After selecting a valid PDF file:

**"Convert"**

active.

One click starts the complete conversion.

There are:

- no document preview,
- No additional conversion settings.

The processing is automatic.

### 3. Save the Word file

After successful conversion, the generated .docx file is saved or output via the browser's download mechanism.

## Status display

Below the buttons, a status text provides information about the current state.

Possible messages include:

- **No PDF selected**
- **Being converted…**
- **Ready**
- **Cancelled**
- **Conversion failed**

This makes it immediately apparent whether:

- no file has been selected yet
- The processing is underway.

- the conversion was successfully completed,
- the saving process was aborted
- An error occurred during processing.

## How the conversion works

When the conversion starts, the PDF file goes through several automatic processing steps.

### Text extraction

For each PDF page, the available text characters are included, along with information about:

- Position,
- Font,
- font size

read out.

This text layer forms the basis for the subsequent reconstruction of the Word document.

## Header and footer recognition

Convert attempts to automatically detect recurring headers and footers.

Lines of text are treated as corresponding Word headers or footers if they:

- recur identically on at least two pages in the upper and lower margin areas

or

- correspond to a typical page numbering pattern.

One example is:

**“Page 3”**

Detected headers and footers are not simply incorporated into the normal body text, but are created as actual Word headers and footers.

## Sales reconstruction

PDF files often store text in a line- or position-oriented manner.

Simply copying each PDF line as a separate Word paragraph would therefore lead to highly fragmented documents.

Instead, Convert attempts to match related lines of text based on:

- Line spacing,
- Move-in,
- Correspondence

to reconstruct coherent paragraphs.

The goal is an **editable flowing text** that can then be meaningfully edited further in Word.

## Table recognition

Convert attempts to automatically recognize tabular structures.

The table lists sequences of at least:

**three consecutive lines**

It is recognized when their column positions match.

A recognized structure is built as a **real, editable Word table**.

This ensures that suitable tabular PDF content is not merely preserved as visually aligned text.

## Pictures

Embedded raster graphics can be extracted from the PDF file and then embedded into the Word document.

This includes, for example:

- Photos,
- Logos,
- Graphics other than pixel images.

PDF vector graphics that are stored as paths and not as raster images are not supported.

## Font assignment

PDF font names are mapped to the most suitable standard Word fonts during conversion.

Examples from the manual are:

**Helvetica → Arial**

**Times → Times New Roman**

For unknown or embedded fonts, Convert uses the following as a fallback solution:

**Calibri**

The font mapping supports the goal of creating a Word document that is subsequently easy to edit.

However, the goal is not to reproduce every font used in the PDF exactly.

## Data and data exchange

Convert has a deliberately simple data flow:

**PDF file → Convert → DOCX file**

There are:

- no proprietary Convert project format,
- no exchange with project.dbm
- No documented intermediate file.

The PDF file remains unchanged.

The result of the conversion is saved as a new Word file.

The proposed filename corresponds to the original name of the PDF file, where:

**.pdf**

through:

**.docx**

will be replaced.

## Choose storage location

The storage behavior depends on the browser used.

### Chrome and Edge

If the browser supports the **File System Access API**, the following can be done after the conversion of the native:

**"Save As" dialog**

of the operating system.

There you can:

- Target folder,
- File name

freely chosen.

### Firefox and Safari

If the browser does not support this function, the generated Word file will be automatically saved via the browser's normal download mechanism.

The file will then be placed in the corresponding download folder.

In this case, too, no additional file transfer takes place.

## Local working methods and significance for JUMBO BLOCK Rescue

Convert runs entirely locally in the browser.

There are:

- no cloud connection,
- no user account
- no cookies,
- No transfer of the PDF file to a server.

The processing takes place exclusively in the working memory of the open browser.

This also allows:

- confidential documents,
- technical reports,
- PDF files recovered from a damaged system,
- locally generated engineering reports

be converted into an editable Word document without transferring the original file to an external conversion service.

This is particularly relevant for use within JUMBO BLOCK Rescue.

## No continuous work session

Convert is not a document editor and does not have its own persistent work session.

The selected PDF file is processed and then output as a new .docx file.

There is no documented convert work state that needs to be saved separately and reopened later.

The processing takes place in the browser's working memory.

## Language

Convert supports German and English.

The language can be switched via the global menu of the JUMBO BLOCK Engineering Suite.

This will include:

- Labels,
- Status messages

switched between German and English.

The chosen interface language has **no influence on the content of the generated Word document**.

Convert does not translate the PDF content.

# Limits and Notes

## No pixel-perfect 1:1 conversion

Convert explicitly does not aim to reproduce the PDF template with visual accuracy.

The priority is the **editability of the result in Word**.

Minor layout deviations are therefore part of the chosen conversion approach.

## Multi-column layouts

Multi-column page layouts – for example, a newspaper layout with several adjacent text columns – are not recognized separately.

This can cause the text in the generated Word document to appear in an unexpected reading order.

## Short tables

Automatic table recognition requires at least three consecutive, column-oriented rows.

Tables containing only:

- one line
- two lines

are not reconstructed as a true Word table.

Instead, they will be displayed as normal body text with visible spaces.

## Headers and footers

Headers and footers are only automatically recognized if a line:

- It recurs identically on at least two pages.

or

- corresponds to a page number pattern.

Marginal text that is unique or changes from page to page – for example, chapter headings – deliberately remains part of the normal running text.

## vector graphics

Vector graphics stored as paths are not converted.

Only embedded raster graphics are supported.

### No OCR

Scanned PDF files without an embedded text layer cannot be converted into editable text using Convert.

Convert does **not have OCR text recognition**.

The conversion is based on the existing text layer of the PDF file.

A pure image PDF does not provide this basis.

## Technical classification in JUMBO BLOCK Rescue

**fully local PDF-to-Word converter** to JUMBO BLOCK Rescue.

The functional assignment is as follows:

**Reader → Read PDF documents**

**PDF Merge → Structurally merge PDF pages**

**PDF Printer → Prepare PDF documents for printing**

**Convert → Reconstruct PDF content for subsequent Word editing**

Convert takes a deliberately different approach than an optical PDF copy:

**Priority is not given to pixel-perfect reproduction, but rather to the subsequent editability.**

Through:

- Text extraction,
- Header/footer detection,
- Sales reconstruction,
- Table recognition,
- Embedded raster images are imported,
- Mapping PDF fonts to Word fonts

The aim is to rebuild a practically editable document from the static PDF structure.

Fully local processing is particularly important in rescue operations because confidential or recovered documents do not need to be transferred to an external service for conversion.

## Further documentation

JUMBO BLOCK Convert – User Manual · Version 0.3.2

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Convert” at https://jumboblock.org/**.**

# D.19 – Archive

**Area:** Archive
**Program:** JUMBO BLOCK Archive
**Baseline 20261001:** 1.1.1 or later

## Purpose

**JUMBO BLOCK Archive** is the fully local program of the JUMBO BLOCK Engineering Suite for **packing and unpacking files and folders**.

The interface is intentionally kept simple. Files, folders, or archives can be dragged into the program; Archives automatically detects whether:

- Existing files or folders were compressed

or

- unpacked an existing archive

are to be.

For the actual processing:

- No files uploaded,
- no user account
- No internet connection after loading the application

needed.

Archive is designed for desktop, tablet and smartphone.

## Role within the Engineering Suite

Archive handles the **local archiving, packaging, and unpacking functions within the Engineering Suite**.

It serves in particular to:

- to combine multiple files
- to package complete folder structures,
- to distribute compressed files,
- To protect ZIP files with AES-256,
- to examine and unpack existing archives,
- to convert certain archive formats to ZIP.

Archive does not work with the shared project.dbm of the project-specific engineering applications.

The selected files and archives remain the actual data sources.

## Supported formats

**Unzip:** ZIP (also with password), 7z, RAR, TAR, TAR.GZ, TAR.XZ, TAR.BZ2, GZ
**Pack:** ZIP (also with password), TAR.GZ, TAR

RAR and 7z archives can only be **extracted**, not created.

## Technical basis of the archive formats

The processing of:

- ZIP,
- TAR,
- GZIP

is proprietary program code of the JUMBO BLOCK Engineering Suite.

For:

- 7z,
- RARE
- TAR.XZ,
- TAR.BZ2

Archive uses the freely licensed library:

**libarchive**

under BSD license.

The relevant library is only loaded when such an archive is opened.

Is the file missing?

vendor/libarchive/libarchive-wasm.js

function:

- ZIP,
- TAR,
- GZIP

furthermore.

A corresponding message will then appear for formats that depend on libarchive.

## Packing files and folders

There are three ways to create a new archive:

- **"Compress files"**
- **“Pack the folders”**
- Drag and drop files or folders into the window.

When packing a folder, the existing folder structure is preserved.

**Empty folders** are also included.

## File list

Before the actual packing process, Archive displays the selected files and folders, including their size.

Above:

×

An entry can be removed again.

With:

**“+ Add files”**

and:

**“+ Add folder”**

Additional content can be added.

Additional files and folders can also be added at any time via drag and drop.

## Archive name

The field:

**“Name of the archive”**

The appropriate options are automatically pre-selected.

When packing a folder, for example, its name can be used as a starting point.

The suggested archive name can be changed before saving.

In addition, the desired archive format is selected.

For general exchange, the manual recommends:

**ZIP**

## compression

When creating ZIP files, Archive uses the compression method:

**Deflate**

Compression is performed by the browser.

File types that are already compressed will not be compressed again if no further meaningful size reduction is expected.

This includes, for example:

- Photos,
- Videos,
- Music,
- Archives,
- Office documents.

Under:

**"Expanded"**

Compression can be completely disabled in ZIP files.

This makes the packing process faster, but the resulting archive file becomes larger.

## File names and timestamps

File names are formatted as follows:

**UTF-8**

saved.

This means that, for example, German umlauts are preserved.

When packing, the following are also taken into account:

- Date of change,
- time

the files were transferred.

## ZIP64

If a ZIP archive exceeds:

- 4 GB

or does it contain more than:

- 65,535 entries

Archives are used automatically:

**ZIP64**

This also allows the creation of ZIP archives that exceed the classic ZIP limits.

## Password protection with AES-256

With ZIP files:

**“🔒 Protect with password”**

be activated.

The password must be entered twice.

Above:

**"Show password"**

It can be made visible for inspection purposes.

Encryption is done using:

**AES-256 using the WinZip method.**

The manual recommends a password with at least:

**12 characters.**

## Compatibility of encrypted ZIP files

The AES-encrypted ZIP files generated by Archive can be opened with programs including:

- JUMBO BLOCK Archive,
- 7-Zip,
- WinZip,
- WinRAR,
- Keka,
- The Unarchiver.

The ZIP folder integrated into Windows cannot open AES-encrypted ZIP files.

## Visible metadata in ZIP encryption

AES encryption protects the **contents of the files**.

As is typical with ZIP, however, the following remain unaffected:

- File names,
- Folder structure,
- File sizes

Visible without a password.

Confidential information should therefore not already be included in file names or folder names.

# Handling passwords

Archive does not save the passwords used.

Without the correct password, the encrypted content cannot be recovered.

The password should therefore be stored separately from the archive.

The manual gives the following as a possible example:

**JUMBO BLOCK Vault**

Furthermore, the archive and password should not be shared via the same transmission method.

# Open and check the archive

An existing archive can be accessed via:

**"Unpack archive"**

can be selected or dragged directly into the window.

After loading, Archive shows:

- Archive format,
- Number of files,
- unpacked total size,
- Full table of contents.

This allows the contents of an archive to be checked before the actual unpacking.

# Unpack everything

Above:

**"Unpack everything"**

The complete archive content will be output.

## Chrome and Edge

In Chrome and Edge, a destination folder can be selected.

Archive creates a new folder with the name of the archive and restores the contained folder structure.

If a folder with the same name already exists, for example:

**“Name (2)”**

used.

Existing files will not be overwritten.

### Firefox and Safari

In browsers without corresponding folder access, the files are output individually to the download folder.

Subfolders are identified in the file name by:

**"_"**

hinted at.

Archive will ask you beforehand if you have a larger number of files.

Additionally, the browser may request permission for multiple downloads.

## Target folder and browser protection

In browsers with folder access, the folder selection starts in the Downloads folder.

For security reasons, certain sensitive areas are not allowed by the browser as direct extraction destinations.

The manual mentions, for example:

- the download folder itself,
- Documents,
- Desktop,
- User directory,
- System drive such as C:\.

In such a case, a subfolder can be selected or created within the desired area.

USB sticks and other drives can be selected directly as the destination.

If no destination folder is selected, Archive offers:

- **"Select folder again"**

or

- **"Download instead"**

to.

## Extract individual files

Next to each file within an open archive is the following:

**"Save"**

available.

This allows only this single file to be extracted.

If an archive contains only one file, this file will be:

**"Unpack everything"**

saved directly.

## Convert archive to ZIP

Above:

**"Save as ZIP"**

can:

- 7z,
- RARE
- TAR

be converted into a new ZIP file.

This allows a widely used ZIP archive to be created from an archive format that may require additional software on a target system.

## Integrity check during unpacking

Archive checks each file during unpacking based on the check information contained in the archive.

Be used:

- CRC-32

or in the case of AES-protected ZIP files:

- an authenticity code.

If damage to the archive is detected, Archive displays a corresponding message.

This check is distinct from an external cryptographic verification of origin. It serves to detect damage or faulty archive data.

## Safe pathway treatment

During unpacking, Archives cleans up the paths contained in the archive.

No archive entry may be written outside the selected target folder.

This prevents manipulated path information from leaving the intended unpacking area.

Additionally:

- In Windows, invalid characters are replaced with _.
- symbolic links,
- Special entries

skipped.

These protective measures are particularly relevant when opening external archives within a rescue environment.

## Open password-protected archives

If Archive detects a password-protected ZIP file, a corresponding message will appear.

After entering the password and selecting:

**"Further"**

The password will be checked immediately.

A message will appear if the password is incorrect.

Encrypted ZIP files are supported with:

- AES-128,
- AES-192,
- AES-256,

as well as the older ZIP method:

**ZipCrypto**

## Limits of encrypted 7z and RAR archives

Encrypted:

- 7z,
- Rare

cannot be unpacked with JUMBO BLOCK Archive.

The program indicates this in this case.

The manual refers to suitable external programs for this purpose, such as:

- 7-Zip,
- WinRAR.

## Progress and Cancel

For longer processes, Archive displays:

- Progress in percent
- Currently edited file.

Above:

**"Cancel"**

or:

**Esc**

The current process can be terminated.

Important when unpacking:

Files already written to a target folder will remain there even after the process is aborted.

Therefore, canceling the process does not automatically restore the target folder to its previous state.

## Local working methods and significance for JUMBO BLOCK Rescue

Archive processes all files exclusively on the device used.

Neither will:

- File contents,
- File names,
- Passwords

transferred or stored externally.

This provides a general local tool within JUMBO BLOCK Rescue to, for example:

- to combine the recovered files,
- To back up folder structures,
- To package data for sharing,
- to archive sensitive data in encrypted form,
- to examine foreign archives
- To unpack archive data onto a USB drive,
- To convert RAR, 7z or TAR contents into a generally usable ZIP archive.

No internet connection is required for these tasks after the application has loaded.

## Importance for recovered data

Archive can be used to transfer already identified and secured data into a transportable archive file.

The general rescue principle still applies:

**Archiving is neither a substitute for data recovery nor for trust verification.**

Packaging a file into:

- ZIP,
- TAR.GZ,

- TAR

This does not change the technical or safety-related assessment of its content.

Therefore, files that have been recovered or originated from a compromised system should be checked accordingly before further use.

## RAM and archive size

The amount of data that can actually be processed depends on the available RAM of the device.

### ZIP, TAR and GZIP

These formats are largely processed piecemeal.

Depending on the device used, data volumes of several gigabytes are also possible.

### 7z, RAR, TAR.XZ and TAR.BZ2

These archives are fully loaded into memory.

The manual therefore recommends archive sizes of approximately the following for these formats:

**1 to 2 GB.**

For larger archives, available RAM can become a limiting factor.

## Change data during unpacking

At:

- 7z,
- RARE
- TAR.XZ,
- TAR.BZ2

The original modification data of the files will not be retained.

The extracted files will instead receive the current date.

This is particularly important to consider in forensic or documentary workflows where file timestamps may be relevant.

## Unsupported archive variants

Not supported:

- individual .xz files without a TAR archive,
- individual .bz2 files without a TAR archive,
- multi-part archives.

The handbook gives the following examples of multi-part archives:

- .001,
- .z01,
- .part1.rar.

## Rare ZIP compression procedures

Rarely used compression methods within ZIP can be processed via libarchive, provided the archive in question is not encrypted.

The manual mentions, for example:

- Deflate64,
- BZIP2,
- LZMA within ZIP.

## Mobile use

Archive is also designed for mobile devices.

On smartphones, the large buttons are arranged one below the other.

Packing entire folders is technically not possible on certain mobile devices – for example, an iPhone.

In this case, the corresponding button will be hidden.

## Language

Archive supports German and English.

The language can be switched via the shared menu of the JUMBO BLOCK Engineering Suite.

The selected language changes the user interface, but not:

- File contents,
- Archive names,
- Archive structure.

## Limits and safety information

Archive is a tool for packing and unpacking files.

It does not perform a content-related security assessment of the archived files.

A successfully extracted archive therefore does not mean that its contents:

- trustworthy,

- malware-free
- technically correct

is.

In particular, archives from unknown or untrusted sources should initially be handled in a controlled manner in the rescue context and, if necessary, checked with the analysis and security tools provided for this purpose.

AES-256 protects the file contents of a correspondingly created ZIP archive, but not its visible metadata such as file names and folder structure.

The password used is not saved and cannot be recovered from archives.

Aborting the unpacking process does not automatically remove files that have already been written.

For important data, an additional data backup should be available, regardless of the archiving program.

## Technical classification in JUMBO BLOCK Rescue

Archive complements JUMBO BLOCK Rescue with a **fully local tool for archiving, compression, encryption and unpacking**.

The core functions include:

**Unzip:** ZIP (also with password), 7z, RAR, TAR, TAR.GZ, TAR.XZ, TAR.BZ2, GZ
**Pack:** ZIP (also with password), TAR.GZ, TAR

Especially for rescue operations, the protective mechanisms during unpacking are also relevant:

- Path cleanup,
- No writing outside the selected destination folder.
- Skipping symbolic links and special entries,
- Examination of the archive data,
- No automatic overwriting of existing files when outputting folders.

Archive thus serves both the practical packaging of rescued data and the controlled local opening of existing archive files.

The processing remains on the rescue system or the data carrier used and does not require an external cloud infrastructure.

## Further documentation

JUMBO BLOCK Archive – User Manual · Version 1.1.1

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Archive” at https://jumboblock.org/.

# D.20 – Files

**Area:** Files
**Program:** JUMBO BLOCK Files
**Baseline 20261001:** 1.0.0 or later

## Purpose

**JUMBO BLOCK Files** is the local file manager of the JUMBO BLOCK Engineering Suite.

With it, you can:

- local folders,
- USB sticks,
- external drives

browse and manage.

Files enables, in particular:

- View files and folders
- Open files in a preview window,
- search by file name
- Search text and code files for content,
- Create files and folders,
- rename,
- copy,
- move,
- delete,
- Download files.

Files runs entirely locally in the browser and also works when the Engineering Suite is started directly from a USB stick.

No files are being uploaded.

The application does not contain any third-party program libraries.

## Role within the Engineering Suite

Within the Engineering Suite, Files handles **general local file management**.

While the other applications process specific document or data formats, Files provides basic access to the underlying file and folder structure.

The functional distinction is, for example:

**Files → Organize files and folders**

**Archive → Pack files and folders or unpack archives**

**Reader → Read supported documents**

**Text → Edit text and code files**

**Writer → Create and edit formatted documents**

Files does not use the shared project.dbm as its own project format. For Files, a project.dbm is simply a file within the file system.

Direct transfer to other programs in the Engineering Suite via an **"Open with" function** is not included in version 1.0.0.

## Browser and access rights

For security reasons, web browsers are not allowed to independently access arbitrary areas of a data carrier.

Therefore, a folder must first be created via:

**“ Open folder”**

selected and made available for files.

Files can then work within this folder and its subfolders.

### Chrome, Edge and Opera on the computer

These browsers offer the full range of functions.

When you open a folder, the browser asks whether changes should be allowed.

If this permission is not granted, the folder in question can only be read.

### Firefox, Safari and most smartphones

According to the manual, only read access is available here.

Folders can:

- searched
- Files viewed in preview,
- Files downloaded

become.

Changes to the file system are not possible in this mode.

Files indicates this at startup.

## Reading and writing / Reading only

The current access mode is displayed in the upper right corner of Files:

✎ **Reading and writing**

or:

👁 **Read only**

Functions that are not available in the respective mode are automatically grayed out.

This makes it immediately clear whether Files is allowed to make changes within the currently open folder.

## Protected areas

The browser protects certain system areas from direct access.

System folders such as:

C:\Windows

Programs

AppData

will be completely blocked.

Also:

- Documents,
- Downloads,
- Desktop,
- User directory,
- System drive such as C:\

cannot be opened directly.

In this case, the browser may, for example, report that the selected folder contains system files.

Subfolders within these areas, however, can be opened.

If necessary, a new subfolder can first be created in the selection window.

**USB sticks and external drives can be opened directly.**

These browser-related access restrictions are important for classifying Files as a rescue file manager: Files does not replace a file manager with direct operating system or root rights.

## Open folder

Above:

**“ Open folder”**

A folder is selected.

Alternatively, a folder can be dragged to the left sidebar.

Under:

**"Open folders"**

Multiple folders can be included simultaneously.

For example, the following can be done simultaneously:

- a USB stick,
- a folder on an internal storage device

be ready.

The directory tree can be accessed via:

▶

and:

▼

fold up and down.

Clicking on a folder opens it.

Above:

×

A folder will simply be removed from the list of open folders.

**The folder itself is not deleted.**

## Recently opened folders

Files remembers up to:

**eight most recently opened folders.**

Clicking on such an entry will reopen the folder.

The browser may ask for the necessary permission again.

For example, if a previously used USB stick is no longer connected, Files displays a corresponding message.

**"Recently Opened"** list is stored in the browser of the respective computer.

It will **not be saved on the USB stick**.

## navigation

The path bar above the file list shows the current storage location.

Each part of the path can be clicked.

With:

⬆

or the:

**Backspace**

The user will be moved to the parent folder.

With:

↻

or rather:

**F5**

The current display will be updated.

## File list

The list view shows:

- Name,
- Size,
- Type,
- Date of change.

Clicking on a column header sorts the sort by the corresponding attribute.

Another click reverses the sort order.

**Folders are always at the top, regardless of the sorting order.**

## List and tile view

Above:

▦

The screen switches to tile view.

Thumbnail images are displayed for photos.

Above:

☰

The system returns to the list view.

## Select files and folders

A simple click selects an element.

The usual procedures are available for multiple selections:

**Ctrl + Click**
to add or remove individual elements from the selection

**Shift + Click**
to select contiguous area

**Ctrl + A**
select all elements

On touch devices, the first tap selects an item.

A second tap opens it.

## Open files and folders

A:

**Double-click**

or:

**Enter key**

opens a folder or a file preview.

## File preview

Files can display various file types directly in the browser.

According to the manual, the following are supported:

- Pictures,
- PDF,
- Audio,
- Video,
- Text files,
- Code files.

For text files, the first:

**megabyte**

displayed.

With:

**◀ / ▶**

The arrow keys can be used to switch to the previous or next file in the preview.

Above:

**"Download"**

A copy of the displayed file can be saved.

**Esc**

closes the preview.

## Status bar and folder size

The status bar shows:

- Number of entries,
- Size of entries,
- Number and size of the current selection.

Double-clicking this information calculates the total size of the:

- selected folder

or rather:

- currently open folder

including all subfolders.

## Create a new folder

Above:

**“ + Folder”**

A new folder will be created.

Files suggests a name that can then be changed.

## Create new text file

Above:

**“ + Text file”**

A new empty text file will be created.

Here too, a file name is suggested and can be customized.

## Rename

Above:

**“✏ Rename”**

or rather:

**F2**

The selected element will be renamed.

The following symbols are not permitted:

\ /: * ? " < > |

Also not permitted are:

- a point at the end,
- a space at the end

of the name.

## Copy

To copy, select the desired elements and then:

**“⧉ Copy”**

or rather:

**Ctrl + C**

used.

The target folder will then open.

It may also be located in another open folder or on a different storage device.

With:

**“♢ Insert”**

or rather:

**Ctrl + V**

The elements are copied to the target.

## Move

The following is used for moving:

**“✂ Cut out”**

or rather:

**Ctrl + X**

Then the target folder is opened and via:

**“♢ Insert”**

or rather:

**Ctrl + V**

inserted.

Cut-out elements are displayed in a pale color until they are pasted.

The clipboard only works **within Files**.

## Copying between storage media

Since multiple folders can be opened simultaneously, files can be moved between, for example:

- internal storage device
- USB stick,
- external drive

Copy or move, provided the browser has the necessary access rights for both areas.

This is one of the central functions of Files within JUMBO BLOCK Rescue.

## Name conflicts

If a file is copied to the same folder, Files automatically creates a copy with the following suffix:

**"- Copy"**

If an element with the same name already exists in the target, the following options are available:

- **Substitute**
- **Both retain**
- **Skip**
- **Cancel**

At:

**"Keep both"**

A number is appended to the name.

With:

**"Take over for all others"**

The same decision can be applied to the remaining conflicts in the ongoing process.

When a folder is replaced, its contents are merged.

## Protection against recursive copying

A folder cannot:

- within itself
- in one of its own subfolders

be copied.

Files recognizes this case and prevents the process with a corresponding warning.

## Drag-and-drop from the outside

Files copies files and folders that are dragged onto the file list from another file manager, for example, into the currently open folder.

The manual cites the following as an example:

**Windows Explorer.**

## Progress and Cancel

Longer file operations display a progress dialog.

The ongoing process can be tracked via:

**"Cancel"**

to be completed.

Elements that have already been completely copied will be retained.

A file that is in the process of being copied at the time of cancellation will not be left as a partially written file.

## Delete

Above:

**“🗑 Delete”**

or rather:

**Remove**

The current selection will be deleted.

A security prompt appears beforehand.

Especially important:

**Files does not have a recycle bin.**

Deleted files and folders will be **permanently removed**.

If a folder is deleted, this applies to all of its contents.

For this reason, the following is included in the security check:

**"Cancel"**

Provided as a highlighted default setting.

## Seek

The search function searches:

**the current folder and all subfolders.**

Various search methods are available.

### Search by name

Part of the name is sufficient.

Capitalization is irrelevant.

Placeholders are also supported:

*****
any number of characters

**?**
exactly one sign

Examples:

*.pdf

Report_20??.docx

## Search by type

Above:

**"All types"**

The search can be restricted to specific categories.

Be available:

- Folder,
- Pictures,
- Documents,
- Audio,
- Video,
- Archives,
- Code/Text.

Type filtering can also be used without an additional search term.

## Search within text content

With:

**"also in the content of text files"**

Additionally, text and code files are searched for their content.

Files up to a size of: will be searched.

**10 MB.**

The manual mentions, for example:

- TXT,
- CSV,
- HTML,
- JSON,
- XML,
- Markdown
- Source code.

Content cannot be searched:

- Office documents,
- PDF files.

## Search results

The hit list also shows the location of each file found.

The maximum number of items displayed is:

**5,000 hits.**

A double-click opens a result.

With:

**"End search"**

The file view will be returned to normal.

A search in progress can be cancelled.

## Download files

Above:

⬇

The selected files will be saved as a copy in the browser's download folder.

Folders cannot be downloaded directly.

If you want to output entire folders as a single file, the manual refers to:

**JUMBO BLOCK Archive**

This means that files and archives complement each other perfectly:

**Files → Manage files and folders**

**Archive → Pack complete folder structures into archive files**

## Booting from USB stick

When the JUMBO BLOCK Engineering Suite is started from a USB stick, Files continues to work with folders:

- on the computer,
- on the USB stick itself.

All that's needed on a computer is a suitable browser such as:

- Chrome,
- Edge

to be available.

No file installation is required.

Sharing a folder applies to the respective browser on the respective computer.

If the same rescue stick is used on a different computer, the desired folders must be reselected and shared there.

## Local working methods and data protection

All files are processed exclusively on the device being used.

Neither will:

- File contents,
- File names

transmitted.

Only the following are stored locally in the browser:

- View and sorting settings,
- the list **"Recently opened"**.

This means that the actual file processing remains local.

## Significance for JUMBO BLOCK Rescue

Within JUMBO BLOCK Rescue, Files provides an additional browser-based interface for common file operations.

Particularly relevant are:

- Access to USB sticks and external drives,
- opening multiple folders simultaneously,
- Copying between different storage locations,
- Preview of important file types,
- Search for files,
- Content search in text and code files,
- Determining folder sizes,
- Local processing without file transfer.

This allows, for example, previously recovered data to be:

- checked and sighted
- wanted,
- sorted.
- copied to another data carrier
- prepared for further processing

become.

Files itself is **not a data recovery program**.

The actual recovery of damaged or deleted data remains the responsibility of the rescue tools provided for this purpose in Block A.

## File timestamps and attributes

When copying, the resulting copies retain:

**the current date.**

File attributes and access rights are not transferred.

This is particularly important to consider when the original source is:

- Timestamp
- File attributes,
- Access rights

relevant for documentation, analysis or forensic examination.

Files is therefore not intended for a legally valid 1:1 backup.

## Move

The behavior during the moving process depends on the browser and the storage device.

Within a drive, moving files occurs directly if the browser supports this.

Otherwise:

1. the element is copied,
2. The original was then deleted.

This distinction must be taken into account when dealing with large amounts of data and during rescue operations.

## Performance limits

The working speed depends on, among other things:

- Data storage devices,
- browser,
- Number of files.

Very large folders containing many thousands of files can therefore:

- Open,
- Seek,
- Copy

This will require more time.

## Locked files

Files that are currently being used by another program may be locked by the operating system or browser.

Files reports such a condition accordingly.

## Language and presentation

The surface supports:

- German,
- English.

The language selection is made via:

**DE / EN**

in the common menu of the Engineering Suite.

Smartphones can be used via:

☰

The folder bar can be shown or hidden.

## Limits and safety information

Files operates within the browser's security boundaries.

It does not possess direct system or root privileges.

Certain protected areas therefore cannot be opened.

The delete function is particularly important to note:

**There is no recycle bin. Deleted files and folders are permanently removed.**

Before any deletion operation, it must therefore be clearly checked whether:

- the correct data carrier
- the correct folder
- the correct files

are selected.

The general rescue principle remains paramount:

**Preserve original and damaged sources as unchanged as possible.**

Therefore, Files should preferably be used for already backed-up data or working copies during data recovery operations when write operations are required.

Copying does not fully preserve original file timestamps, attributes, and access rights.

Direct transfer to other applications of the Engineering Suite via **"Open with"** is not available in version 1.0.0.

Important data should be backed up separately from the file manager.

## Important keyboard shortcuts

**Backspace / Up arrow**
to parent folder

**F5 / ↻**
Update view

Double-click / EnterOpen folder or file preview

Ctrl + ClickAdd or remove individual selection

Shift + ClickSelect range

Ctrl + ASelect all

F2Rename

Ctrl + CCopy

Ctrl + XCut

Ctrl + VPaste

DeleteDelete permanently – after security prompt

Up / Down arrowMove selection within the list

Left / Right arrowSwitch to the previous or next file in the preview

EscClose preview or dialog, or clear selection

## Technical classification in JUMBO BLOCK Rescue

Within the Engineering Suite, Files forms the **general local file management layer**.

The functional assignment is as follows:

**Files → View, search, copy, move and manage files and folders**

**Archive → Pack files and folders or unpack archives**

**Reader → View documents**

**Text → Edit text and code files**

For rescue operations, the combination with the tools from Block A is particularly relevant:

**Rescue tool → Identify and recover data**

**Files → View, search and organize recovered data**

**Archive → package or encrypt selected data as needed**

Files extends JUMBO BLOCK Rescue with a fully local file manager that can also be used directly from the Engineering Suite started from the USB stick.

The crucial safety limit remains clear:

**Files manages existing files. It does not replace sector-based data backup, data recovery, integrity, or forensic tools.**

## Further documentation

JUMBO BLOCK Files – User Manual · Version 1.0.0

Detailed operating instructions and technical details: see separate JUMBO BLOCK manual “Files” at https://jumboblock.org/**.**

# The journey continues

This manual ends with **JUMBO BLOCK Rescue – Baseline 20261001.**

The development behind this doesn't end here.

What began with the question of how we can better protect infrastructure from water has long since led us to a much bigger question:

**How do we maintain our ability to act when familiar systems no longer work?**

There is no single answer to that.

Sometimes a retention space is needed beneath a sealed surface.

Sometimes, robust engineering.

Sometimes local documentation that can be accessed even without the cloud.

And sometimes you need a USB stick from which an independent system can boot, while the actual computer refuses to start.

JUMBO BLOCK Rescue is therefore not an end point.

It is another building block in our understanding of what **resilient infrastructure** can be.

Baseline 20261001 documents the current state: four tool blocks, 70 programs and functions, and a fully local engineering environment in case normal technical practice suddenly becomes anything but normal.

We don't know what will be necessary tomorrow.

But that's precisely what resilience is about.

It's not possible to predict every disruption.

Rather, it's about having structures, tools, and knowledge available to react to things no one has foreseen.

Therefore, we will continue to develop.

Continue testing.

Keep asking questions.

And we will probably continue to build things that initially prompt people to ask us:

**"Why would a company specializing in rainwater retention do something like this?"**

Our answer to that is now quite simple:

**Because infrastructure belongs together.**

In an emergency, water, buildings, energy, communication, engineering, data and IT do not exist in separate drawers.

They must work together.

And if one of them fails, someone has to be able to carry on.

In this sense, the journey is far from over.

**Block by block.**

**JUMBO BLOCK**
*Resilience does not end at the structure.*

Warmest regards,

your JUMBO BLOCK team

Imprint:

**Development**

**JUMBO BLOCK**
ZANNI GROUP
Brauhof 12
44866 Bochum

**Telephone**: +49 (0) 2327 4178 191
**Fax**: +49 (0) 2327 4178 192
**E-mail**: mail@jumboblock.de

Registry Court: Bochum District Court
Commercial registration number: HRA 7687
VAT identification number according to § 27a of the German VAT Act: DE340292357

**Distribution**

**AQUATO® Environmental Technologies GmbH**

Ernstmeierstraße 24
32052 Herford

**Telephone**: +49 (0) 5221 10 21 90
**Fax**: +49 (0) 5221 10 21 920
**E-mail**: info@aquato.de