Dr.WEB

Industrial for Unix File Servers

Administrator Manual



© Doctor Web, 2026. All rights reserved

This document is intended for information and reference purposes regarding the Dr.Web software discussed herein. This document is not a basis for exhaustive conclusions about the presence or absence of any functional and/or technical features in Dr.Web software and cannot be used to determine whether Dr.Web software meets any requirements, technical specifications and/or parameters, and other third-party documents.

This document is the property of Doctor Web and may be used solely for the personal purposes of the purchaser of the product. No part of this document may be reproduced, published or transmitted in any form or by any means, without proper attribution, for any purpose other than the purchaser's personal use.

Trademarks

Dr.Web, SpIDer Mail, SpIDer Guard, CureIt!, CureNet!, AV-Desk, KATANA and the Dr.WEB logo are trademarks and registered trademarks of Doctor Web in Russia and/or other countries. Other trademarks, registered trademarks and company names used in this document are the property of their respective owners.

Disclaimer

In no event shall Doctor Web and its resellers or distributors be liable for any errors or omissions, or for any loss of profit or any other damage caused or alleged to be caused directly or indirectly by this document, or by the use of or inability to use the information contained in this document.

Dr.Web Industrial for Unix File Servers
Version 11.1
Administrator Manual
10/7/2026

For information on regional and international offices of Doctor Web, please visit the official website.

Doctor Web

Doctor Web develops and distributes Dr.Web information security solutions that provide effective protection against malicious software and spam.

Doctor Web customers include home users around the world, government agencies, small businesses, and nationwide corporations.

Since 1992, Dr.Web anti-virus solutions have been known for their continuous excellence in malware detection and compliance with international information security standards.

The state certificates and awards received by Dr.Web solutions, as well as the worldwide use of our products, are the best evidence of exceptional trust in the company products.

We thank all our customers for their support and devotion to Dr.Web products!

Table of Contents

1. Introduction 8

2. Conventions and Abbreviations 9

3. About This Product 10

3.1. Basic Features of Dr.Web Industrial for Unix File Servers 11

3.2. Structure of Dr.Web Industrial for Unix File Servers 12

3.2.1. Basic Anti-Virus Components 13

3.2.2. Threat Search Components 14

3.2.3. Service Components 15

3.2.4. Interface Components 16

3.3. Putting in Quarantine 19

3.4. File Permissions 20

3.5. Operation Modes 21

4. System Requirements and Compatibility 24

5. Licensing 29

6. Installing and Uninstalling 30

6.1. Installing Dr.Web Industrial for Unix File Servers 31

6.1.1. Installing the Universal Package 32

6.1.1.1. Installing from the Command Line 34

6.1.2. Installing from the Repository 34

6.2. Upgrading Dr.Web Industrial for Unix File Servers 38

6.2.1. Getting Current Updates 38

6.2.2. Upgrading to a Newer Version 39

6.2.3. Updating Offline 42

6.3. Uninstalling Dr.Web Industrial for Unix File Servers 43

6.3.1. Uninstalling the Universal Package 43

6.3.1.1. Uninstalling from the Command Line 44

6.3.2. Uninstallation of Dr.Web Industrial for Unix File Servers Installed from the Repository 44

6.4. Additional Information 47

6.4.1. Dr.Web Industrial for Unix File Servers Packages and Files 47

6.4.2. Custom Installation and Uninstallation of Components 50

6.5. Configuring Security Subsystems 55

6.5.1. Configuring SELinux Security Policies 55

6.5.2. Starting in CSE Mode (Astra Linux SE 1.6 and 1.7) 59
7. Getting Started 60
7.1. Registration and Activation 60
7.1.1. Key File 62
7.2. Testing Product Operation 63
7.3. Configuring File System Monitoring 64
7.4. Integration with a Samba File Server 66
8. Brief Instructions 70
9. Dr.Web Industrial for Unix File Servers Components 75
9.1. Dr.Web ConfigD 75
9.1.1. Operating Principles 75
9.1.2. Command-Line Arguments 77
9.1.3. Configuration Parameters 78
9.2. Dr.Web Ctl 81
9.2.1. Command-Line Call Format 83
9.2.2. General Options 83
9.2.3. Commands 83
9.2.3.1. Anti-Virus Scanning Commands 84
9.2.3.2. Commands to Manage Detected Threats and Quarantine 97
9.2.3.3. Configuration Management Commands 102
9.2.3.4. Advanced Commands 104
9.2.3.4.1. Commands to Manage Updates and Operation in Centralized Protection Mode 105
9.2.3.4.2. Information Commands 106
9.2.4. Usage Examples 111
9.2.5. Configuration Parameters 114
9.3. Dr.Web Management Web Interface 115
9.3.1. Managing the Components 117
9.3.2. Threat Management 118
9.3.3. Managing Settings 119
9.3.3.1. Managing Centralized Protection 125
9.3.4. Scanning Local Files 127
9.4. SpIDer Guard 131
9.4.1. Operating Principles 131
9.4.2. Command-Line Arguments 134
9.4.3. Configuration Parameters 135

9.5. SpIDer Guard for SMB **143**
9.5.1. Operating Principles 143
9.5.2. Command-Line Arguments 145
9.5.3. Configuration Parameters 146
9.5.4. Building the VFS SMB Module 152
9.6. Dr.Web File Checker **155**
9.6.1. Operating Principles 155
9.6.2. Command-Line Arguments 156
9.6.3. Configuration Parameters 157
9.7. Dr.Web Network Checker **159**
9.7.1. Operating Principles 159
9.7.2. Command-Line Arguments 160
9.7.3. Configuration Parameters 162
9.7.4. Creating a Scanning Cluster 167
9.8. Dr.Web Scanning Engine **170**
9.8.1. Operating Principles 170
9.8.2. Command-Line Arguments 171
9.8.3. Configuration Parameters 175
9.9. Dr.Web Updater **178**
9.9.1. Operating Principles 178
9.9.2. Command-Line Arguments 179
9.9.3. Configuration Parameters 180
9.10. Dr.Web ES Agent **186**
9.10.1. Operating Principles 186
9.10.2. Command-Line Arguments 187
9.10.3. Configuration Parameters 188
9.11. Dr.Web HTTPD **190**
9.11.1. Operating Principles 190
9.11.2. Command-Line Arguments 191
9.11.3. Configuration Parameters 192
9.11.4. Description of the HTTP API 195
9.12. Dr.Web SNMPD **216**
9.12.1. Operating Principles 216
9.12.2. Command-Line Arguments 217
9.12.3. Configuration Parameters 218
9.12.4. Integration with Monitoring Systems 222

9.12.5. Dr.Web SNMP MIB 231
9.13. Dr.Web MeshD 250
9.13.1. Operating Principles 250
9.13.2. Command-Line Arguments 252
9.13.3. Configuration Parameters 253
9.14. Dr.Web StatD 257
9.14.1. Operating Principles 257
9.14.2. Command-Line Arguments 257
9.14.3. Configuration Parameters 258
10. Appendices 260
10.1. Appendix A. Types of Computer Threats 260
10.2. Appendix B. Neutralizing Computer Threats 264
10.3. Appendix C. Technical Support 266
10.4. Appendix D. Dr.Web Industrial for Unix File Servers Configuration File 268
10.4.1. File Structure 268
10.4.2. Parameter Types 269
10.5. Appendix E. Generating SSL Certificates 273
10.6. Appendix F. Known Errors 276
10.7. Appendix G. List of Abbreviations 321

1. Introduction

The Dr.Web Industrial for Unix File Servers solution offers reliable protection of your server from distribution of all possible types of computer threats by using the most advanced threat detection and neutralization technologies. This improves the quality of services provided by the server.

This manual is intended to help administrators of the servers running Unix-like OSes such as GNU/Linux and FreeBSD (hereinafter, they will be referred to as Unix), to install and use Dr.Web Industrial for Unix File Servers.

If the previous version of Dr.Web Industrial for Unix File Servers is already installed on your computer and you wish to upgrade the product to an up-to-date version, follow the steps described in the upgrade procedure (see the Upgrading to a Newer Version section).

2. Conventions and Abbreviations

The following symbols and text conventions are used in this guide:

Convention	Comment
	An important note or instruction
	A warning about possible errors or important notes that require special attention
Anti-virus network	A new term or an emphasis on a term in descriptions
<IP-address>	Placeholders
Save	Names of buttons, windows, menu items and other program interface elements
CTRL	Names of keyboard keys
`/home/user`	Names of files and folders, code examples
Appendix A	Cross-references to document chapters or internal hyperlinks to webpages



Commands entered in the command line using a keyboard (in a terminal or a terminal emulator) are marked with the command prompt character `$` or `#` in the manual. The character indicates the privileges required for running the specified command. According to the standard convention for Unix-based systems:

`$`—command can be run with user rights.

`#`—command must be run with superuser (usually *root*) privileges. To elevate the privileges, use `su` and `sudo` commands.

The list of abbreviations is provided in the Appendix G. List of Abbreviations section.

3. About This Product

Function

Dr.Web Industrial for Unix File Servers is designed to protect files of physical and virtual file servers running Unix-like OSes (GNU/Linux and FreeBSD) from viruses and other types of malware, as well as to prevent distribution of threats for various platforms. When operating in centralized protection mode, Dr.Web Industrial for Unix File Servers is controlled by a server managed by Dr.Web Industrial. By default, Dr.Web Industrial for Unix File Servers does not interfere with the operating system and only notifies the user of detected threats; this allows to use the anti-virus solution in an industrial infrastructure, in which emergency shutdown of machinery and outage are unacceptable.

Data protection

Main components (scan engine and virus databases) are not only highly effective and resource-sparing, but also cross-platform, which lets Doctor Web experts create reliable anti-virus solutions protecting computers running popular operating systems from threats that target various platforms. Currently, along with Dr.Web Industrial for Unix File Servers, Doctor Web offers other anti-virus solutions for Unix-like operating systems (GNU/Linux and FreeBSD), macOS and Windows. Moreover, other anti-virus products have been developed to deliver protection for devices running Android and Aurora OSes.

Components of Dr.Web Industrial for Unix File Servers are constantly updated, and virus databases are regularly supplemented with new records to ensure up-to-date protection of files of physical and virtual file servers. For additional protection against unknown malware, heuristic analysis methods implemented by the anti-virus engine are used.

Operation in infrastructure

Before using Dr.Web software, you must make sure there are no conflicts between the anti-virus software and the software and/or firmware of your corporate infrastructure. Install all the latest updates and perform all secure configuration steps outlined in this documentation. If critical corporate infrastructure segments interface with Dr.Web software, it is strongly recommended that all updates to program modules and virus databases be tested in an isolated test environment.

Additional information:

- Basic Features of Dr.Web Industrial for Unix File Servers
- Structure of Dr.Web Industrial for Unix File Servers
- Putting in Quarantine
- File Permissions

- Operation Modes

3.1. Basic Features of Dr.Web Industrial for Unix File Servers

1. **Detection and neutralization of threats.** Scanning for malicious programs of any kind (various viruses, including those that infect mail files and boot records, trojans, email worms and so on) and unwanted software (adware, joke programs and dialers). For details on threat types, refer to Appendix A. Types of Computer Threats.

   Threat detection methods:

   - a *signature analysis*—a scan method allowing to detect known threats registered in virus databases;
   - a *heuristic analysis*—a set of scan methods allowing to detect threats that are not known yet.

   The heuristic analyzer may cause false-positive detections. Thus, objects that contain threats detected by the analyzer are considered "suspicious". It is recommended that you quarantine such files and send them for analysis to the Doctor Web anti-virus laboratory. For details on methods used to neutralize threats, refer to Appendix B. Neutralizing Computer Threats.

   When scanning the file system on the user request, it is possible to perform either a full scan of all the file system objects available to the user, or a custom scan of the specified objects only (individual directories or files that meet the specified criteria). In addition, it is possible to perform an individual check of boot records of volumes and executable files which started the processes that are currently active in the system. In the latter case, when a threat is detected, a malicious executable file is not only neutralized, but all processes started by it are forcibly terminated. On systems that implement a mandatory model of file access with a set of different access levels, the scanning of files that are not available at the current access level can be done in special standalone instance mode.

   All objects containing threats detected in the file system are registered in a permanent threat registry, except those threats that were detected in standalone instance mode.

   The Dr.Web Ctl command-line tool included in Dr.Web Industrial for Unix File Servers allows to scan file systems of remote network hosts providing remote terminal access via SSH or Telnet for threats.

   Remote scanning can be used only for detection of malicious or suspicious files on a remote host. To eliminate detected threats on the remote host, use administration tools provided directly by this host. For example, update firmware on routers and other "smart" devices, connect to computing machines (including in remote terminal mode) and perform corresponding operations in their file system (deleting or moving files, and so on), or run anti-virus software installed on them.

2. **Monitoring of access to files within:**
   - **An OS file system.** Monitors file events and attempts to run executables. This feature allows to detect and neutralize malware at an attempt to infect the server file system.

Besides the standard monitoring mode, you can enable the enhanced (Paranoid) mode in which the monitor blocks access to files until their scanning is finished (this allows you to prevent access to an infected file, but a scanning result is available only after an application accesses a file). The enhanced mode increases the security level but slows down access to the files that are not scanned yet.



The volume monitoring feature is available only on operating systems of the GNU/Linux family. The component that provides this feature is not shipped for other supported OSes.

- **Samba shared directories.** Read and write operations of local and remote users of the file server are monitored. This feature allows to detect and neutralize malware instantly at an attempt of copying it to the file storage, which prevents its further distribution over the network.
- **Reliable isolation of infected or suspicious objects** detected within the server file system in a special storage known as quarantine to prevent any harm to the system. When quarantined, objects are renamed according to special rules and, if necessary, they can be restored to their original location only on user demand.

3. **Automatic update** of the scanning engine, virus databases to maintain the high level of protection against malware.
4. **Collection of statistics** on scans and threat events. Logging detected threats. Sending of notifications of detected threats via SNMP to external monitoring systems and a centralized protection server if Dr.Web Industrial for Unix File Servers operates in centralized protection mode.
5. **Operation in centralized protection mode** to implement single security policies adopted within an anti-virus network comprising this server and managed by Dr.Web Industrial.

3.2. Structure of Dr.Web Industrial for Unix File Servers

Dr.Web Industrial for Unix File Servers is a software suite consisting of a set of components, where each component has its own set of functions. The components are separated into the following categories according to their objectives:

- Basic anti-virus components that form the Dr.Web Industrial for Unix File Servers core. In the absence of the components under this category, the product cannot scan files (and other data) for viruses and other threats.
- Threat search components. These components are used to solve Dr.Web Industrial for Unix File Servers basic tasks—detecting threats and potentially dangerous objects. In their operation the components falling under this category use basic anti-virus components.
- Service components that complete auxiliary tasks to maintain anti-virus protection (updating anti-virus databases, connecting to centralized protection servers, managing general Dr.Web Industrial for Unix File Servers operation and so on).
- Interface components that provide (the user or third-party applications) with the Dr.Web Industrial for Unix File Servers management interface.

Below is the list of Dr.Web Industrial for Unix File Servers components.

3.2.1. Basic Anti-Virus Components

Component	Description
Dr.Web Virus-Finding Engine	Anti-virus engine. Implements algorithms to detect viruses and other malware (by using signature and heuristic analyses). Managed by the Dr.Web Scanning Engine component Library file: `drweb32.dll`. Logged internal name: `CoreEngine`
Dr.Web Scanning Engine	Scanning engine. This component loads Dr.Web Virus-Finding Engine and virus databases. • Passes the contents of files and boot records to the anti-virus engine for scanning. • Manages a queue of the files to be scanned. • Cures threats to which this action is applicable. Operates under the control of the Dr.Web ConfigD daemon or in standalone mode. Used by the Dr.Web File Checker and Dr.Web Network Checker components. Also can be used by the Dr.Web MeshD component (in particular modes) and by external (in relation to Dr.Web Industrial for Unix File Servers) applications using directly the Dr.Web Scanning Engine API Executable file: `drweb-se`. Logged internal name: `ScanEngine`
Virus databases	Automatically updated database of signatures of viruses and other threats, as well as of malware detection and neutralization algorithms. Used by Dr.Web Virus-Finding Engine and is supplied with it
Dr.Web File Checker	Component for scanning file system objects and a quarantine manager. • Receives tasks from the threat scanning component on scanning files in the local (relative to Dr.Web Scanning Engine) file system. • Surfs the file system directories according to the task, sends files for scanning to Dr.Web Scanning Engine and notifies the client components about the scanning progress. • Deletes infected files, puts them in and restores them from quarantine, manages quarantine directories. • Builds the cache and keeps it up-to-date. The cache contains information about previously scanned files to reduce the frequency of rescanning files.

Component	Description
	Used by components that scan file system objects, such as SpIDer Guard, SpIDer Guard for SMB Executable file: `drweb-filecheck`. Logged internal name: `FileCheck`
Dr.Web Network Checker	Network data scanning agent. • Used to send data to the scanning engine for actual scanning. The data is sent by components of the product via the network. • Allows Dr.Web Industrial for Unix File Servers to manage distributed file scanning: to receive/transmit files for scanning from/to remote hosts. For that purpose, remote hosts must feature an installed and running Dr.Web for Unix operating systems. In the distributed scanning mode, it allows automatic distribution of scanning load among available hosts by reducing load on hosts with a large number of scanning tasks (for example, on mail servers and internet gateways). If the network contains partner hosts that can receive data for scanning, the components that use Dr.Web Network Checker for scanning may operate without local Dr.Web Scanning Engine. Thus, local Dr.Web Scanning Engine, Dr.Web Virus-Finding Engine and virus databases may be absent. For security reasons, files are transmitted over the network using SSL Executable file: `drweb-netcheck`. Logged internal name: `NetCheck`
Dr.Web MeshD	Component that connects Dr.Web Industrial for Unix File Servers to a local cloud, which allows Dr.Web for Unix products to exchange updates, results of file scanning, transmit files to each other for scanning, as well as to provide scanning engine services directly. If this component is included in the product and the local cloud to which it is connected contains hosts providing scanning engine services, local Dr.Web Scanning Engine, Dr.Web Virus-Finding Engine and virus databases may be absent Executable file: `drweb-meshd`. Logged internal name: `MeshD`

3.2.2. Threat Search Components

Component	Description
SpIDer Guard	GNU/Linux File System Monitor.

Component	Description
	Operates in background mode and controls file operations (such as creation, opening, closing, running) in GNU/Linux file systems. It sends the Dr.Web File Checker component requests to scan new or modified files as well as executables of programs when they are run. Operates with a file system using the *fanotify* system mechanism. The monitor can operate in enhanced or "paranoid" mode, blocking access to the files that have not been scanned yet until the scan is complete. By default, a regular monitoring mode is enabled. The component is supplied only with the distributions designed for GNU/Linux OSes. Executable file: `drweb-spider`. Logged internal name: `LinuxSpider`
SpIDer Guard for SMB	Samba shared directory monitor. Operates in a background mode and monitors file system operations (such as creating, opening, closing, read and write operations) in directories selected as the Samba SMB server file storage. Sends requests for scanning of new and modified files to the Dr.Web File Checker component. For integration with the file server uses VFS SMB modules that operate on the Samba server side Executable file: `drweb-smbspider-daemon`. Logged internal name: `SMBSpider`

3.2.3. Service Components

Component	Description
Dr.Web ConfigD	Dr.Web Industrial for Unix File Servers configuration daemon. • Starts and stops other product components depending on the settings. • Restarts components if a failure in their operation occurs. Starts components at the request of other components. Informs active components when another component starts or shuts down. • Stores information about current license keys and settings and provides this information to all components. Receives adjusted settings and license keys from the designated components of Dr.Web Industrial for Unix File Servers. Notifies other components of changes in license keys and settings Executable file: `drweb-configd`.

Component	Description
	Logged internal name: `ConfigD`
Dr.Web ES Agent	Centralized protection agent. Ensures product operation in the centralized protection and mobile modes. • Provides connection between the product and the centralized protection server, receives a license key file, updates of the virus databases and anti-virus engine. • Sends the information about the Dr.Web Industrial for Unix File Servers components, their status and statistics on threat events to the server Executable file: `drweb-esagent`. Logged internal name: `ESAgent`
Dr.Web StatD	Component for storing Dr.Web Industrial for Unix File Servers component operation events. Receives and stores product component events, such as unexpected shutdown, threat detection and so on. Executable file: `drweb-statd`. Logged internal name: `StatD`
Dr.Web Updater	Updating component. Downloads updates of virus databases, the anti-virus engine from Doctor Web servers. The updates can be downloaded automatically, on schedule, and on user demand (via the Dr.Web Ctl utility or management web interface) Executable file: `drweb-update`. Logged internal name: `Update`

3.2.4. Interface Components

Component	Description
Dr.Web HTTPD	Dr.Web Industrial for Unix File Servers component management web server. Provides a custom HTTP API for managing Dr.Web Industrial for Unix File Servers components. This API is used by the management web interface. For security reasons, the component uses HTTPS to connect to the management web interface. Uses Dr.Web Network Checker to send data for scanning to Dr.Web Scanning Engine

Component	Description
	Executable file: `drweb-httpd`. Logged internal name: `HTTPD`
Dr.Web Management Web Interface	Management Web Interface. Can be accessed using any browser on a local or remote host. The management web interface enables the product not to use third-party web servers, such as Apache HTTP Server, or remote administration tools, such as Webmin. The functionality is ensured by the Dr.Web HTTPD web server
Dr.Web Ctl	Tool designed to manage Dr.Web Industrial for Unix File Servers from the command line. Allows the user to start file scanning, view and manage quarantined objects, start a virus database update procedure, connect Dr.Web Industrial for Unix File Servers to or disconnect it from a centralized protection server, view and change product parameters Executable file: `drweb-ctl`. Logged internal name: `Ctl`
Dr.Web SNMPD	SNMP agent. Designed for integration of Dr.Web Industrial for Unix File Servers into external monitoring systems over SNMP. Such integration allows you to monitor the state of the product components and to collect statistics on threat detection and neutralization. Supports SNMP v2c and v3 protocols Executable file: `drweb-snmpd`. Logged internal name: `SNMPD`

The figure below shows the structure of Dr.Web Industrial for Unix File Servers and its interaction with external applications.

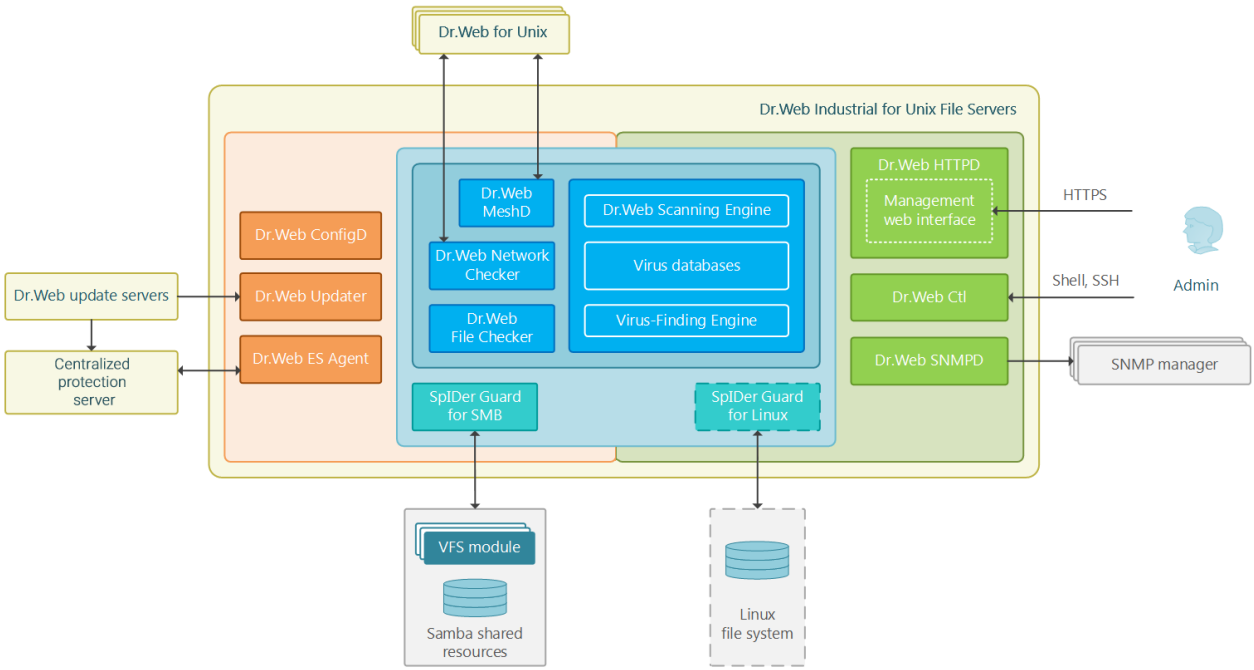


Figure 1. The structure of Dr.Web Industrial for Unix File Servers

In this scheme, the following notations are used:

	Dr.Web Industrial for Unix File Servers as a whole and external Dr.Web applications not distributed with it
	Programs external to Dr.Web Industrial for Unix File Servers and products that integrate with it
	Service components that perform particular anti-virus protection tasks (updating anti-virus databases, connecting to centralized protection servers, overall coordination of operation and so on)
	Interface components that provide (the user or third party applications) with the Dr.Web Industrial for Unix File Servers management interface
	Components used for anti-virus scanning
	Basic anti-virus components that form the Dr.Web Industrial for Unix File Servers core. Used by the components that perform data and file scans

Components marked with a dotted line can be absent depending on the Dr.Web Industrial for Unix File Servers distribution or usage.

For details on Dr.Web Industrial for Unix File Servers components, refer to Dr.Web Industrial for Unix File Servers Components.

3.3. Putting in Quarantine

The quarantine of Dr.Web Industrial for Unix File Servers is a system of directories designed to isolate files containing detected threats that cannot be currently cured for some reason. For example, a detected threat can be incurable because Dr.Web Industrial for Unix File Servers is still unaware of it (for example, the threat was detected by the heuristic analyzer, but the virus databases do not cover the threat signature and a method to cure) or curing causes errors. Moreover, a file can be quarantined on user demand if the user selected the corresponding action in the list of detected threats or specified this action in settings as a reaction to threats of a specific type.

When a file is quarantined, it is renamed according to special rules to prevent its identification by users and applications and inhibit accessing it without quarantine management tools implemented in Dr.Web Industrial for Unix File Servers. Moreover, when a file is quarantined, its execution bit is always reset to prevent running this file.

Quarantine directories are located in:

- *user home directory* (if multiple user accounts exist on the computer, a separate quarantine directory can be created for each of the users);
- *root directory of each logical volume* mounted on the file system.

Dr.Web Industrial for Unix File Servers quarantine directories are always named `.com.drweb.quarantine` and are not created until the "**Quarantine**" (`QUARANTINE`) action is applied, that is, quarantine directories are not created until a threat is detected. At that, only a directory required for isolation of the file is created. When selecting the directory, the name of the file owner is used. Search is performed upwards from the directory containing the file to the file system root `/`; if the home directory of the owner is reached, the file is isolated in the quarantine directory under the home directory. Otherwise, the file is isolated in the quarantine directory created under the volume root directory (which is not always the same as the file system root directory). Thus, any infected file put in quarantine is always kept on the same volume, which provides for correct operation of quarantine in case there are removable data storage devices and other volumes that can be mounted in the file system occasionally and on different mount points.

A user can manage quarantine contents in command-line mode using the Dr.Web Ctl utility, or via the management web interface. At that, all currently available quarantine directories containing isolated objects are always processed as a single entity. From the point of view of a user who views the contents of the combined quarantine, the quarantine directory located in the home directory is a *User* quarantine, and all other quarantine directories are a *System* quarantine.



You can manage quarantine even if no active license was found; however, isolated objects cannot be cured in this case.

3.4. File Permissions

To scan objects of the file system and neutralize threats, Dr.Web Industrial for Unix File Servers (or rather the user under whom it runs) requires the following permissions:

Action	Required rights
Listing all detected threats	Unrestricted. No special permission required.
Output of container contents (an archive, email file, and so on) (display only corrupted or malicious elements)	Unrestricted. No special permission required.
Moving to quarantine	Unrestricted. The user can quarantine all infected files regardless of read or write permissions on them.
Deleting threats	The user needs to have write permissions for the file that is being deleted. If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined and not deleted.
Curing	Unrestricted. The access permissions and owner of a cured file remain the same after curing. The file can be removed if deletion can cure the detected threat.
Restoring a file from quarantine	The user should have permissions to read the file and to write to the restore directory.
Deleting a file from quarantine	The user must possess write permissions to the file that was moved to quarantine.

To start the Dr.Web Ctl command-line management tool with superuser privileges (usually *root*), use the `su` command to change the user or the `sudo` command to run the command as another user.

3.5. Operation Modes

In this section

- Centralized protection concept
- Connecting to a centralized protection server
- Disconnecting from a centralized protection server

Dr.Web Industrial for Unix File Servers can operate either in standalone mode or as a part of a corporate or private *anti-virus network* managed by a *centralized protection server*. Such operation mode is called a *centralized protection mode*. Using this mode does not require installation of additional software or Dr.Web Industrial for Unix File Servers re-installation or uninstallation.

- In *standalone mode*, a protected computer is not connected to the anti-virus network and its operation is managed locally. In this mode, configuration and license key files are located on local disks and Dr.Web Industrial for Unix File Servers is fully managed by the protected computer. Updates for virus databases are received from Doctor Web update servers.
- In *centralized protection mode*, protection of the computer is managed by the centralized protection server. In this mode, some functions and settings of Dr.Web Industrial for Unix File Servers can be adjusted in accordance with the general (corporate) anti-virus protection policy implemented on the anti-virus network. The license key file used for operating in centralized protection mode is received from the centralized protection server to which Dr.Web Industrial for Unix File Servers is connected. The license or demo key file stored on the local computer, if any, is not used. Statistics on threat events together with information on Dr.Web Industrial for Unix File Servers operation are sent to the centralized protection server. Updates for virus databases are also received from the centralized protection server. Configuring the centralized protection mode is described in the Dr.Web Enterprise Security Suite Administrator Manual for managing Dr.Web Industrial for Unix File Servers.
- In *mobile mode*, Dr.Web Industrial for Unix File Servers receives updates from Doctor Web update servers, but uses settings stored locally and a custom license key file that were received from the centralized protection server.

A possibility of configuring the settings of the SpIDer Guard file system monitor as well as enabling or disabling it while Dr.Web Industrial for Unix File Servers is controlled by the centralized protection server are dependent on permissions specified on the server.

Centralized protection concept

Doctor Web solutions for managing centralized protection use a client-server model (see the figure below).

Corporate computers or computers of clients of an IT service provider are protected by *local anti-virus components* (in this case, by Dr.Web Industrial for Unix File Servers), which ensure anti-virus protection and maintain connection to the centralized protection server.

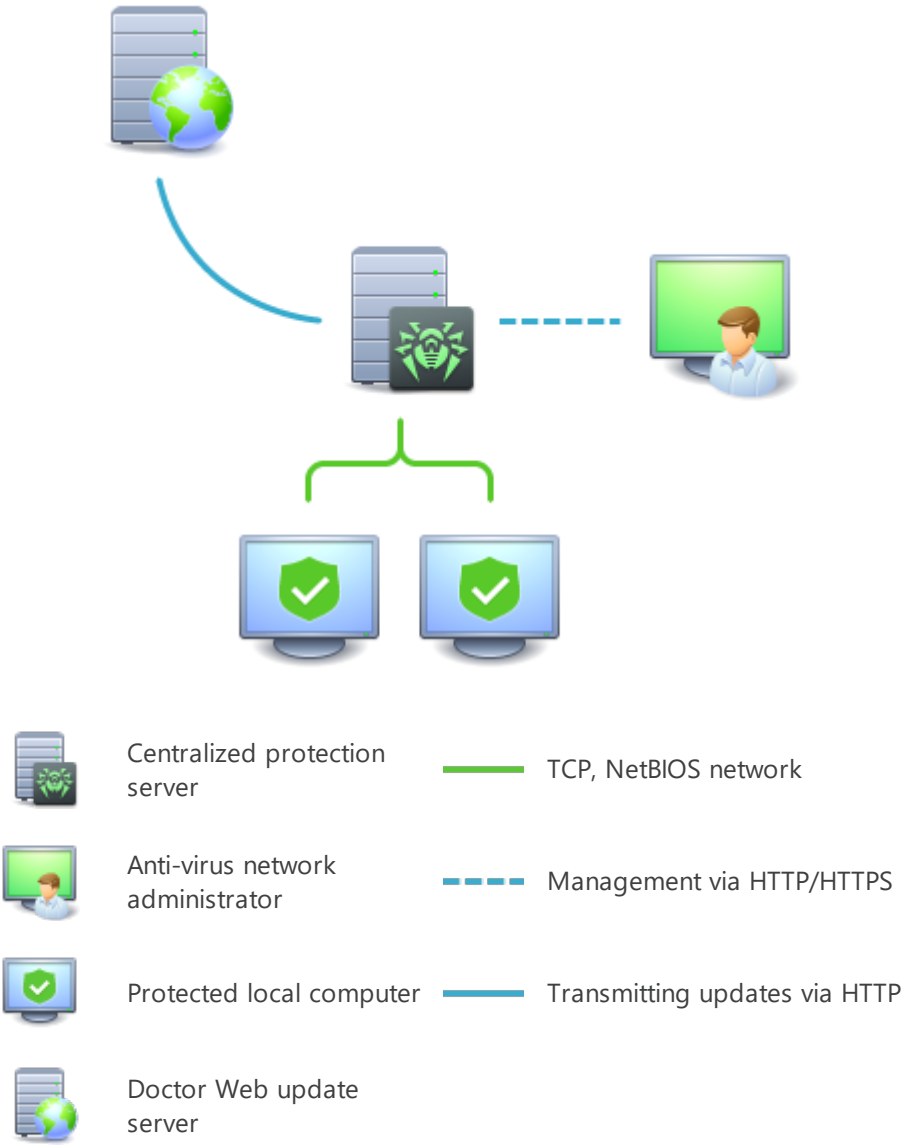


Figure 2. Logical structure of the anti-virus network

Local components are updated and configured from the *centralized protection server*. The entire stream of instructions, data and statistics in the anti-virus network also passes the centralized protection server. The volume of traffic between protected computers and the centralized protection server can be significant, therefore an option for traffic compression is provided. Using encryption while sending data prevents a leak of sensitive data or substitution of software downloaded onto protected computers.

All necessary updates are downloaded to the centralized protection server from Doctor Web update servers.

Changes in the configuration of local anti-virus components and command transfer are performed by anti-virus network administrators using the centralized protection server. The administrators manage configuration of the centralized protection server and topology of the

anti-virus network (for example, they validate connection of a local station to the network) and configure operation of individual local anti-virus components when necessary.



Local anti-virus components are incompatible with anti-virus products of other companies or Dr.Web anti-virus solutions if the latter do not support operation in the centralized protection mode (for example, Dr.Web Anti-virus version 5.0). Installation of two anti-virus programs on the same computer can cause a system crash or a loss of important data.

The centralized protection mode allows exporting and saving Dr.Web Industrial for Unix File Servers operation reports using the centralized protection server. Reports can be exported and saved in the following formats: HTML, CSV, PDF and XML.

Connecting to a centralized protection server

Dr.Web Industrial for Unix File Servers can be connected to the centralized protection server of the anti-virus network using the `esconnect` command of the `drweb-ctl` command-line management tool.



For the verification of the centralized protection server the certificate corresponding to the unique public key of the server is used. By default, the Dr.Web ES Agent centralized protection agent will not allow you to connect to the server unless you specify a certificate file. The certificate file must first be obtained from the administrator of the anti-virus network served by the server to which you want to connect Dr.Web Industrial for Unix File Servers.

If Dr.Web Industrial for Unix File Servers is connected to the centralized protection server, you can switch the product into the mobile mode or switch it back into the centralized protection mode. Switching the mobile mode on or off is accomplished with the help of the `MobileMode` configuration parameter of the Dr.Web ES Agent component.



Dr.Web Industrial for Unix File Servers can switch to the mobile mode only if this is allowed by the settings of the centralized protection server.

Disconnecting from a centralized protection server

Dr.Web Industrial for Unix File Servers can be disconnected from the centralized protection server of the anti-virus network using the `esdisconnect` command of the `drweb-ctl` command-line management tool.

4. System Requirements and Compatibility

In this section

- System requirements
- List of supported operating system versions
  - GNU/Linux
  - FreeBSD
- Required additional packages and components
- Disclaimer
- Supported file servers
  - Samba file service
- Compatibility with security subsystems

System requirements

You can use Dr.Web Industrial for Unix File Servers on a computer that meets the following requirements:

Parameter	Requirements
Platform	Processors of the following architectures and command systems are supported: • Intel/AMD: 32-bit (*IA-32*, *x86*); 64-bit (*x86-64*, *x64*, *amd64*) • ARM64 • E2K (*Elbrus*) • IBM POWER9, Power10 (*ppc64el*)
RAM	At least 500 MB of free RAM (1 GB or more is recommended)
Free disk space	At least 2 GB of free disk space on a volume where the product directories are located
Operating system	GNU/Linux (based on kernel version 2.6.37 or later, using `glibc` library 2.13 or later, `systemd` initialization system ver. 209 or later), FreeBSD. The supported operating system versions are listed below. The operating system must support the PAM authentication mechanism
Other	The following valid network connections: • valid Internet connection to enable updates for virus databases and Dr.Web components; • when operating in the centralized protection mode, connection to the server on the local network is enough; connection to the Internet is not required

To enable the correct operation of Dr.Web Industrial for Unix File Servers, open the following ports:

Purpose	Direction	Port numbers
To receive updates	outgoing	80

List of supported operating system versions

GNU/Linux

Platform	Supported GNU/Linux versions
x86_64	• ALT 8 SP • ALT Server 9, 10, 11 • ALT Workstation 9, 10, 11 • Astra Linux Common Edition (Orel) 2.12 • Astra Linux Special Edition 1.6 (with cumulative patch 20200722SE16), 1.7, 1.8 • CentOS 7, 8 • Debian 9, 10, 11, 12, 13 • Fedora 37, 38 • GosLinux IC6 • Red Hat Enterprise Linux 7, 8 • RED OS 7.2 MUROM, RED OS 7.3 MUROM, RED OS 8 • SUSE Linux Enterprise Server 12 SP3 • Ubuntu 18.04, 20.04, 22.04, 24.04
x86	• ALT 8 SP • ALT Workstation 9, 10 • CentOS 7 • Debian 10
ARM64	• ALT 8 SP • ALT Server 9, 10, 11 • ALT Workstation 9, 10, 11 • Astra Linux Special Edition (Novorossiysk) 4.7 • CentOS 7, 8 • Debian 11, 12 • Ubuntu 18.04
E2K	• ALT 8 SP • ALT Server 10

Platform	Supported GNU/Linux versions
	• ALT Workstation 10 • Astra Linux Special Edition (Leningrad) 8.1 (with cumulative patch 8.120200429SE81) • Elbrus-D MCST 1.4 • GS CS Elbrus 8.32 TVGI.00311-28
ppc64el	• CentOS 8 • Ubuntu 20.04



In ALT 8 SP, Elbrus-D MCST 1.4 and GosLinux IC6 mandatory access control is not supported.

For other GNU/Linux versions that meet the abovementioned requirements, full compatibility with Dr.Web Industrial for Unix File Servers is not guaranteed. If a compatibility issue occurs, contact our technical support.

FreeBSD

Platform	Supported FreeBSD versions
x86	11, 12, 13, 14
x86_64	11, 12, 13, 14, 15



Dr.Web Industrial for Unix File Servers can be installed on FreeBSD only from a universal package.

The correct operation of Dr.Web Industrial for Unix File Servers on FreeBSD 13, 14 and 15 requires that the `compat12x` compatibility package is installed.

Required additional packages and components

Dr.Web Industrial for Unix File Servers does not require installation of additional packages and OS components (except for the protected server software, see below).

For convenient work with Dr.Web Industrial for Unix File Servers from the command line, it is recommended to enable command auto-completion in your command shell (if disabled).

If you encounter any issue with installation of additional packages and components, refer to the documentation of your operating system version.

Disclaimer

- SpIDer Guard can operate in the enhanced (Paranoid) mode, which blocks access to the files that have not been scanned yet, *only* via fanotify and providing that the OS kernel is built with the enabled `CONFIG_FANOTIFY_ACCESS_PERMISSIONS` option.

Supported file servers

Samba file service

For integration with the Samba file service, the installed and configured Samba file server of versions 3.6–4.24 is required.

The SpIDer Guard for SMB monitor of Dr.Web Industrial for Unix File Servers uses a specialized VFS SMB module to integrate into Samba. Several versions of the VFS SMB module built for different versions of Samba are supplied with the SpIDer Guard for SMB component; however, they may be incompatible with the version of Samba installed on your file server, for example, if your Samba server uses the `CLUSTER_SUPPORT` option.

If VFS SMB modules are incompatible with your Samba server, the *corresponding message* is shown during the Dr.Web Industrial for Unix File Servers installation. In this case, build the VFS SMB module for your Samba server manually (with the `CLUSTER_SUPPORT` option if necessary).

The procedure of building the VFS SMB module from source code is described in the Building the VFS SMB Module section.

Compatibility with security subsystems

By default, Dr.Web Industrial for Unix File Servers does not support the SELinux security subsystem. Moreover, Dr.Web Industrial for Unix File Servers operates by default in a reduced functionality mode on the GNU/Linux systems that use mandatory access models (for example, on the systems distributed with the PARSEC mandatory access subsystem, which assigns different privilege levels, so-called mandatory levels, to users and files).

In case of installing Dr.Web Industrial for Unix File Servers on systems with SELinux (as well as on systems that use mandatory access models), it is necessary to additionally configure the

security subsystem, so that Dr.Web Industrial for Unix File Servers would operate in a full functionality mode. For details, refer to the Configuring Security Subsystems section.

5. Licensing

The rights to use Dr.Web Industrial for Unix File Servers are granted by a license purchased from the Doctor Web company or from its partners. License parameters determining user rights are set in accordance with the License agreement, which the user accepts during Dr.Web Industrial for Unix File Servers installation. The license contains information on the user and the vendor as well as usage parameters of the purchased product, including:

- a list of components licensed to the user;
- a Dr.Web Industrial for Unix File Servers license period;
- other restrictions (for example, a number of computers on which the purchased copy of Dr.Web Industrial for Unix File Servers is allowed for use).

Each Doctor Web product license has a unique serial number associated with a special file stored on the local computer. This file regulates operation of the Dr.Web Industrial for Unix File Servers components in accordance with the license parameters and is called a *license key file*.

You can activate a *demo period* for evaluation purposes. Upon activating the demo period, you gain the right to use the installed copy of Dr.Web Industrial for Unix File Servers with full functionality for this entire period if its activation requirements are observed. At that, a special key file named a *demo* key file is automatically generated.

To use a demo period, fill out the web form. Each request is considered individually. If it is approved, a custom serial number or an archive with a demo key file will be sent to a specified email address.

If there is no valid license or a demo period is not activated (including cases when a license purchased earlier or a demo period is expired), anti-virus functions of Dr.Web Industrial for Unix File Servers are blocked. Furthermore, the service for receiving updates for Dr.Web virus databases from Doctor Web update servers becomes unavailable. However, you can activate the Dr.Web Industrial for Unix File Servers by connecting it to a centralized protection server as a part of an anti-virus network administered by an enterprise or an internet service provider. In this case, anti-virus functions and updates of the product instance installed on the computer included in the anti-virus network are managed by the centralized protection server.

6. Installing and Uninstalling

This section describes how to install and uninstall Dr.Web Industrial for Unix File Servers, as well as how to obtain current updates and upgrade to a new version, if an earlier version of Dr.Web Industrial for Unix File Servers is already installed on your computer.

Moreover, this section describes the procedure of custom installation and uninstallation of the Dr.Web Industrial for Unix File Servers components (for example, to resolve errors that occurred during the operation of Dr.Web Industrial for Unix File Servers or to install the product with a limited function set) and configuration of advanced security subsystems (such as SELinux) that could be necessary for installation or operation of Dr.Web Industrial for Unix File Servers.

To perform these procedures, superuser (usually the *root* user) privileges are required. To gain superuser privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.



Compatibility of Dr.Web Industrial for Unix File Servers with anti-virus products of other developers is not guaranteed. Due to the fact that the installation of two anti-viruses on one computer can cause errors of the operating system and a loss of important data, it is strongly recommended that you uninstall anti-virus products of other developers before the installation of Dr.Web Industrial for Unix File Servers.

If your computer already has another Dr.Web anti-virus product installed from the universal package (`.run`), and you want to install one more Dr.Web anti-virus product (for example, you have Dr.Web Desktop Security Suite installed from the universal package, and you want to install Dr.Web Industrial for Unix File Servers in addition to it), it is necessary to make sure that the version of the already installed product is the same as the version of Dr.Web Industrial for Unix File Servers to be installed. If the version of the product to be installed is newer than the installed product version, it is necessary, before installation, to upgrade the installed Dr.Web Industrial for Unix File Servers to the version of the product you want to install additionally.

Dr.Web Industrial for Unix File Servers can be installed on FreeBSD only from a universal package.

Additional information:

- Installing Dr.Web Industrial for Unix File Servers
- Upgrading Dr.Web Industrial for Unix File Servers
- Uninstalling Dr.Web Industrial for Unix File Servers
- Configuring Security Subsystems
- Dr.Web Industrial for Unix File Servers Packages and Files
- Custom Installation and Uninstallation of Components

6.1. Installing Dr.Web Industrial for Unix File Servers

To install Dr.Web Industrial for Unix File Servers, do one of the following:

1. Download the installation file, which is a universal package for Unix systems, from the Doctor Web official website. The package is supplied with a console installer; its operation in the graphical desktop mode requires a terminal emulator.
2. Install Dr.Web Industrial for Unix File Servers as a set of native packages (to do that, connect to the corresponding package repository of Doctor Web).



Dr.Web Industrial for Unix File Servers can be installed on FreeBSD only from a universal package.

The correct operation of Dr.Web Industrial for Unix File Servers on FreeBSD 13, 14 and 15 requires that the `compat12x` compatibility package is installed.

It is recommended to install Dr.Web Industrial for Unix File Servers from a universal package on ALT 8 SP and other OSes that use outdated versions of a package manager.

After installing Dr.Web Industrial for Unix File Servers, the IMA/EVM subsystem of ALT 8 SP 11100-01 can issue a warning about potential integrity violation. To avoid this warning, run the command after installing Dr.Web Industrial for Unix File Servers:

```
# integalert fix
```

ALT 8 SP 11100-02 and ALT 8 SP 11100-03 OSes are not subjected to this issue.



After installing Dr.Web Industrial for Unix File Servers using any of the methods provided in this manual, you will need to activate the license or install the key file. In addition, you can connect Dr.Web Industrial for Unix File Servers to a centralized protection server (for details, refer to the Licensing section). Until that, *anti-virus protection is disabled*. In addition, in some cases it is necessary to configure the basic functionality of Dr.Web Industrial for Unix File Servers, as described in the Getting Started section.

During the installation (using either a `.run` universal package or native packages by means of a package manager), messages about Dr.Web Industrial for Unix File Servers installation results are sent at the local email address `root@localhost`.

Dr.Web Industrial for Unix File Servers installed using any of the methods provided in this section can be uninstalled or updated if fixes for its components are available or a new version of Dr.Web Industrial for Unix File Servers is released. If required, you can also configure GNU/Linux security subsystems for correct operation of Dr.Web Industrial for Unix File Servers. If an issue with individual components occurs, you can perform their custom installation and uninstallation without uninstalling Dr.Web Industrial for Unix File Servers.

6.1.1. Installing the Universal Package

The Dr.Web Industrial for Unix File Servers universal package is distributed as an installation file named `drweb-<version>-av-srv-<OS>-<platform>.run`, where <*OS*> is a Unix-like OS, <*platform*> is the platform for which Dr.Web Industrial for Unix File Servers is intended (`x86` for 32-bit platforms, `amd64`, `arm64`, `e2s` and `ppc64el` for 64-bit platforms), for example:

```
drweb-11.1.0-av-srv-linux-amd64.run
```



The name of the installation file corresponding to the above-mentioned format is referred to as <*.run file name*>, and a path to this file—as <*.run file path*> below in this section.

The version of Dr.Web Industrial for Unix File Servers is defined by the first two dot-separated figures in the name of the universal package.

To install Dr.Web Industrial for Unix File Servers components

1. If you do not have the installation file (universal package), download it from the official website of the Doctor Web company.
2. Save the installation file to the hard disk drive of the computer to any writeable directory (for example, `/home/<username>`, where <*username*> is the current user name).
3. Allow the file to start, for example, by running the command:

   ```
   # chmod +x <.run file path>
   ```
4. Start it by running the command:

   ```
   # <.run file path>
   ```

   or use a standard file manager of your graphical shell for both changing the file properties (permissions) and running the file.



If you install Dr.Web Industrial for Unix File Servers on the Astra Linux SE OS of versions 1.6 and 1.7 operating in *closed software environment* (CSE) mode, the installer can fail to start because the Doctor Web public key is not on the list of trusted keys. In this case, configure the CSE mode (see Starting in CSE Mode (Astra Linux SE 1.6 and 1.7)) and start the installer again.

First, an integrity check of the archive is run, after which the archived files are unpacked to a temporary directory and the installer is started. If the installer is started without root privileges, it attempts to elevate its privileges asking you for the root password (the `sudo` utility is used). If the attempt fails, the installation process aborts.

If the file system partition containing the temporary directory has not enough free space for the unpacked files, the installation process is aborted and a corresponding message is displayed. In this case, change the value of the `TMPDIR` system environment variable so that it points to a directory located on a partition with enough free space and restart the installation. You can also use the `--target` option to set the target directory (for more details, see the Custom Installation and Uninstallation of Components section).

An installer for a command-line mode runs (to run it in a graphical desktop environment, you need a terminal emulator).

5. Follow the installer instructions.

You can also start the installer in silent mode by running the command:

```
# <.run file path> -- --non-interactive
```

In this case the installer is started in silent mode without displaying program dialogs of the command-line mode.

Using this option means that you accept the terms of the Dr.Web License Agreement. After installing Dr.Web Industrial for Unix File Servers, you can find the License Agreement text in the file `/opt/drweb.com/share/doc/LICENSE` for GNU/Linux or `/usr/local/libexec/drweb.com/share/doc/LICENSE` for FreeBSD. The file extension indicates the language of the License Agreement. The `LICENSE` file (without any extension) stores the Dr.Web License Agreement in English. If you do not accept the terms of the License Agreement, you must uninstall Dr.Web Industrial for Unix File Servers after its installation.

Superuser (usually the *root* user) privileges are required to start the installer in silent mode. To elevate your privileges, use the `su` or `sudo` command.

If the GNU/Linux distribution you use features the SELinux security subsystem, it can interrupt the installation process. If such situation occurs, temporarily set SELinux to the *Permissive* mode. To do this, run the command:

```
# setenforce 0
```

Restart the installer afterwards. When the installation completes, configure SELinux security policies to enable correct operation of the product components.

All unpacked installation files are deleted once the installation process completes.

It is recommended that you save the `.run` file used for the installation to reinstall Dr.Web Industrial for Unix File Servers or its components without the need to update its version.

6.1.1.1. Installing from the Command Line

Once the installation program for the command line starts, the command prompt is displayed on the screen.

1. To start installation, enter `Yes` or `Y` in response to the "Do you want to continue?" question. To exit the installer, enter `No` or `N`. In this case, the installation will be canceled.
2. After that, you need to view the terms of the Doctor Web License Agreement displayed on the screen. Press ENTER to scroll the text down one line or SPACEBAR to scroll down one screen.

   There are no options to scroll the License Agreement up.

3. Once you have read the License agreement, you are prompted to accept its terms. Enter `Yes` or `Y` if you accept the License agreement. If you refuse to accept it, enter `No` or `N`. In the latter case, the installer exits.
4. After you accept the terms of the License Agreement, the installation of the Dr.Web Industrial for Unix File Servers components automatically starts. During the procedure, the information about the installation process (installation log), including the list of installed components, is displayed on the screen.
5. If installation completes successfully, a message is displayed that informs you on how to manage Dr.Web Industrial for Unix File Servers operation.

If an error occurs, a message describing the error is displayed on the screen, and then the installer exits. When the installation process fails due to an error, resolve the issues that caused this error and start the installation again.

6.1.2. Installing from the Repository

Native packages of Dr.Web Industrial for Unix File Servers are stored in the Dr.Web official repository. Once you have added the Dr.Web repository to the list of those used by your operating system package manager, you can install the product from native packages the same way you install any other programs from the operating system repositories. Required dependencies are automatically resolved. Futhermore, in case of installing Dr.Web Industrial for Unix File Servers from the connected repository your OS package manager detects updates for all Dr.Web components and suggests installing them.

To access the Dr.Web repository, internet access is required.

All the commands mentioned below, which are used to add repositories, import digital signature keys, install and uninstall packages, must be run with superuser (usually the *root* user) privileges. To elevate your privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.

Dr.Web Industrial for Unix File Servers can be installed on FreeBSD only from a universal package.

Steps below are for the following OSes (package managers):

- Astra Linux, Debian, Mint, Ubuntu (apt);
- ALT (apt-rpm);
- Mageia, OpenMandriva Lx (urpmi);
- Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf);
- SUSE Linux (zypper).

Astra Linux, Debian, Mint, Ubuntu (apt)

1. The repository for these operating systems is protected with the digital signature of Doctor Web. To access the repository, import and add the digital signature key to the package manager storage by running the command:

```
# wget https://repo.drweb.com/drweb/drweb.gpg -O /etc/apt/trusted.gpg.d/drweb.gpg
```

The `apt-key` utility was used earlier to install the digital signature key from Doctor Web. This utility is deprecated and not recommended for usage. Furthermore, OS developers recommend using the `/etc/apt/trusted.gpg.d` directory instead of the `/etc/apt/trusted.gpg` file. For this reason, when you install Dr.Web Industrial for Unix File Servers from the repository, if the signature key is installed to the `/etc/apt/trusted.gpg` directory, the `apt-get update` command (see below) displays the following warning: `Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.` At that, the installation of Dr.Web Industrial for Unix File Servers continues as usual and the product functionality is not affected. To avoid this warning while installing or reinstalling Doctor Web products, install the signature key as indicated above.

2. To add the repository, run the command:

```
# echo "deb https://repo.drweb.com/drweb/debian 11.1 non-free" > /etc/apt/sources.list.d/drweb.list
```

You can complete steps 1 and 2 by downloading and installing a dedicated DEB package.

3. To install Dr.Web Industrial for Unix File Servers from the repository, run the commands:

```
# apt-get update
# apt-get install drweb-industrial-file-servers
```

You can also use alternative package managers (for example, Synaptic or aptitude) to install the product. Furthermore, it is recommended to use alternative managers, such as aptitude, to solve a package conflict if it occurs.

ALT (apt-rpm)

1. To add the repository, add the following line to the file `/etc/apt/sources.list.d/drweb-apt-rpm-<arch>.list`:

```
rpm http://repo.drweb.com/drweb/altlinux 11.1/<arch> drweb
```

where <*arch*> represents the package architecture in use:

- for the 32-bit version: `i386`;
- for the AMD64 architecture: `x86_64`;
- for the ARM64 architecture: `aarch64`;
- for the E2K architecture: `e2s`;
- for the IBM POWER (ppc64el) architecture: `ppc64le`.

2. Prior to installing Dr.Web Industrial for Unix File Servers on a computer of the E2K architecture running ALT, add the following lines to the `/etc/rpmrc` file:

```
arch_compat: e2kv4: e2s
arch_compat: e2k: e2s
arch_compat: e2s: noarch
```

3. To install Dr.Web Industrial for Unix File Servers from the repository, run the commands:

```
# apt-get update
# apt-get install drweb-industrial-file-servers
```

You can also use alternative package managers (for example, Synaptic or aptitude) to install the product.

Mageia, OpenMandriva Lx (urpmi)

1. Add the repository by running the command:

```
# urpmi.addmedia drweb https://repo.drweb.com/drweb/linux/11.1/<arch>/
```

where <*arch*> represents the package architecture in use:

- for the 32-bit version: `i386`;
- for the 64-bit version: `x86_64`.

2. To install Dr.Web Industrial for Unix File Servers from the repository, run the command:

```
# urpmi drweb-industrial-file-servers
```

You can also use alternative package managers (for example, rpmdrake) to install the product.

Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf)

1. Add the `drweb.repo` file with the content provided below to the `/etc/yum.repos.d` directory:

```
[drweb]
name=DrWeb - 11.1
baseurl=https://repo.drweb.com/drweb/linux/11.1/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://repo.drweb.com/drweb/drweb.key
```



If you plan to write the contents indicated above to a file by using such commands as `echo` with redirecting of an output, the `$` character must be escaped: `\$`.

You can complete step 1 by downloading and installing a dedicated RPM package.

2. To install Dr.Web Industrial for Unix File Servers from the repository, run the command:

```
# yum install drweb-industrial-file-servers
```

On Fedora 22 and later, it is recommended to use the `dnf` manager instead of the `yum` manager, for example:

```
# dnf install drweb-industrial-file-servers
```

You can also use alternative package managers (for example, PackageKit or Yumex) to install the product.

SUSE Linux (zypper)

1. To add the repository, run the command:

```
# zypper ar https://repo.drweb.com/drweb/linux/11.1/\$basearch/ drweb
```

2. To install Dr.Web Industrial for Unix File Servers from the repository, run the commands:

```
# zypper refresh
# zypper install drweb-industrial-file-servers
```

You can also use alternative package managers (for example, YaST) to install the product.

6.2. Upgrading Dr.Web Industrial for Unix File Servers

Dr.Web Industrial for Unix File Servers has two update modes:

1. Getting current updates for packages and components of the up-to-date Dr.Web Industrial for Unix File Servers version (usually such updates comprise bug fixes and minor improvements in component operation).
2. Upgrading to a newer version. This updating method is used if Doctor Web has released a new version of Dr.Web Industrial for Unix File Servers with new features.



Dr.Web Industrial for Unix File Servers allows to update virus databases and the anti-virus engine even if the protected server does not have access to the internet.

6.2.1. Getting Current Updates

After the installation of Dr.Web Industrial for Unix File Servers using any method described in the corresponding section, the package manager automatically connects to the Dr.Web package repository:

- If installation was performed from a universal package (a `.run` file) and the system uses DEB packages (for example, Astra Linux, Debian, Mint or Ubuntu), or the system does not have a package manager (FreeBSD), a separate version of the `zypper` package manager is used for operation with Dr.Web packages. It is automatically installed during Dr.Web Industrial for Unix File Servers installation.

  To get and install updated Dr.Web packages with this manager, go to the directory `/opt/drweb.com/bin` for GNU/Linux or `/usr/local/libexec/drweb.com/bin` for FreeBSD and run the commands:

```
# ./zypper refresh
# ./zypper update
```



On FreeBSD 11.x for the `amd64` architecture, a repository update error can occur while updating using the `zypper` manager. In this case, install the `compat10x-amd64` compatibility package and try again.

To install the package, run the command:

```
# pkg install compat10x-amd64
```

- In all other cases use updating commands of the package manager of your OS, for example:
  - on Red Hat Enterprise Linux or CentOS, use the `yum` command;
  - on Fedora, use the `yum` or `dnf` command;
  - on SUSE Linux, use the `zypper` command;
  - on Mageia or OpenMandriva Lx, use the `urpmi` command;

- on ALT, Astra Linux, Debian, Mint or Ubuntu, use the `apt-get` command.

You can also use alternative package managers developed for your operating system. If necessary, refer to the reference guide for your package manager.

If a new Dr.Web Industrial for Unix File Servers version is released, packages with its components are put into the section of the Dr.Web repository corresponding to the new product version. In this case, updating requires switching the package manager to the new Dr.Web repository section (refer to Upgrading to a Newer Version).

6.2.2. Upgrading to a Newer Version

In this section

- Introductory remarks
- Upgrading previous versions
  - Upgrading by installing a universal package
  - Upgrading from the repository
  - Key file transfer
  - Restoring connection to a centralized protection server

Introductory remarks

Before you upgrade to a new version, make sure that your server meets the system requirements of the new version.

The upgrade procedure for Dr.Web Industrial for Unix File Servers of earlier versions to a newer version is supported. Migrate to the newer version of Dr.Web Industrial for Unix File Servers in the same way the earlier version of Dr.Web Industrial for Unix File Servers was installed:

- If the current Dr.Web Industrial for Unix File Servers version was installed from the repository, an upgrade requires updating program packages from the repository.
- If the current Dr.Web Industrial for Unix File Servers version was installed from the universal package, to upgrade the product, you need to install another universal package that contains a newer version.

To determine how the version of Dr.Web Industrial for Unix File Servers to be updated was installed, check whether the Dr.Web Industrial for Unix File Servers executable directory contains the `uninst.sh` program uninstallation script. If so, the current version of Dr.Web Industrial for Unix File Servers was installed from the universal package; otherwise, it was installed from the repository.

Dr.Web Industrial for Unix File Servers can be installed on FreeBSD only from a universal package.

If you cannot update Dr.Web Industrial for Unix File Servers the same way you installed it initially, uninstall your current version, and then install a new version using any convenient method. Installation and uninstallation procedures for previous versions of Dr.Web Industrial for Unix File Servers are the same as installation and uninstallation methods described in the current manual. For additional information, refer to the Administrator manual for your current version of Dr.Web Industrial for Unix File Servers.

If the version of Dr.Web Industrial for Unix File Servers to be updated is controlled by a centralized protection server, it is recommended that you save the address of this server. For example, to determine the address of the centralized protection server to which Dr.Web Industrial for Unix File Servers is connected, run the command:

```
$ drweb-ctl appinfo
```

In the output provided by this command, from the line like

```
ESAgent; <PID>; RUNNING 1; Connected <address>, on-line
```

save the <*address*> part (which can look like `tcp://<IP address>:<port>`, for example, `tcp://10.20.30.40:1234`). In addition, it is recommended that you save the server certificate file.

In case of any difficulties with finding out the parameters of the current connection, refer to the Administrator Manual for the installed version of Dr.Web Industrial for Unix File Servers and to the administrator of your anti-virus network.

Upgrading previous versions

Upgrading by installing a universal package

Install the new version of Dr.Web Industrial for Unix File Servers from the universal package. If necessary, during the installation you will be prompted to automatically uninstall installed components of the older version of Dr.Web Industrial for Unix File Servers.

Upgrading from the repository

To upgrade your current version of Dr.Web Industrial for Unix File Servers installed from the repository of Doctor Web

Depending on the type of packages in use, do the following:

- **In case of using RPM packages (yum, dnf):**
  1. Change the repository in use (switch from the package repository of your current version to the package repository of the new version).



You can find the name of the repository storing the packages of the new version in the Installing from the Repository section. For details on how to switch between repositories, refer to help guides of your distribution.

2. Install the new version of Dr.Web Industrial for Unix File Servers from the repository by running the command:

```
# yum update
```

or, if the dnf manager is used (such as on Fedora 22 and later):

```
# dnf update
```



If an error has occurred while updating the packages, uninstall and reinstall Dr.Web Industrial for Unix File Servers. If necessary, see sections Uninstallation of Dr.Web Industrial for Unix File Servers Installed from the Repository and Installing from the Repository (paragraphs related to your OS and package manager).

- **In case of using DEB packages (apt-get)**:
  1. Change the repository in use (switch from the package repository of your current version to the package repository of the new version).
  2. Update the packages of Dr.Web Industrial for Unix File Servers by running the commands:

```
# apt-get update
# apt-get dist-upgrade
```

Key file transfer

Regardless of the selected method to upgrade Dr.Web Industrial for Unix File Servers, your current license key file will be automatically transferred to a proper location required by the new version of Dr.Web Industrial for Unix File Servers.



If any issue occurs during automatic installation of the key file, you can install it manually. The license key file of Dr.Web Industrial for Unix File Servers is stored in the directory `/etc/opt/drweb.com/` for GNU/Linux or `/usr/local/etc/drweb.com/` for FreeBSD. If the valid license key file is lost, contact the Doctor Web technical support.

Restoring connection to a centralized protection server

If possible, your connection to a centralized protection server will be restored automatically after the upgrade (if the version to be upgraded was connected to the centralized protection server). In case the connection has not been automatically restored, to reestablish the connection of the upgraded version of Dr.Web Industrial for Unix File Servers to the anti-virus network, run the command:

```
$ drweb-ctl esconnect <server address> --Certificate <path to server certificate file>
```

In case of any issues with the connection process, contact the administrator of your anti-virus network.

6.2.3. Updating Offline

In highly secure environments where an internet connection is blocked or limited, it is possible to update virus bases and the scanning engine offline. You need to download updates to a computer connected to the internet, copy them to a USB drive or a network drive and then install them on another computer, which is not connected to the internet. The update procedure must be run from the command line.

To get updates

1. Run the command on a computer connected to the internet:

   ```
   $ drweb-ctl update --Path <path to directory to store downloaded updates>
   ```

2. Copy the downloaded updates to a USB drive or a network drive.
3. Mount the network drive or the removable drive on the computer to be updated. If the updates are from the USB drive, run the commands:

   ```
   # mkdir /mnt/usb
   # mount <path to device> /mnt/usb
   ```

4. Install the updates by running the command:

   ```
   $ drweb-ctl update --From /mnt/usb
   ```

6.3. Uninstalling Dr.Web Industrial for Unix File Servers

Depending on the method that you used to install Dr.Web Industrial for Unix File Servers, you can uninstall the product using one of two methods:

- start the uninstaller to uninstall the universal package in command-line mode;
- uninstall the packages installed from the Doctor Web repository using a system package manager.



After uninstalling Dr.Web Industrial for Unix File Servers, you will have to manually remove a symbolic link to the Dr.Web VFS SMB module from the Samba server directory and to edit the Samba configuration file (`smb.conf`), having removed the `vfs objects = smb_spider` line from sections covering parameters of shared directories (where `smb_spider` is the name of the symbolic link to the Dr.Web VFS SMB module).

6.3.1. Uninstalling the Universal Package

Dr.Web Industrial for Unix File Servers that was installed from the universal package can be uninstalled from the command line (if you are using a graphical desktop environment, you will need a terminal emulator).



The uninstaller uninstalls not only Dr.Web Industrial for Unix File Servers, but also *all other* Dr.Web products installed on your computer.

If any other Dr.Web products, besides Dr.Web Industrial for Unix File Servers, are installed on your computer, follow a custom component installation and uninstallation procedure to uninstall Dr.Web Industrial for Unix File Servers only instead of starting the uninstaller.

Uninstalling Dr.Web Industrial for Unix File Servers from the command line

To run the uninstaller from the command line, run the command:

```
# /opt/drweb.com/bin/uninst.sh
```

for GNU/Linux or

```
# /usr/local/libexec/drweb.com/bin/uninst.sh
```

for FreeBSD. The uninstallation procedure of Dr.Web Industrial for Unix File Servers is described in the Uninstalling from the Command Line section.

To start the uninstaller in silent mode without displaying the user interface (including uninstaller dialogs in command-line mode), run the command:

```
# env DRWEB_NON_INTERACTIVE=yes /opt/drweb.com/bin/uninst.sh
```



On ALT 8 SP and OSes using outdated versions of the package manager, you may see the following messages in the process of uninstalling:

```
/etc/init.d/drweb-configd: No such file or directory
```

These messages do not affect the system functioning. The uninstallation procedure is being performed correctly.

6.3.1.1. Uninstalling from the Command Line

Once the command-line uninstaller runs, a prompt to uninstall the product is displayed on the command line.

1. To start uninstallation, enter `Yes` or `Y` in response to the "Do you want to continue?" request. To cancel uninstalling Dr.Web products from your computer, enter `No` or `N`. In this case, the uninstaller will exit.
2. An automatic uninstallation procedure of all installed Dr.Web packages will run after you confirm it. During this procedure, the information about the uninstallation progress will be displayed and logged.
3. Once the process is completed, the uninstaller will exit automatically.

6.3.2. Uninstallation of Dr.Web Industrial for Unix File Servers Installed from the Repository

Steps below are for the following OSes (package managers):

- Astra Linux, Debian, Mint, Ubuntu (apt);
- ALT (apt-rpm);
- Mageia, OpenMandriva Lx (urpme);
- Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf);
- SUSE Linux (zypper).



All commands mentioned below for package uninstallation require superuser (usually the *root* user) privileges. To elevate your privileges, use the `su` command (to change the current user) or the `sudo` command (to run a command as another user).

Astra Linux, Debian, Mint, Ubuntu (apt)

To uninstall the root meta-package of Dr.Web Industrial for Unix File Servers, run the command:

```
# apt-get remove drweb-industrial-file-servers
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall the root meta-package together with all dependencies, run the command:

```
# apt-get remove drweb-industrial-file-servers --autoremove
```

In this case, all packages (not only those of Dr.Web Industrial for Unix File Servers) that were automatically installed to resolve dependencies of other packages but are no longer used are uninstalled.

If the root meta-package was already uninstalled, to automatically uninstall all unused packages, run the command:

```
# apt-get autoremove
```

You can also use alternative package managers (for example, Synaptic or aptitude) to uninstall Dr.Web Industrial for Unix File Servers packages.

ALT (apt-rpm)

To uninstall Dr.Web Industrial for Unix File Servers, use the commands for Debian and Ubuntu OSes (see above).

You can also use alternative package managers (for example, Synaptic or aptitude) to uninstall Dr.Web Industrial for Unix File Servers packages.



On ALT 8 SP, the following messages can be displayed in the console during uninstallation:

```
/etc/init.d/drweb-configd: No such file or directory
```

These messages do not affect the functioning of the system. The uninstallation procedure is being performed correctly.

Mageia, OpenMandriva Lx (urpme)

To uninstall the root meta-package of Dr.Web Industrial for Unix File Servers, run the command:

```
# urpme drweb-industrial-file-servers
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall the root meta-package together with all dependencies, run the command:

```
# urpme --auto-orphans drweb-industrial-file-servers
```

In this case, all packages (not only those of Dr.Web Industrial for Unix File Servers) that were automatically installed to resolve dependencies of other packages but are no longer used are uninstalled.

You can also use alternative package managers (for example, `rpmdrake`) to uninstall Dr.Web Industrial for Unix File Servers packages.

Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf)

To uninstall all installed Dr.Web packages, run the command (the `*` character must be escaped on some OSes: `\*`):

```
# yum remove drweb*
```

On Fedora 22 and later, it is recommended to use the `dnf` manager instead of the `yum` manager, for example:

```
# dnf remove drweb*
```



These commands uninstall all packages that name starts with `drweb` (the standard name prefix of Dr.Web products). All packages with this prefix will be uninstalled, not only those of Dr.Web Industrial for Unix File Servers.

You can also use alternative package managers (for example, PackageKit or Yumex) to uninstall Dr.Web Industrial for Unix File Servers packages.

SUSE Linux (zypper)

To uninstall the root meta-package of Dr.Web Industrial for Unix File Servers, run the command:

```
# zypper remove drweb-industrial-file-servers
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall all installed Dr.Web packages, run the command (the `*` character must be escaped on some OSes: `\*`):

```
# zypper remove drweb*
```



This command uninstalls all packages whose name starts with `drweb` (the standard name prefix for Dr.Web products). All packages with this prefix will be uninstalled, not only those of Dr.Web Industrial for Unix File Servers.

You can also use alternative package managers (for example, YaST) to uninstall Dr.Web Industrial for Unix File Servers packages.

6.4. Additional Information

6.4.1. Dr.Web Industrial for Unix File Servers Packages and Files

In this section

- Packages
- Files

Packages

Dr.Web Industrial for Unix File Servers consists of the following packages:

Package	Contents
`drweb-bases`	Virus database files
`drweb-boost`	Boost libraries
`drweb-common`	The `drweb.ini` unified configuration file, main libraries, documentation, hierarchy of Dr.Web Industrial for Unix File Servers directories, and a tool for collecting information on product configuration and system environment. During the installation of this package, a user named `drweb` and a group named `drweb` are created
`drweb-configd`	Dr.Web ConfigD component files
`drweb-configure`	Files of an auxiliary tool for Dr.Web Industrial for Unix File Servers configuration
`drweb-ctl`	Dr.Web Ctl component files
`drweb-documentation`	Documentation files for Dr.Web for Unix products in the HTML format
`drweb-engine`	Dr.Web Virus-Finding Engine files
`drweb-esagent`	Dr.Web ES Agent component files

Package	Contents
drweb-filecheck	Dr.Web File Checker component files
drweb-industrial-file-servers-doc	Documentation in the PDF format
drweb-industrial-file-servers	Root meta-package of Dr.Web Industrial for Unix File Servers
drweb-httpd	Files of the Dr.Web HTTPD component and of the management web interface (a meta-package)
drweb-httpd-bin	Dr.Web HTTPD component files
drweb-httpd-webconsole	Files of the management web interface
drweb-icu	Libraries for Unicode support and internationalization
drweb-libs	Files of main libraries
drweb-netcheck	Dr.Web Network Checker component files
drweb-openssl	OpenSSL libraries
drweb-protobuf	Google Protobuf libraries
drweb-se	Dr.Web Scanning Engine component files
drweb-smbspider-daemon	Files of the SpIDer Guard for SMB component (SMB monitoring daemon)
drweb-smbspider	SpIDer Guard for SMB component files
drweb-smbspider-modules	SpIDer Guard for SMB component files (VFS SMB modules)
drweb-smbspider-modules-src	SpIDer Guard for SMB component files (VFS SMB module source code)
drweb-snmpd	Dr.Web SNMPD component files
drweb-spider	SpIDer Guard component files
drweb-update	Dr.Web Updater component files

The Custom Installation and Uninstallation of Components section describes typical sets of the components for custom installation that implement typical tasks of Dr.Web Industrial for Unix File Servers.

Files

Dr.Web Industrial for Unix File Servers files are stored in `/opt`, `/etc` and `/var` directories of the file system tree.

Structure of the directories in use:

Directory	Contents
• For GNU/Linux: `/etc/init.d/` • For FreeBSD: `/usr/local/etc/rc.d/`	The `drweb-configd` management script for the Dr.Web ConfigD configuration management daemon
• For GNU/Linux: `/etc/opt/drweb.com/` • For FreeBSD: `/usr/local/etc/drweb.com/`	The `drweb.ini` configuration file and the `drweb32.key` key file. In addition, it contains:
`certs/`	– files of certificates in use.
• For GNU/Linux: `/opt/drweb.com/` • For FreeBSD: `/usr/local/libexec/drweb.com/`	Main directory of Dr.Web Industrial for Unix File Servers. It contains:
`bin/`	– executable files of all components (except Dr.Web Virus-Finding Engine);
`include/`	– header files of libraries in use;
`lib/`	– libraries in use;
`man/`	– system help files: `man`;
`share/`	– auxiliary product files, including:
`doc/`	▫ Dr.Web Industrial for Unix File Servers documentation (License Agreement and Administrator manual, if the documentation packages are installed),
`drweb-bases/`	▫ files of Dr.Web virus databases (original files supplied during installation),
`scripts/`	▫ auxiliary script files.
• For GNU/Linux: `/var/opt/drweb.com/`	Auxiliary and temporary files, including:

Directory	Contents
• For FreeBSD: `/var/drweb.com/`	
`bases/`	– files of Dr.Web virus databases (the relevant updated version);
`cache/`	– cache of updates;
`drl/`	– lists of update servers in use;
`lib/`	– Dr.Web Virus-Finding Engine as a dynamic-link library (`drweb32.dll`) and the settings for operation in the centralized protection mode distribution, is also stored here;
`update/`	– directory for temporarily storing updates during their download.

6.4.2. Custom Installation and Uninstallation of Components

In this section

- Typical component sets for custom installation
- Installation and uninstallation of Dr.Web Industrial for Unix File Servers components installed from the repository
- Installation and uninstallation of Dr.Web Industrial for Unix File Servers components installed from the universal package
  - Unpacking the installation file
  - Custom installation of components
  - Custom uninstallation of components

If necessary, you can choose to install or uninstall only certain Dr.Web Industrial for Unix File Servers components by installing or uninstalling the respective packages. Perform custom installation or uninstallation the same way Dr.Web Industrial for Unix File Servers was installed.

To reinstall a component, you can uninstall it first and then install again.

Typical component sets for custom installation

If you need to install Dr.Web Industrial for Unix File Servers with limited functionality, you can install only component packages that provide the necessary functionality instead of installing a root meta-package from the repository or from the universal package. The packages required to resolve dependencies will be automatically installed. The table below displays component sets designed to resolve typical Dr.Web Industrial for Unix File Servers tasks. The **Packages for**

Installation column lists packages required for installation to obtain the specified component set.

Custom component set	Packages for installation	To be installed
Minimal set for console scanning	• `drweb-filecheck`, • `drweb-se`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web File Checker, • Dr.Web Scanning Engine, • Dr.Web Updater, • Virus databases
Set for GNU/Linux file system monitoring	• `drweb-se`, • `drweb-spider`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web File Checker, • Dr.Web Scanning Engine, • Dr.Web Updater, • SpIDer Guard, • Virus databases
Set for monitoring shared Samba directories	• `drweb-se`, • `drweb-smbspider`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web File Checker, • Dr.Web Scanning Engine, • Dr.Web Updater, • SpIDer Guard for SMB, • Virus databases

Installation and uninstallation of Dr.Web Industrial for Unix File Servers components installed from the repository

If Dr.Web Industrial for Unix File Servers is installed from a repository, to install or uninstall a component, use a respective command of the package manager of your OS, for example:

1. To remove the Dr.Web Network Checker component (the `drweb-netcheck` package) from Dr.Web Industrial for Unix File Servers installed on CentOS, run the command:

   ```
   # yum remove drweb-netcheck
   ```

2. To add the Dr.Web Network Checker component (the `drweb-netcheck` package) to Dr.Web Industrial for Unix File Servers installed on Ubuntu, run the command:

   ```
   # apt-get install drweb-netcheck
   ```

If necessary, use help on the package manager of your OS.

Installation and uninstallation of Dr.Web Industrial for Unix File Servers components installed from the universal package

If Dr.Web Industrial for Unix File Servers is installed from a universal package and you want to additionally install or reinstall a package of a component, you will need an installation file (with the `.run` extension) that was used to install Dr.Web Industrial for Unix File Servers. If you do not have this file anymore, download it from the Doctor Web company website.

Unpacking the installation file

When you start the `.run` file, you can specify the following command-line parameters:

`--noexec`—unpack Dr.Web Industrial for Unix File Servers installation files instead of starting the installation process;

`--keep`—do not delete Dr.Web Industrial for Unix File Servers installation files and the installation log after the installation;

`--target` <*directory*>—unpack Dr.Web Industrial for Unix File Servers installation files to the specified <*directory*>.

To get a full list of command-line parameters that can be used with the installation file, run the command:

```
$ ./<.run file name> --help
```

For custom installation of Dr.Web Industrial for Unix File Servers components, unpack the contents of the installation file. To do that, run the command:

```
$ ./<.run file name> --noexec --target <directory>
```

Custom installation of components

The `.run` installation file contains packages of all Dr.Web Industrial for Unix File Servers components (in the RPM format) and auxiliary files. Package files of each component are named as follows:

```
<component_name>_<version>~linux_<platform>.rpm
```

where <*version*> is a string that contains the version and date of a package release, and <*platform*> is a platform to run Dr.Web Industrial for Unix File Servers. Names of all Dr.Web Industrial for Unix File Servers component packages start with the `drweb` prefix.

The installation kit includes the `zypper` package manager intended for the installation of packages. For custom installation, use the `installpkg.sh` service script. To do that, firstly unpack the contents of the installation package to any writeable directory.



To install packages, superuser (usually the *root* user) privileges are required. To gain superuser privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.

To install a component package, go to the directory containing the unpacked installation file and run the command from the command line or from a terminal emulator:

```
# ./scripts/installpkg.sh <package_name>
```

For example:

```
# ./scripts/installpkg.sh drweb-netcheck
```

If it is necessary to install Dr.Web Industrial for Unix File Servers in full, start the installation script by running the command:

```
$ ./install.sh
```

Furthermore, you can install all Dr.Web Industrial for Unix File Servers packages (among other things, to install missing or accidentally removed components) by starting the installation of the root meta-package:

```
# ./scripts/installpkg.sh drweb-industrial-file-servers
```

Custom uninstallation of components

If your OS uses RPM packages, to uninstall a package of a component, run the corresponding uninstallation command of the package manager of your OS:

- on Red Hat Enterprise Linux and CentOS, run the `yum remove` <*package_name*> command;
- on Fedora, run the `yum remove` <*package_name*> or `dnf remove` <*package_name*> command;
- on SUSE Linux, run the `zypper remove` <*package_name*> command;
- on Mageia or OpenMandriva Lx, run the `urpme` <*package_name*> command;
- on ALT, run the `apt-get remove` <*package_name*> command.

For example, on Red Hat Enterprise Linux:

```
# yum remove drweb-netcheck
```

If your OS uses DEB packages, or there is no package manager in your system (FreeBSD), for the custom uninstallation, use the `zypper` package manager supplied with Dr.Web Industrial for Unix File Servers. To do that, go to the directory `/opt/drweb.com/bin` for GNU/Linux or `/usr/local/libexec/drweb.com/bin` for FreeBSD and run the command:

```
# ./zypper remove <package_name>
```

For example:

```
# ./zypper remove drweb-netcheck
```

If you want to uninstall Dr.Web Industrial for Unix File Servers, start the uninstallation script by running the command:

```
# ./uninst.sh
```

To reinstall a component, you can uninstall it first and then install again by starting custom or full installation.

6.5. Configuring Security Subsystems

Presence of the SELinux enhanced security subsystem in the OS as well as the use of mandatory access control systems, such as PARSEC—as opposed to the classical discretionary model used by Unix—causes issues in the operation of Dr.Web Industrial for Unix File Servers when its default settings are used. To ensure correct operation of Dr.Web Industrial for Unix File Servers in this case, it is necessary to make additional changes to the settings of the security subsystem and/or to the settings of Dr.Web Industrial for Unix File Servers.



Configuring the permissions of the PARSEC mandatory access control system for Dr.Web Industrial for Unix File Servers allows its components to bypass the restrictions of the set security policies and to get access to the files that belong to different privilege levels.

Even if you have not configured the permissions of the PARSEC mandatory access control system for Dr.Web Industrial for Unix File Servers, you still will be able to start file scanning directly from the command line. To do this, run the `drweb-ctl` command in standalone mode by indicating the `--Autonomous` parameter at startup.

In standalone mode, it will be possible to scan files that require a level of privileges not higher than the level of the user who started the scanning session. This mode has the following aspects:

- To start in standalone instance mode, you will need a valid key file, operation in centralized protection mode is not supported (it is possible to install the key file exported from a centralized protection server). In this case, even if Dr.Web Industrial for Unix File Servers is connected to the centralized protection server, the standalone instance *does not notify* the centralized protection server of the threats detected in standalone instance mode.
- All supplementary components that support the functioning of the standalone instance will be started on behalf of the current user and will work with a specifically generated configuration file.
- All temporary files and Unix sockets used for interaction of components are created only in a directory with an unique name. This directory is created by the started standalone instance in the directory for temporary files (specified by the `TMPDIR` environment variable).
- Paths to virus databases, anti-virus engine and executable files of service components are set to default values.
- The number of the standalone instances working simultaneously is not limited.
- When the standalone instance is shut down, the set of components maintaining it is also shut down.

6.5.1. Configuring SELinux Security Policies

In this section

- Universal package installation issues

- Dr.Web Industrial for Unix File Servers operation issues

If your GNU/Linux distribution features the SELinux (*Security-Enhanced Linux*) security subsystem, you may need to adjust SELinux security policies to enable correct operation of Dr.Web Industrial for Unix File Servers service components (such as the scanning engine) after their installation.

Universal package installation issues

If SELinux is enabled, the installation of the Dr.Web Industrial for Unix File Servers universal package from the installation file (`.run`) can fail because an attempt to create the *drweb* special user, as which Dr.Web Industrial for Unix File Servers components run, will be blocked.

If installation of Dr.Web Industrial for Unix File Servers from the installation file (`.run`) fails due to inability to create the *drweb* user, check the SELinux operation mode by running the `getenforce` command. The command outputs the current protection mode:

- `Enforcing`—protection is active and a restrictive strategy is used: actions that violate security policies are blocked and registered in the audit log;
- `Permissive`—protection is active but a permissive strategy is used: actions that violate the security policy are only registered in an audit log but not blocked;
- `Disabled`—SELinux is installed but not active.

If SELinux is operating in *Enforcing* mode, temporarily (during the installation of Dr.Web Industrial for Unix File Servers) change its mode to *Permissive*. For that purpose, run the command:

```
# setenforce 0
```

This command (until the next reboot) enables *Permissive* mode for SELinux.



Regardless of the operation mode enabled by running the `setenforce` command, after the restart of the operating system, SELinux returns to the safe operation mode specified in its settings (the file with SELinux settings is usually stored in the `/etc/selinux` directory).

After Dr.Web Industrial for Unix File Servers is successfully installed from the installation file, enable the *Enforcing* mode again before starting and activating the product. For that purpose, run the command:

```
# setenforce 1
```

Dr.Web Industrial for Unix File Servers operation issues

In certain cases when SELinux is running, several Dr.Web Industrial for Unix File Servers components (such as `drweb-se` and `drweb-filecheck`) cannot start, thereby hindering object scanning and file system monitoring. A failure to start these components causes the

occurrence of 119 and 120 error messages in the system log managed by the `syslog` service (this log is typically located in the `/var/log/` directory).

When the SELinux security system blocks access, such an event is also output to an audit system log. In general, when the audit daemon is used in the system, the audit log is stored in the `/var/log/audit/audit.log` file. Otherwise, messages about blocked operations are written to the general log file (`/var/log/messages` or `/var/log/syslog`).

If the scanning components of the product do not operate because they are blocked by SELinux, compile special *security policies* for them.



Certain GNU/Linux distributions do not feature the utilities mentioned below. If so, you may need to additionally install them.

To configure SELinux security policies

1. Create a new file with the SELinux policy source code (a `.te` file). This file defines restrictions related to the described module. The policy source code file can be created in one of the following ways:
   1) Using the `audit2allow` utility, which is the simplest method. The utility generates permissive rules from messages on access denial in system log files. You can set to search messages automatically or specify a path to the log file manually.



You can use this method only if Dr.Web Industrial for Unix File Servers components have violated SELinux security policies and these events have been registered in the audit system log. If not, wait for such an incident triggered by Dr.Web Industrial for Unix File Servers to occur or force-create permissive policies by using the `policygentool` utility (see below).

The `audit2allow` utility is contained either in the `policycoreutils-python` package or in the `policycoreutils-devel` package (for Red Hat Enterprise Linux, CentOS, Fedora operating systems, depending on the version) or in the `python-sepolgen` package (for Debian and Ubuntu operating systems).

Example of using `audit2allow`:

```
# grep drweb-se.real /var/log/audit/audit.log | audit2allow -M drweb-se
```

In the given example, the `audit2allow` utility performs a search in the `/var/log/audit/audit.log` file to find access denial messages for the `drweb-se` component.

The utility creates two files: the `drweb-se.te` policy source file and the `drweb-se.pp` policy module ready to install.

If no corresponding incidents are found in the system audit log, the utility returns an error message.

In most cases, you do not need to modify the policy file created by the `audit2allow` utility; thus, it is recommended to go to step 4 for the installation of the `drweb-se.pp` policy module.



The `audit2allow` utility outputs the `semodule` command with all arguments. By copying it to the command line and running it, you complete step 4. Go to step 2 only if you want to modify the security policies that were automatically generated for Dr.Web Industrial for Unix File Servers components.

2) Using the `policygentool` utility. For that purpose, specify the name of the component that you want to be treated differently and the full path to its executable file.



The `policygentool` utility included in the `selinux-policy` package for Red Hat Enterprise Linux and CentOS may not operate correctly. If so, use the `audit2allow` utility.

Example of policy creation using `policygentool`:

- for the `drweb-se` component:

```
# policygentool drweb-se /opt/drweb.com/bin/drweb-se.real
```

- for the `drweb-filecheck` component:

```
# policygentool drweb-filecheck /opt/drweb.com/bin/drweb-filecheck.real
```

You will be prompted to specify several general properties for created the domain. After that, three files that determine the policy will be created (for each of the components):

<module_name>`.te`, *<module_name>*`.fc` and *<module_name>*`.if`.

2. If required, edit the generated policy source file *<module_name>.te*, then use the `checkmodule` utility to create a binary representation (a `.mod` file) of this source file of the local policy.



This command requires the `checkpolicy` package to be installed in the system.

Usage example:

```
# checkmodule -M -m -o drweb-se.mod drweb-se.te
```

3. Create a policy module for installation (a `.pp` file) with the help of the `semodule_package` utility.

   Example:

```
# semodule_package -o drweb-se.pp -m drweb-se.mod
```

4. To install the created policy module, use the `semodule` utility.

   Example:

```
# semodule -i drweb-se.pp
```

For details on SELinux operating principles and configuration, refer to documentation on your GNU/Linux distribution.

6.5.2. Starting in CSE Mode (Astra Linux SE 1.6 and 1.7)

The Astra Linux SE OS supports a special *closed software environment* (CSE) mode. In this mode, applications can be started only if their executable files are signed with a digital signature of a developer whose public key is added to the OS list of trusted keys.

Starting with Astra Linux SE 1.6, the signing mechanism has been changed. You must configure Astra Linux SE 1.6 and 1.7 prior to starting Dr.Web Industrial for Unix File Servers in CSE mode.

To configure Astra Linux SE 1.6 and 1.7 to start Dr.Web Industrial for Unix File Servers in CSE mode

1. Install the `astra-digsig-oldkeys` package, if not installed, using an OS installation disk.
2. Add the public key of the Doctor Web company to the directory `/etc/digsig/keys/legacy/keys` (if the directory is absent, create it):

   ```
   # cp /opt/drweb.com/share/doc/digsig.gost.gpg /etc/digsig/keys/legacy/keys
   ```

3. Run the command:

   ```
   # update-initramfs -k all -u
   ```

4. Restart the operating system.

7. Getting Started

1. Activate Dr.Web Industrial for Unix File Servers.
2. Ensure its proper operation.
3. Integrate Dr.Web Industrial for Unix File Servers with the required file services (see the instruction manual for Samba).
4. If necessary, configure the monitoring parameters for GNU/Linux file system sections outside Samba shared directories.
5. Check what components are running and enable additional components, which are disabled by default, if you need them for the protection of your server (for example, SpIDer Guard, Dr.Web SNMPD).



You may also need to perform other actions apart from enabling the additional components, for example, you may need to adjust their default configuration.

To view the list of installed and running components and their settings, use one of the following:

- Dr.Web Ctl command-line management tool (use the `drweb-ctl appinfo`, `drweb-ctl cfshow` and `drweb-ctl cfset` commands);
- Dr.Web Industrial for Unix File Servers management web interface (by default, you can access it via a web browser at `https://127.0.0.1:4443`).

7.1. Registration and Activation

In this section

- Purchasing and registering license
- Dr.Web Industrial for Unix File Servers activation
  - Obtaining demo license
  - Key file installation
- Repeated registration

Purchasing and registering license

After a license is purchased, updates to product components and virus databases can be regularly downloaded from Doctor Web update servers. Moreover, standard technical support provided by Doctor Web and its partners becomes available.

You can purchase any Dr.Web product as well as obtain a product serial number either from our partners or our online store. For details on license options, visit the official website of the Doctor Web company.

License registration is required to prove that you are a legal user of Dr.Web Industrial for Unix File Servers and activate its anti-virus functions, including regular updates of virus databases. We recommend that you register the product and activate the license once the installation is completed.

Dr.Web Industrial for Unix File Servers activation

A license can be activated on the official website of the Doctor Web company.

To activate or renew the license, you need to enter the serial number. The serial number is supplied with Dr.Web Industrial for Unix File Servers or via email when purchasing or renewing the license online.

Obtaining demo license

A demo period for your copy of Dr.Web Industrial for Unix File Servers can be obtained on the official website of the Doctor Web company. After you select the product and fill the registration form, you will receive an email with a serial number or key file for demo period activation.



Another demo period of Dr.Web Industrial for Unix File Servers for the same computer can be obtained only after a certain time period.

Key file installation

If you have a key file corresponding to a valid license for the product (for example, you have obtained the key file from a vendor by email after registration or Dr.Web Industrial for Unix File Servers is being migrated to another server), you can activate Dr.Web Industrial for Unix File Servers by specifying a path to the key file. You can do that as follows:

- Manually. For that:
  1. Unpack the key file if archived.
  2. Next, do one of the following.
     - Copy the key file to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD and rename the file to `drweb32.key`.
     - In the Dr.Web Industrial for Unix File Servers configuration file specify the key file path as the `KeyPath` parameter value. In this case the key file name can be different from `drweb32.key`.
  3. Reload the Dr.Web Industrial for Unix File Servers configuration by running the command:

```
# drweb-ctl reload
```

  to apply changes.

Alternatively, run the command:

```
# drweb-ctl cfset Root.KeyPath <path to key file>
```

In the latter case, it is not required to additionally reload the Dr.Web Industrial for Unix File Servers configuration. The key file will not be copied to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD and will remain at its original location.



If the key file is not copied to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD, the user becomes responsible for ensuring that the file is protected from corruption or deletion. This installation method is not recommended as the key file can be accidentally deleted (for example, if the directory, where the key file is located, is automatically cleaned up by the system). Remember that if a key file is lost, you can request a new one, but the number of such requests is limited.

Repeated registration

If a key file is lost but the license is not expired, you must register again by providing the personal data you specified during the first registration. You can use a different email address. In this case, the license key file will be sent at the new address.

A license key file can be obtained using the license management command a limited number of times. If that amount has been exceeded, you can confirm the registration of your serial number on the website of the Doctor Web company to receive the key file. The key file is sent to the email that was specified during the first registration.

After the key file is delivered to you by email, you need to install it.

7.1.1. Key File

A key file, which is a special file stored on the local computer, corresponds to the purchased license or activated demo period for Dr.Web Industrial for Unix File Servers. The file regulates product usage rights in accordance with the purchased license or activated demo period.

The key file has `.key` extension and is valid if satisfies the following criteria:

- License or demo period is not expired.
- Demo period or license applies to all anti-virus components required by the product.
- Integrity of the key file is not violated.

If any of the conditions are violated, the license key file becomes invalid.



During Dr.Web Industrial for Unix File Servers operation, the key file must be located in the default directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD under the name `drweb32.key`.

Components of Dr.Web Industrial for Unix File Servers regularly check whether the key file is available and valid. The key file is digitally signed to prevent its editing. So, the edited key file becomes invalid. It is not recommended to open your key file in text editors in order to avoid its accidental corruption.

If no valid key file (license or demo) is found, or if the license is expired, operation of the anti-virus components is blocked until a valid key file is installed.

It is recommended that you keep the license key file until it expires. In this case, there is no need to register the serial number again upon reinstalling Dr.Web Industrial for Unix File Servers or installing it on another server, and you can use the license key file obtained during the initial registration process.



Dr.Web key files are usually packed in a `.zip` archive if sent via email. The archive with a key file to activate Dr.Web Industrial for Unix File Servers is usually named `agent.zip`. If there are *several* archives in the email message, you should use only `agent.zip`. Before installing the key file, you must unpack the archive using any suitable tool and extract the key file to any available directory (for example, to your home directory or to a USB flash drive).

7.2. Testing Product Operation

The *EICAR (European Institute for Computer Anti-Virus Research)* test helps testing operation of anti-virus programs that detect viruses using signatures. This test was designed specifically so that users could test reaction of an installed anti-virus to a threat without putting their computers at risk.

Although the *EICAR* test program is not actually malware, it is treated by the majority of anti-viruses as a virus. Dr.Web anti-virus products report the following upon detection of this “virus”: *EICAR Test File (NOT a Virus!)*. Other anti-viruses alert users in a similar way. The EICAR test program is a 68-byte `.com` file for MS-DOS/Windows that outputs the following message to the console or to a terminal emulator screen when running:

```
EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
```

The test program body contains only text characters that form the following string:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

If you create a text file consisting of the string provided above, the resulting file will be the “virus” program.

If Dr.Web Industrial for Unix File Servers operates correctly, this file must be detected during a file system scan regardless of the scan type and the user must be notified of the detected threat: EICAR Test File (NOT a Virus!).

An example of a command to test operation of Dr.Web Industrial for Unix File Servers using the EICAR test program:

```
$ echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' > testfile
&& drweb-ctl scan testfile && rm testfile
```

This command writes the string that represents the body of the EICAR test program to a file named `testfile` created in the current directory, scans the resulting file and removes this file afterwards.



The abovementioned test requires write access to the current directory. In addition, make sure that it does not contain a file named `testfile` (if necessary, change the file name in the command).

If the test "virus" is detected, the following message is displayed:

```
<path to the current directory>/testfile - infected with EICAR Test File (NOT a Virus!)
```

If an error occurs during the test, refer to the description of known errors.



If the SplDer Guard file monitor is enabled, the file can be immediately deleted or quarantined upon detection of the threat (depending on component settings). In this case, after the threat notification is displayed, the `rm` command will inform that the file is missing, which implies that the monitor operates correctly.

7.3. Configuring File System Monitoring

In this section

- Main file monitoring settings
- Switching between file monitoring modes

To configure the GNU/Linux file system monitoring performed by the SplDer Guard monitor, specify the values of the parameters provided in the `[LinuxSpider]` settings section of the configuration file.

Main file monitoring settings

- Enable the monitor by setting the `Start` parameter value to `Yes`.
- Specify the mode of interaction of the monitor with the file system in the `Mode` parameter (it is recommended that you use the `Auto` value).

- If necessary, use the `ExcludedProc` parameter to store executable paths for trusted applications, that is, the applications which access to files will not be controlled by the monitor.
- If necessary, use the `ExcludedFilesystem` parameter to store the names of the file systems (for example, `cifs`) whose files will not be controlled by the monitor.
- Specify the monitoring scope by indicating a set of protected spaces. Each protected space is specified by the separate section `[LinuxSpider.Space.<space name>]`. Specify a path to the directory with the files to be monitored in the `Path` parameter for each space, and set the `Enable` value to `Yes` to include the protected space in the monitoring scope.
- Specify the exclusion scope (the lists of paths to objects that are monitored and excluded from monitoring) in the `ExcludedPath` parameter (for the entire file system or for each protected space). For example, if some paths are controlled by the Samba file server, these paths should be included in the exclusion scope in order to avoid conflicts if the scanning is performed by different monitors.
- Specify the parameters of file scanning and the reaction of the monitor to detection of various types of threats (if necessary, specify them individually for each protected space within the monitoring scope).

Switching between file monitoring modes



The modes for enhanced monitoring of files and pre-blocking are only available if SpIDer Guard operates in `FANOTIFY` mode and the OS kernel is built with the `CONFIG_FANOTIFY_ACCESS_PERMISSIONS` option enabled.

To switch between SpIDer Guard monitoring modes, superuser privileges are required. To gain superuser privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.

- To switch SpIDer Guard to the `FANOTIFY` mode, run the command:

```
# drweb-ctl cfset LinuxSpider.Mode FANOTIFY
```

- To change the monitoring mode, run the command:

```
# drweb-ctl cfset LinuxSpider.BlockBeforeScan <mode>
```

where <*mode*> defines the blocking mode:

- `Off`—access is not blocked, SpIDer Guard operates in regular (non-blocking) monitoring mode;
- `Executables`—access to executable files is blocked, SpIDer Guard performs enhanced monitoring of executable files;
- `All`—access to all files is blocked, SpIDer Guard monitors files in “paranoid” mode.

- To change the interval within which scan results cached by Dr.Web File Checker remain relevant, run the command:

```
# drweb-ctl cfset FileCheck.RescanInterval <interval>
```

where <*interval*> determines the interval during which cached scan results remain relevant. The allowed value is from `0s` to `1m`. If you set the interval of less than 1 second, the scanning is performed upon each request.

After the settings are adjusted, reload the Dr.Web Industrial for Unix File Servers configuration by running the command:

```
# drweb-ctl reload
```

You can also restart Dr.Web Industrial for Unix File Servers by restarting the Dr.Web ConfigD configuration management daemon by running the command:

```
# service drweb-configd restart
```

7.4. Integration with a Samba File Server

In this section

- Steps for integration with Samba
- Configuration tool



The SpIDer Guard for SMB monitor uses a custom VFS SMB module to integrate with Samba. Several versions of the VFS SMB module built for different versions of Samba are supplied with the SpIDer Guard for SMB component; however, they may be incompatible with the version of Samba installed on your file server, for example, if your Samba server uses the `CLUSTER_SUPPORT` option.

If VFS SMB modules are incompatible with your Samba server, the *corresponding message* is shown during the Dr.Web Industrial for Unix File Servers installation. In this case, build the VFS SMB module for your Samba server manually (with the `CLUSTER_SUPPORT` option if necessary).

The procedure of building the VFS SMB module from source code is described in the Building the VFS SMB Module section.

Steps for integration with Samba

To integrate SpIDer Guard for SMB with the Samba file server, do the following:

1. In the directory from which Samba loads its VFS modules (the default directory in GNU/Linux is `/usr/lib/<architecture>-linux-gnu/samba/`), create the `smb_spider.so` symbolic link that points to the Doctor Web-supplied VFS SMB module that corresponds to your version of the Samba server.

   The VFS SMB modules supplied by the Doctor Web company are stored in the directory with libraries `/opt/drweb.com/lib/<architecture>-linux-gnu/samba/` (in Debian, Ubuntu, Mint) or `/opt/drweb.com/lib64/samba/`.

   The files of modules are named as follows: `libsmb_spider.so.<ver>`, where <*ver*> is the version of the Samba server for which the module is intended.

For example, `/opt/drweb.com/lib/x86_64-linux-gnu/samba/libsmb_spider.so.4.13.0` is a VFS SMB module for the Samba server version 4.13.0 for GNU/Linux, `x86_64` architecture.

2. Create sections for shared directories in the `smb.conf` configuration file of the Samba server (by default, stored on GNU/Linux in the `/etc/samba` directory). Such section should be drafted as follows:

```
[<resource name>]
comment = <comment>
path = <path to protected directory>
vfs objects = smb_spider
writeable = yes
browseable = yes
guest ok = yes
public = yes
```

where the <*resource name*> is any name of the shared resource, and <*comment*> is an arbitrary string with a comment (optional). The object name specified in the `vfs objects` parameter must match the file name of the symbolic link (in this case, `smb_spider`).

After that, the directory specified in the `path` parameter, will be monitored by the SpIDer Guard for SMB monitor. At the same time, SpIDer Guard for SMB and the VFS SMB module will interact via a Unix socket `/`<*samba chroot path*>`/var/run/.com.drweb.smb_spider_vfs`. By default, the path to this Unix socket is specified in the SpIDer Guard for SMB settings and in VFS SMB module settings.

You can connect SpIDer Guard for SMB to the shared directories customized in the Samba server configuration file using the `drweb-configure` configuration tool.

3. If you need to change the path to the socket, specify the new path both in the settings of SpIDer Guard for SMB (the `SmbSocketPath` parameter) and in the `smb.conf` Samba configuration file. For that, add the following line to the `[`<*resource name*>`]` section:

```
smb_spider:socket = <path to socket>
```

where <*path to socket*> must be an absolute path to the Unix socket, relative to the root directory that was set for the Samba server with `chroot` (<*samba chroot path*>).

4. If required, you can exclude objects in protected shared directories (both directories and separate files can be excluded). For that, specify paths to these objects in the `ExcludedPath` parameter. Specify paths to the objects that must be scanned in the `IncludedPath` parameter.



The `IncludedPath` parameter takes precedence over the `ExcludedPath` parameter, that is, if the same file or directory is included in both parameter values, this object will be scanned.

5. If you need to specify custom scanning settings for this shared directory different from default settings (for all modules), create a tag identifier for the VFS SMB module that controls this directory:

```
smb_spider:tag = <resource name>
```

Then specify custom settings for controlling this shared directory in the SpIDer Guard for SMB settings as a separate section `[SMBSpider.Share.`<*resource name*>`]`.

To add a new parameter section identified by the <*resource name*> tag with the help of the Dr.Web Ctl command-line tool, run the command: `drweb-ctl cfset SmbSpider.Share.<resource name>.<parameter> <value>`, for example:

```
# drweb-ctl cfset SmbSpider.Share.AccountingFiles.OnAdware Quarantine
```

This command adds the `[SMBSpider.Share.AccountingFiles]` section into the configuration file. This section will contain all parameters for scanning the directory, at that, values for all parameters, except the `OnAdware` parameter specified in the command, will coincide with parameter values from the general `[SMBSpider]` section.

6. Enable SpIDer Guard for SMB by setting the `Start` parameter value to `Yes`.

After the settings are adjusted, restart the Samba server and also reload the Dr.Web Industrial for Unix File Servers configuration by running the command:

```
# drweb-ctl reload
```

You can also restart Dr.Web Industrial for Unix File Servers by restarting the Dr.Web ConfigD configuration management daemon by running the command:

```
# service drweb-configd restart
```



To avoid potential conflicts between SpIDer Guard for SMB and SpIDer Guard, which may occur in the process of scanning files in Samba shared directories, it is recommended that you additionally configure SpIDer Guard by performing one of the following actions:

- exclude the Samba shared directories (specify them in the `ExcludedPath` parameter);
- add the Samba process (`smbd`) to the list of ignored processes (specify `smbd` in the `ExcludedProc` parameter).

Configuration tool

For ease of integration of SpIDer Guard for SMB with shared directories (connecting and disconnecting them) customized in the Samba file server configuration file, a custom tool named `drweb-configure` was designed. To configure the way SpIDer Guard for SMB connects to or disconnects from existing shared directories, run the command:

```
# drweb-configure samba [<parameters>]
```

You can specify the following parameters:

- `+`<*Samba resource*>—name of Samba shared resource (as it is specified in the `smb.conf` configuration file) to be protected by SpIDer Guard for SMB;
- `-`<*Samba resource*>—name of Samba shared resource (as it is specified in the `smb.conf` configuration file) to be excluded from SpIDer Guard for SMB protection;
- `+/all`—protect all shared Samba resources specified in the `smb.conf` configuration file by SpIDer Guard for SMB;
- `-/all`—exclude all shared Samba resources specified in the `smb.conf` configuration file from SpIDer Guard for SMB protection;

- `add_symlink`—create the `smb_spider.so` symbolic link pointing to the VFS SMB Dr.Web module (the path to the source file may differ depending on the version of Samba being used);
- `remove_symlink`—remove the `smb_spider.so` symbolic link;
- <*configuration file*>—path to the Samba file server configuration file (`smb.conf`) to be processed. If this argument is skipped, the `drweb-configure` tool will attempt to locate the relevant `smb.conf` file.



To access help documentation on integrating Samba shared directories in SpIDer Guard for SMB, run the command:

```
$ drweb-configure --help samba
```

8. Brief Instructions

In this section

- Working with file storage:
  - How to configure GNU/Linux file system monitoring
  - How to connect Dr.Web Industrial for Unix File Servers to a Samba server
  - How to add a new SMB shared directory General operation of Dr.Web Industrial for Unix File Servers:
  - How to restart Dr.Web Industrial for Unix File Servers
  - How to connect to a centralized protection server
  - How to disconnect from a centralized protection server
  - How to activate Dr.Web Industrial for Unix File Servers
  - How to upgrade Dr.Web Industrial for Unix File Servers
  - How to add or remove a component of Dr.Web Industrial for Unix File Servers
  - How to manage Dr.Web Industrial for Unix File Servers component operation
  - How to view the log of Dr.Web Industrial for Unix File Servers

How to connect Dr.Web Industrial for Unix File Servers to a Samba server

Follow the instructions provided in the Integration with a Samba File Server section.

How to configure GNU/Linux file system monitoring

Follow the instructions provided in the Configuring File System Monitoring section.

How to restart Dr.Web Industrial for Unix File Servers

To restart already running Dr.Web Industrial for Unix File Servers, you can use the script that controls the Dr.Web ConfigD configuration management daemon. Starting, stopping, or restarting the daemon will respectively start, stop or restart Dr.Web Industrial for Unix File Servers.

The script that controls the operation of Dr.Web ConfigD is stored in the standard OS directory (`/etc/init.d/` for GNU/Linux or `/usr/local/etc/rc.d/` for FreeBSD) and is named `drweb-configd`. The script has the following parameters:

Parameter	Description
`start`	Start Dr.Web ConfigD if it is not running. Upon starting, Dr.Web ConfigD starts all the required components of Dr.Web Industrial for Unix File Servers.
`stop`	Shut down Dr.Web ConfigD if it is running. Upon shutting down, Dr.Web ConfigD shuts down all the components of Dr.Web Industrial for Unix File Servers.
`restart`	Restart (shut down and start) Dr.Web ConfigD that will shut down and start all Dr.Web Industrial for Unix File Servers components. If Dr.Web ConfigD is not running, the parameter has the same effect as `start`.
`condrestart`	Restart Dr.Web ConfigD only if it is running.
`reload`	Send a HUP signal to Dr.Web ConfigD if it is running. Dr.Web ConfigD forwards this signal to all the components of Dr.Web Industrial for Unix File Servers. The parameter is used to make all Dr.Web Industrial for Unix File Servers components reread their configuration.
`status`	Output the current state of Dr.Web ConfigD to the console.

For example, to restart Dr.Web Industrial for Unix File Servers (or start it, if it is not running) on an OS of the GNU/Linux family, run the command:

```
# /etc/init.d/drweb-configd restart
```

How to connect to a centralized protection server

1. Obtain a centralized protection server address and certificate file from your anti-virus network administrator. You may also need additional parameters such as an identifier and a password for your station or identifiers of the main group and the tariff group.
2. Use the `esconnect` command of the Dr.Web Ctl command-line tool included in Dr.Web Industrial for Unix File Servers.

   To establish a connection, you must use the `--Certificate` parameter by specifying a path to the certificate file of the server. You can additionally enter an identifier of your host ("station" in the terms of the centralized protection server) and a password for authentication on the server, if you know them, by using the `--Login` and `--Password` parameters. If these parameters are specified, connection to the server will be established only if you specify a correct identifier-password pair. If the parameters are not specified, connection to the server will be established only if it is approved for the server (automatically or by the administrator of the anti-virus network, depending on the server settings).

   Moreover, you can use the `--Newbie` parameter (connect as a new user). If this mode is allowed on the server, after this connection is approved, the server automatically generates

a unique identifier-password pair for this host, which will be further used to connect it to the server.

> In this mode the centralized protection server generates a new account for the host even if this host already has another account on the server.

A standard example of the command to connect Dr.Web Industrial for Unix File Servers to the centralized protection server:

```
# drweb-ctl esconnect <server address> --Certificate <path to the server certificate file>
```

After establishing a connection to the centralized protection server, Dr.Web Industrial for Unix File Servers will operate in the centralized protection mode or in the mobile mode, depending on permissions set on the server and the value of the `MobileMode` configuration parameter of the Dr.Web ES Agent component. To force the mobile mode, set this parameter to `On`. For operation in the centralized protection mode, set the parameter to `Off`.

A standard example of the command to switch Dr.Web Industrial for Unix File Servers, which is connected to the centralized protection server, to the mobile mode is as follows:

```
# drweb-ctl cfset ESAgent.MobileMode On
```

> If the centralized protection server in use does not support or does not allow the mobile mode, changing the `MobileMode` parameter value will not switch Dr.Web Industrial for Unix File Servers to the mobile mode.

How to disconnect from a centralized protection server

To disconnect Dr.Web Industrial for Unix File Servers from the centralized protection server and switch to the standalone mode, use the `esdisconnect` command of the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line:

```
# drweb-ctl esdisconnect
```

To use Dr.Web Industrial for Unix File Servers in standalone mode, a valid license key file is required. Otherwise, anti-virus functions of Dr.Web Industrial for Unix File Servers *will be blocked* after switching to the standalone mode.

How to activate Dr.Web Industrial for Unix File Servers

1. Register on the website of the Doctor Web company.
2. You will receive an archive containing a valid license key file at the email address that you specified during the registration (you can also download this archive directly from the website after you have finished the registration).
3. Install the key file.

How to add a new SMB shared directory

1. Edit the Samba server configuration file (`smb.conf`) by adding a section describing the shared directory. The section describing the shared directory must be as follows:

```
[<resource name>]
comment = <comment>
path = <path to the protected directory>
vfs objects = smb_spider
writeable = yes
browseable = yes
guest ok = yes
public = yes
```

   where <*resource name*> is any name of the shared resource and <*comment*> is an optional arbitrary comment string.

2. If you need to specify scanning settings for the added shared directory that differ from the default SpIDer Guard for SMB settings, use steps 3 and 4 of the instruction given in the Integration with a Samba File Server section.
3. Restart the Samba server and Dr.Web Industrial for Unix File Servers.

How to upgrade Dr.Web Industrial for Unix File Servers

Update component versions or upgrade to a new version.



You may be asked to remove the current version of Dr.Web Industrial for Unix File Servers.

How to add or remove a component of Dr.Web Industrial for Unix File Servers

Follow the custom component installation and uninstallation procedure.



When installing and uninstalling a component, other Dr.Web Industrial for Unix File Servers components can be additionally installed or uninstalled to resolve dependencies.

How to manage Dr.Web Industrial for Unix File Servers component operation

To view the status of Dr.Web Industrial for Unix File Servers components and to manage their operation, use:

- The Dr.Web Ctl command-line management tool. Use the `drweb-ctl appinfo`, `drweb-ctl cfshow` and `drweb-ctl cfset` commands. To view the list of available management commands, run the `drweb-ctl --help` command.

- Dr.Web Industrial for Unix File Servers management web interface. By default, you can access it via a web browser at `https://127.0.0.1:4443`.

How to view the log of Dr.Web Industrial for Unix File Servers

By default, the unified log of all Dr.Web Industrial for Unix File Servers components is output to `syslog` (a file used by the system component `syslog` to log messages; the file depends on the system and is located in the `/var/log` directory). Unified log settings are defined in the configuration file in the `[Root]` section (`Log` and `DefaultLogLevel` parameters). `Log` and `LogLevel` parameters are provided for each component in its settings section. They set the log storage location and the logging level of messages output by the component to the log.

Alternatively, run the `drweb-ctl log` command.

To change the logging settings, use the Dr.Web Ctl command-line management tool or the Dr.Web Industrial for Unix File Servers management web interface.

To identify errors, configure the output of the unified log of all components to be stored in a separate file and enable the output of detailed debug information. For that, run the commands:

```
# drweb-ctl cfset Root.Log <log path>
# drweb-ctl cfset Root.DefaultLogLevel DEBUG
```

To reset the unified log settings for all components, run the commands:

```
# drweb-ctl cfset Root.Log -r
# drweb-ctl cfset Root.DefaultLogLevel -r
```

9. Dr.Web Industrial for Unix File Servers Components

This section contains a description of the Dr.Web Industrial for Unix File Servers components. For each of them, you can find information about its functions, operation principles and parameters stored in the configuration file.

9.1. Dr.Web ConfigD

The Dr.Web ConfigD configuration management daemon is the main control component of Dr.Web Industrial for Unix File Servers. It provides centralized storage for settings of all Dr.Web Industrial for Unix File Servers components, manages the operation of all components and organizes trusted data exchange among them.

Dr.Web ConfigD performs the following functions:

- starting and stopping the components of Dr.Web Industrial for Unix File Servers depending on settings;
- restarting the components automatically in case of failures;
- starting the components upon the request of other components;
- informing the components of changed settings;
- enabling the centralized management of configuration parameters;
- passing the information from the key file in use to the components;
- receiving the license information from the components;
- receiving the new license information from the specialized components;
- informing the running components of license changes.

9.1.1. Operating Principles

The Dr.Web ConfigD configuration management daemon always runs with superuser privileges (*root*). It starts other Dr.Web Industrial for Unix File Servers components and communicates with them via a preliminarily open socket. The configuration daemon accepts connections from other Dr.Web Industrial for Unix File Servers components via an information socket (publicly accessible) and an administrative socket (accessible only to the components running with superuser privileges). The daemon loads configuration parameters and license information from files or receives this information from a centralized protection server via Dr.Web ES Agent and sets valid default values of the configuration parameters. By the time any component starts or receives the SIGHUP signal, the configuration management daemon has a comprehensive and consistent set of configuration parameters for all Dr.Web Industrial for Unix File Servers components.

Upon receiving the SIGHUP signal, Dr.Web ConfigD reloads the configuration parameters and license information. If required, the daemon also instructs all components to reload their configuration parameters.

Upon receiving the SIGTERM signal, Dr.Web ConfigD shuts down all components and then finishes its own operation. Dr.Web ConfigD removes all temporary files of the components after they are shut down.

Principles of interaction with other components

1. All components receive the configuration parameters and license information from Dr.Web ConfigD at startup. Only these settings are used by the components in their further operation.
2. Dr.Web ConfigD forwards messages from all components started with it to a unified log. All messages output to *stderr* by the components are gathered by Dr.Web ConfigD and stored in the unified log of Dr.Web Industrial for Unix File Servers with a mark indicating the component that reported an error and time of its occurrence.
3. Upon shutting down, all controlled components return an exit code. If the code differs from `101`, `102` and `103`, the component will be restarted and the corresponding message from *stderr* will be output to the Dr.Web Industrial for Unix File Servers log.
   - Code 101 is returned when the component cannot operate with the current license. The component will be restarted only after the license parameters are modified.
   - Code 102 is returned when the component cannot operate with the current configuration parameters. If some configuration parameters were modified, Dr.Web ConfigD will try to restart the component.
   - Code `103` is returned in case the components started by Dr.Web ConfigD upon request (Dr.Web Scanning Engine and Dr.Web File Checker) have been idle for a long time. The timeout after which the component is shut down with error code `103` is specified in the settings of the corresponding component (the `IdleTimeLimit` parameter).
   - If new configuration parameters received from Dr.Web ConfigD by the component cannot be applied on the fly, the component exits with code `0` so that Dr.Web ConfigD can restart it.
   - If the component cannot connect to Dr.Web ConfigD or a communication protocol error occurs, the component outputs a corresponding message to *stderr* and exits with code `1`.
4. Signal exchange is maintained.
   - Dr.Web ConfigD sends the SIGHUP signal to the component instructing it to apply the modified configuration parameters.
   - Dr.Web ConfigD sends the SIGTERM signal to the component instructing it to shut down. After receiving the signal, the component should shut down within 30 seconds.
   - If the component does not shut down within 30 seconds, Dr.Web ConfigD sends the SIGKILL signal to forcibly shut down the component.

9.1.2. Command-Line Arguments

To start the Dr.Web ConfigD configuration management daemon from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-configd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-configd [<parameters>]
```

Dr.Web ConfigD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None
`--config`	Function: Use the specified configuration file for further operation. Short form: `-c` Arguments: <*path to file*>—path to the configuration file in use.
`--daemonize`	Function: Run the component as a daemon; that is, without access to the terminal. Short form: `-d` Arguments: None
`--pid-file`	Function: Use the specified PID file for further operation. Short form: `-p` Arguments: <*path to file*>—path to the file to store the process identifier (PID).

Example:

```
$ /opt/drweb.com/bin/drweb-configd -d -c /etc/opt/drweb.com/drweb.ini
```

This command runs Dr.Web ConfigD as a daemon, which forces the component to use the configuration file `/etc/opt/drweb.com/drweb.ini`.

Startup notes

To enable the operation of Dr.Web Industrial for Unix File Servers, the component must run as a daemon. Under normal conditions, Dr.Web ConfigD starts automatically at the startup of the

operating system; for this purpose the component is supplied with the `drweb-configd` management script located in a system directory (`/etc/init.d` for GNU/Linux or `/usr/local/etc/rc.d` for FreeBSD). To manage the operation of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line (the tool is started by running the `drweb-ctl` command).



To get documentation on this component from the command line, run the `man 1 drweb-configd` command.

9.1.3. Configuration Parameters

The Dr.Web ConfigD configuration management daemon uses configuration parameters specified in the `[Root]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the Dr.Web ConfigD component. Default value: `Notice`
`Log` *{log type}*	Logging method of the configuration daemon and of those components for which another value of this parameter is not specified.  Upon its initial startup, before the configuration file is read, the configuration daemon uses the following values of the parameter: • as a daemon (if run with the `-d` option)—`SYSLOG:Daemon`; • `Stderr` otherwise. If a component is operating in the background (was started with the `-d` option from the command line), the `Stderr` value *cannot* be used for this parameter. Default value: `SYSLOG:Daemon`
`PublicSocketPath` *{path to file}*	Path to the socket used for interaction by all Dr.Web Industrial for Unix File Servers components. Default value: `/var/run/.com.drweb.public`
`AdminSocketPath` *{path to file}*	Path to the socket used for interaction by Dr.Web Industrial for Unix File Servers components with elevated (administrative) privileges. Default value: `/var/run/.com.drweb.admin`

Parameter	Description
`DebugIpc` *{boolean}*	Log or do not log detailed IPC messages at the debug level (with `LogLevel = DEBUG`). These messages reflect interaction of the configuration management daemon with other components. Default value: `No`
`UseMesh` *{boolean}*	Use or do not use the Dr.Web MeshD component for scanning. If this parameter is disabled, Dr.Web Scanning Engine is used for scanning instead of Dr.Web MeshD. Allowed values: • `On`, `Yes`, `True`—use Dr.Web MeshD; • `Off`, `No`, `False`—do not use Dr.Web MeshD. If the Dr.Web MeshD component is not installed, the Dr.Web Scanning Engine component is used irrespective of the set value. If none of the components is installed, the scan ends with error x119. Default value: `No`
`UseVxcube` *{boolean}*	Use or do not use Dr.Web vxCube to analyze email attachments as an external filter connected to the MTA. Default value: `No`
`KeyPath` *{path to file}*	Path to the key file (license or demo). Default value: • for GNU/Linux: `/etc/opt/drweb.com/drweb32.key` • for FreeBSD: `/usr/local/etc/drweb.com/drweb32.key`
`CoreEnginePath` *{path to file}*	Path to the dynamic library of Dr.Web Virus-Finding Engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/lib/drweb32.dll` • for FreeBSD: `/var/drweb.com/lib/drweb32.dll`
`VirusBaseDir` *{path to directory}*	Path to the directory with virus database files. Default value: • for GNU/Linux: `/var/opt/drweb.com/bases` • for FreeBSD: `/var/drweb.com/bases`
`VersionDir` *{path to directory}*	Path to a directory, where the information about the current versions of Dr.Web Industrial for Unix File Servers components is stored. Default value: • for GNU/Linux: `/var/opt/drweb.com/version` • for FreeBSD: `/var/drweb.com/version`
`CacheDir` *{path to directory}*	Path to the cache directory (being used to store cache for updates as well as cache for information about scanned files).

Parameter	Description
	Default value: • for GNU/Linux: `/var/opt/drweb.com/cache` • for FreeBSD: `/var/drweb.com/cache`
`TempDir` *{path to directory}*	Path to the directory with temporary files. Default value: *Path from the system environment variable* `TMPDIR`, `TMP`, `TEMP` or `TEMPDIR` (the environment variables are searched in this particular order). Otherwise `/tmp`, if none of them was found.
`RunDir` *{path to directory}*	Path to the directory with all PID files of running components and sockets used for interaction by Dr.Web Industrial for Unix File Servers components. Default value: `/var/run`
`VarLibDir` *{path to directory}*	Path to the directory with libraries used by Dr.Web Industrial for Unix File Servers components. Default value: • for GNU/Linux: `/var/opt/drweb.com/lib` • for FreeBSD: `/var/drweb.com/lib`
`AdminGroup` *{group name | GID}*	Group of users with administrative privileges for Dr.Web Industrial for Unix File Servers management. These users, in addition to the superuser (the *root* user), are allowed to elevate privileges of Dr.Web Industrial for Unix File Servers components to superuser privileges. Default value: *Defined automatically* during Dr.Web Industrial for Unix File Servers installation
`VersionNotification` *{boolean}*	Notify or do not notify the user of updates for the current version of Dr.Web Industrial for Unix File Servers. Default value: `Yes`
`DefaultLogLevel` *{logging level}*	Default event logging level *for all* Dr.Web Industrial for Unix File Servers components. The value of this parameter is used if a custom logging level is not specified for some component. Default value: `Notice`
`VxcubeApiAddress` *{string}*	Domain name (FQDN) or IP address of a host running the Dr.Web vxCube API server to be connected to. Default value: *(not specified)*
`VxcubeApiKey` *{string}*	Dr.Web vxCube API key. Default value: *(not specified)*
`VxcubeProxyUrl` *{connection address}*	Address of the proxy server used for connecting to Dr.Web vxCube.

Parameter	Description
	Only HTTP proxies without authorization are supported. Possible values: <*connection address*>—proxy server connection parameters in the following format: `http://<host>:<port>`, where: • <*host*> is the host address of the proxy server (an IP address or a domain name, i.e. FQDN); • <*port*> is the port to be used. Default value: *(not specified)*

9.2. Dr.Web Ctl

In this section

- General information
- Remote host scanning

General information

You can manage operation of Dr.Web Industrial for Unix File Servers from the command line of the operating system. For that, you can use the special Dr.Web Ctl utility (`drweb-ctl`). You can use it to perform the following operations:

- start scanning files, disk boot records and executables of active processes;
- start scanning files on remote network hosts (see the note);
- start updating anti-virus components (virus databases, the scanning engine and so on depending on the distribution);
- view and change parameters of the Dr.Web Industrial for Unix File Servers configuration;
- view the status of the Dr.Web Industrial for Unix File Servers components and statistics on detected threats;
- view quarantine and manage quarantined objects (via the Dr.Web File Checker component);
- connect to a centralized protection server or disconnect from it.

User commands to control Dr.Web Industrial for Unix File Servers will only take effect if the Dr.Web ConfigD configuration daemon is running (by default, it is automatically run on system startup).



Note that some control commands require superuser privileges.

To elevate privileges, use the `su` command (change the current user) or the `sudo` command (run the specified command as another user).

The `drweb-ctl` tool supports auto-completion of commands for managing Dr.Web Industrial for Unix File Servers operation if this option is enabled your command shell. If the command shell does not allow auto-completion, you can configure this option. For that purpose, refer to the instruction manual for the used OS distribution.



When shutting down, the tool returns the exit code according to convention for the POSIX compliant systems: 0 (zero)—if an operation is successfully completed, non-zero—if otherwise.

The tool only returns a non-null exit code in the case of an internal error (for example, the tool could not connect to a component, the requested operation could not be executed, and so on). If the tool detects and possibly neutralizes a threat, it returns the null exit code, because the requested operation (such as `scan` and so on) is successfully completed. If you need to define the list of the detected threats and applied actions, analyze the messages displayed on the console.

Codes of all errors are listed in the Appendix F. Known Errors section.

Remote host scanning

Dr.Web Industrial for Unix File Servers allows you to scan files located on remote network hosts for threats. Such hosts can be not only fully-featured computing machines, such as workstations and servers, but also routers, set-top boxes, and other smart devices of the Internet of Things. To perform the remote scanning, the remote host has to provide a remote terminal access via *SSH* (*Secure Shell*) or *Telnet*. To access the device, you need to know an IP address and a domain name of the remote host, as well as the credentials of the user that can remotely access the system via *SSH* or *Telnet*. This user must have access rights to the scanned files (at least the reading rights).

This function can be used only for detection of malicious and suspicious files on a remote host. Elimination of threats (i.e. isolation in the quarantine, removal, and cure of malicious objects) using remote scanning is impossible. To eliminate the detected threats on the remote host, use administration tools provided directly by this host. For example, for routers and other smart devices, update the firmware; for computing machines, establish a connection (in a remote terminal mode, as one of the options) and perform the respective operations in the file system (remove or move files, etc.), or run the anti-virus software installed on them.

Remote scanning is performed only with the `drweb-ctl` command-line tool (using the `remotescan` command).

9.2.1. Command-Line Call Format

The call format for the command-line tool which manages Dr.Web Industrial for Unix File Servers operation is as follows:

```
$ drweb-ctl [<general options> | <command> [<argument>] [<command options>]]
```

where:

- *<general options>*—options that can be applied on startup when the command is not specified or can be applied for any command. Not mandatory for startup.
- *<command>*—command to be run by Dr.Web Industrial for Unix File Servers (for example, start scanning, output the list of quarantined objects and so on).
- *<argument>*—command argument. Depends on the specified command. Some commands do not accept arguments.
- *<command options>*—options for managing the operation of the specified command. Depend on the command. Some commands do not accept options.

9.2.2. General Options

The following general options are available:

Option	Description
`-h, --help`	Show general help information and exit. To display the help information on any command, use the call: `$ drweb-ctl <command> -h`
`-v, --version`	Show the module version and exit
`-d, --debug`	Show debug information when running the specified command. It cannot be run if a command is not specified. Use the call: `$ drweb-ctl <command> -d`

9.2.3. Commands

In this section

- Anti-Virus Scanning Commands
- Commands to Manage Detected Threats and Quarantine
- Configuration Management Commands
- Advanced Commands
  - Commands to Manage Updates and Operation in Centralized Protection Mode
  - Information Commands

9.2.3.1. Anti-Virus Scanning Commands

The following commands to manage anti-virus scanning are available:

Command	Description
`scan <path>`	**Purpose:** Initiate scanning the specified file or directory by the file scanning component Dr.Web File Checker. **Arguments** <*path*>—path (can be relative) to the file or directory to be scanned. This argument may be omitted if you use the `--stdin` or the `--stdin0` option. To specify several files that satisfy a certain criterion, use the `find` utility (see Usage Examples) and the `--stdin` or `--stdin0` option. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed. ⚠ Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Industrial for Unix File Servers is controlled by it. `--Report <type>`—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout <time interval>`—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--FollowSymlinks`—resolve symlinks automatically. `--PackerMaxLevel <number>`—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`.

Command	Description
	`--ArchiveMaxLevel` <*number*>—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel` <*number*>—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio` <*ratio*>—maximum compression ratio of scanned objects. The value must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract` <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Exclude` <*path*>—excluded path. The path can be relative and contain a file mask (with the following wildcards: `?` and `*`, as well as character classes `[ ]`, `[! ]` and `[^ ]`). Optional parameter; can be set more than once. `--OnKnownVirus` <*action*>—action to perform upon detection of a known threat by using the signature-based analysis. Allowed actions: `REPORT`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnIncurable` <*action*>—action to perform upon detection an incurable threat or when the curing action (`CURE`) has failed.

Command	Description
	Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnSuspicious` <*action*>—action to perform upon detection of a suspicious object using the heuristic analysis. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnAdware` <*action*>—action to perform upon detection of adware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnDialers` <*action*>—action to perform upon detection of a dialer. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnJokes` <*action*>—action to perform upon detection of joke software. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnRiskware` <*action*>—action to perform upon detection of riskware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnHacktools` <*action*>—action to perform upon detection of a hacktool. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`.  If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined (`QUARANTINE`) and not deleted (`DELETE`). `--stdin`—get the list of paths to be scanned from the standard input stream (*stdin*). Paths in the list must be separated with the new line character (`\n`). `--stdin0`—get the list of paths to scan from the standard input string (*stdin*). Paths in the list must be separated by the zero character NUL (`\0`).  When using `--stdin` and `--stdin0` options, the paths on the list should not contain patterns or regular expressions for a search. We recomment that you use the `--stdin` and `--stdin0` options to process a path list generated by an external utility, for example, `find` in the `scan` command (see Usage Examples).

Command	Description
`bootscan <device> | ALL`	**Purpose:** Start scanning boot records on specified disks using the file scan component Dr.Web File Checker. Both MBR and VBR records are scanned. **Arguments** <*disk drive*>—path to the block file of a disk device whose boot record you want to scan. You can specify several disk devices separated by spaces. The argument is mandatory. If `ALL` is specified instead of the device file, all boot records on all available disk devices will be checked. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed.  Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Industrial for Unix File Servers is controlled by it. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure` <*Yes*|*No*>—attempt or do not attempt to cure detected threats. If the value is set to `No`, only a notification about a detected threat is displayed. Default value: `No`. `--ShellTrace`—display additional debug information when scanning a boot record
`procscan`	**Purpose:** Initiate scanning of executables containing the code of currently running system processes with the Dr.Web File Checker component. If a malicious executable file is detected, it is neutralized, and all processes run by this file are forced to terminate.

Command	Description
	Arguments: None. This command is intended only for Dr.Web Industrial for Unix File Servers distributions running on GNU/Linux OSes. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed. Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Industrial for Unix File Servers is controlled by it. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned.

Command	Description
	If the value is set to `0`, nested objects are skipped. Default value: `8`. `--Exclude` <*path*>—path to be excluded from scanning. The path can contain a file mask with the following allowed symbols: `?` and `*`, as well as the symbol classes `[ ]`, `[! ]`, `[^ ]`. The path (including the path with the file mask) must be absolute. Optional parameter; can be set more than once. `--OnKnownVirus` <*action*>—action to perform upon detection of a known threat by using the signature-based analysis. Allowed actions: `REPORT`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnIncurable` <*action*>—action to perform upon detection an incurable threat or when the curing action (`CURE`) has failed. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnSuspicious` <*action*>—action to perform upon detection of a suspicious object using the heuristic analysis. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnAdware` <*action*>—action to perform upon detection of adware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnDialers` <*action*>—action to perform upon detection of a dialer. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnJokes` <*action*>—action to perform upon detection of joke software. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnRiskware` <*action*>—action to perform upon detection of riskware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnHacktools` <*action*>—action to perform upon detection of a hacktool. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`.

Command	Description
	 If a threat is detected in an executable file, Dr.Web Industrial for Unix File Servers terminates all processes started by the file.
`memscan --Cure` <*boolean*>	**Purpose**: Start process scan. Unlike the `procscan` command, scanning occurs only in random-access memory. **Arguments**: None. **Options** `--Cure` <*boolean*>—cure a threat upon detection. Mandatory parameter. Allowed values: • `On`, `Yes`, `True`—cure a threat upon detection; • `Off`, `No`, `False`—do not cure a threat upon detection. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one process in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `-d [--Debug]`—display debug information while running the specified command.
`netscan [`<*path*>`]`	**Purpose:** Start distributed scanning of the specified file or directory using the Dr.Web Network Checker agent for network data scanning. If there are no configured connections to other hosts that are running Dr.Web for Unix, then the scanning will be done only using the locally available scan engine (similar to the `scan` command). **Arguments** <*path*>—path to the file or directory to be scanned. If this argument is omitted, data from the stdin input stream will be scanned. **Options** `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report;

Command	Description
	• DEBUG—detailed report; • JSON—serialized report in the JSON format. Default value: BRIEF. --ScanTimeout <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to 0, scanning time is not limited. Default value: 0. --PackerMaxLevel <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to 0, nested objects are skipped. Default value: 8. --ArchiveMaxLevel <*number*>—maximum nesting level while scanning archives (.zip, .rar and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to 0, nested objects are skipped. Default value: 8. --MailMaxLevel <*number*>—maximum nesting level while scanning files of mailers (.pst, .tbb and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to 0, nested objects are skipped. Default value: 8. --ContainerMaxLevel <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, .jar files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to 0, nested objects are skipped. Default value: 8. --MaxCompressionRatio <*ratio*>—maximum compression ratio of scanned objects. The value must be no less than 2. Default value: 3000. --MaxSizeToExtract <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be

Command	Description
	skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure` <*Yes*|*No*>—attempt or do not attempt to cure detected threats. If the value is set to `No`, only a notification about a detected threat is displayed. Default value: `No`
`rawscan` <*path*>	**Purpose**: Start "raw" scanning of the specified file or directory with Dr.Web Scanning Engine directly, without the use of Dr.Web File Checker. ⚠ Threats detected during "raw" scanning are not included in the list of detected threats that can be displayed using the `threats` command. It is recommended that you use this command only to debug the functioning of Dr.Web Scanning Engine. Note that the command outputs the "cured" status, if at least *one* threat is neutralized of those threats that are detected in a file (not *all* threats might be neutralized). Thus, it is *not recommended* to use this command if you need thorough file scanning. In the latter case it is recommended to use the `scan` command. **Arguments** <*path*>—path to the file or directory to be scanned. **Options** `--ScanEngine` <*path*>—path to the Unix socket of Dr.Web Scanning Engine. If not specified, a standalone instance of the scan engine will be started (which will be shut down once the scanning is complete). `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited.

Command	Description
	Default value: `0`. `--PackerMaxLevel <number>`—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ArchiveMaxLevel <number>`—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel <number>`—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel <number>`—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio <ratio>`—maximum compression ratio of scanned objects. Must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract <number>`—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis <On|Off>`—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure <Yes|No>`—attempt or do not attempt to cure detected threats.

Command	Description
	If the value is set to `No`, only a notification about a detected threat is displayed. Default value: `No`. `--ListCleanItem`—output the list of clean (non-infected) files found inside the container that was scanned. `--ShellTrace`—enable display of additional debug information when scanning a file. `--Output` <*path to file*>—duplicate the output of the command to the specified file
`remotescan` <*host*> <*path*>	**Purpose**: Start scanning the specified file or directory at the specified remote host having connected to it using *SSH* or *Telnet*. ⚠ Threats detected during remote scanning are not neutralized and are also not added to the list of detected threats displayed using the `threats` command. This function can be used only for detection of malicious and suspicious files on a remote host. To eliminate detected threats on the remote host, it is necessary to use administration tools provided directly by this host. For example, for routers, set-top boxes, and other "smart" devices, a mechanism for a firmware update can be used; for computing machines, it can be done by connecting to them (as an option, using a remote terminal mode) and by performing corresponding operations in their file system (file removal or moving and so on), or by running an anti-virus software installed on them. **Arguments** • <*host*>—IP address or a domain name of the remote host to be connected to for scanning. • <*path*>—path to the file or directory to be scanned (the path must be absolute). **Options** `-l [--Login]` <*name*>—login (user name) used for authorization on the remote host via the selected protocol. If a user name is not specified, an attempt is made to connect to a remote host as the user who started the command. `-i [--Identity]` <*path to file*>—private key file used for authentication of the specified user via the selected protocol.

Command	Description
	 The `remotescan` command is designed to scan SOHO (Small Office / Home Office) routers. A key in the format used by default by the `ssh-keygen` utility is not supported. To scan a remote host via SSH, a key in the PEM format is required. To generate the key, use the command: `$ ssh-keygen -t rsa -m PEM -f rsa_pem` `-m [--Method]` <*SSH*|*Telnet*>—remote host connection method (protocol). If the method is not specified, SSH is used. `-p [--Port]` <*number*>—number of the port on the remote host for connecting via the selected protocol. Default value: Default port for the selected protocol (`22` for SSH, `23` for Telnet). `--UseChannels` <*number*>—number of data transfer channels. Default value: `5`. `--Password` <*password*>—password used for authentication of a user via the selected protocol.  Password is passed as plain text. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`.

Command	Description
	`--ArchiveMaxLevel` <*number*>—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel` <*number*>—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio` <*ratio*>—maximum compression ratio of scanned objects. Must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract` <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Exclude` <*path*>—path to be excluded from scanning. The path can contain a file mask with the following allowed symbols: `?` and `*`, as well as the symbol classes `[ ]`, `[! ]`, `[^ ]`. The path (including the path with the file mask) must be absolute. Optional parameter; can be set more than once. `--TransferListenAddress` <*address*>—address for receiving files transferred from the remote device for scanning. Optional parameter. If not indicated, an arbitrary address is used. `--TransferListenPort` <*port*>—port for receiving files transferred from the remote device for scanning.

Command	Description
	Optional parameter. If not indicated, an arbitrary port is used. `--TransferExternalAddress` <*address*>—address for the remote device to send files for scanning. Optional parameter. If not indicated, the `--TransferListenAddress` option value or the outgoing address of the already established session is used. `--TransferExternalPort` <*port*>—port to transfer files for scanning, specified for the remote device. Optional parameter. If not indicated, an automatically determined port is used. `--ForceInteractive`—use the SSH interactive session (only for SSH connections). *Optional parameter.*

9.2.3.2. Commands to Manage Detected Threats and Quarantine

The following commands for managing threats and quarantine are available:

Command	Description
`threats [`<*action*> <*object*>`]`	**Purpose**: Apply the specified action to earlier detected threats according to their identifiers. A type of the action is specified by the command option. If the action is not specified, displays information about detected but not neutralized threats. The information about threats is displayed according the format, specified using the non-mandatory `--Format` option. If the `--Format` option is not specified, the following information is displayed for each threat: • an identifier assigned to the threat (its ordinal number); • the full path to the infected file; • information about the threat (its name and type according to the classification of the Doctor Web company); • information about the file: its size, owner, time of last modification; • history of operations applied to an infected file: detection, applied actions and so on. **Arguments**: None. **Options** `--Format "`<*format string*>`"`—output information about threats in the specified format. The description of the format string is below.  If this option is specified together with any action option, it is ignored.

Command	Description
	`-f [--Follow]`—wait for new messages about new threats and display them once they are received (CTRL+C interrupts the waiting). If this option is specified together with any action option, it is ignored. `--Cure` <*threat list*>—attempt to cure the listed threats (threat identifiers are comma-separated); `--Quarantine` <*threat list*>—quarantine the listed threats (threat identifiers are comma-separated); `--Delete` <*threat list*>—delete the listed threats (threat identifiers are comma-separated); `--Ignore` <*threat list*>—ignore the listed threats (threat identifiers are comma-separated). If you need to apply the action to all detected threats, specify `All` instead of <*threat list*>. For example, the command: `$ drweb-ctl threats --Quarantine All` quarantines all detected malicious objects. `--Directory` <*list of directories*>—output only threats detected in files in directories from <*list of directories*>. If this option is specified together with any action option, it is ignored.
`quarantine [`<*action*> <*object*>`]`	**Purpose:** Apply an action to the specified object in quarantine. If the action is not specified, information about quarantined objects and their identifiers together with brief information about original files put in quarantine is displayed. Information about isolated objects is output according to a format specified with the optional `--Format` parameter. If the `--Format` parameter is not specified, the following information is output for every isolated (quarantined) object: • an identifier assigned to a quarantined object; • the original path to the file that was moved to quarantine; • the date of putting the file in quarantine; • information about the file: its size, owner, time of last modification; • information about the threat (name of the threat, threat type according to the classification used by the Doctor Web company). **Arguments**: None. **Options** `--Format "`<*format string*>`"`—display information about quarantined objects in the specified format. The description of format string is below.

Command	Description
	If this option is specified together with any action option, it is ignored. `-f [--Follow]`—wait for new messages about new threats and display them once they are received (CTRL+C interrupts the waiting). If this option is specified together with any action option, it is ignored. `-a [--Autonomous]`—run a standalone instance of the Dr.Web File Checker file scanning component to perform the specified quarantine action and shut down the component upon completion. This option can be used together with any options mentioned below. `--Discovery [`<*list of directories*>`,]` searches for quarantine directories in the specified list of directories and add them to the consolidated quarantine upon detecting a threat. If the <*list of directories*> is not specified, search for quarantine directories in the common locations of the file system (volume mounting points and user home directories). This option can be specified not only with the `-a` (`--Autonomous`) option (see above), but also with any options/actions listed below. Moreover, if the `quarantine` command is run in standalone instance mode, that is, with the `-a` (`--Autonomous`) option but without the `--Discovery` option, then it has the same effect as calling: `quarantine --Autonomous --Discovery` `--Delete` <*object*>—delete the specified quarantined object. Quarantined objects are deleted permanently—this action is irreversible. `--Restore` <*object*>—restore the specified object from the quarantine to its original location. This command may require to run `drweb-ctl` with superuser (the *root* user) privileges. You can restore the file from quarantine even if it is infected. `--Cure` <*object*>—attempt to cure the specified object in the quarantine. Even if the object was successfully cured, it will remain in quarantine. To restore the cured object from quarantine, use the `--Restore` option. `--TargetPath` <*path*>—restore an object from quarantine to the specified location: either as a file with the specified name (if <*path*> is a

Command	Description
	path to a file), or to the specified directory (if <*path*> is a path to a directory). A path can be absolute or relative (with regard to the current directory). This option can only be used in combination with the `--Restore` option. As an <*object*>, specify the object identifier in quarantine. To apply the action to all quarantined objects, specify `All` instead of <*object*>. For example, the command `$ drweb-ctl quarantine --Restore All --TargetPath test` restores all quarantined objects and puts them in the `test` subdirectory located in the current directory from which the `drweb-ctl` command was run. If the `--Restore All` variant is indicated together with the additional option `--TargetPath`, this option must set a path to a directory, not to a file.

Formatted output for threats and quarantine commands

The output format is defined using the format string specified as the optional argument `--Format`. The format string must be put in quotes. The format string can include common symbols (displayed "as is"), as well as special markers which will be replaced with corresponding information at the output. The following markers are available:

1. Common for `threats` and `quarantine` commands:

Marker	Description
`%{n}`	New line
`%{t}`	Tabulation
`%{threat_name}`	The name of the detected threat according to the classification of the Doctor Web company
`%{threat_type}`	Threat type ("known virus" and so on) according to Doctor Web classification
`%{size}`	Original file size
`%{origin}`	The full name of the original file with path
`%{path}`	Synonym of `%{origin}`

Marker	Description
`%{ctime}`	Modification date/time of the original file in "*%Y-%b-%d %H:%M:%S*" format (for example, "`2018-Jul-20 15:58:01`")
`%{timestamp}`	Similar to `%{ctime}`, but in the *Unix timestamp* format
`%{owner}`	The original file owner
`%{rowner}`	The remote owner of the original file (if not applicable or value is unknown it is replaced with `?`)

2. Specific for `threats` command:

Marker	Description
`%{hid}`	The identifier of the threat record in the history of events associated with the threat
`%{tid}`	Threat identifier
`%{htime}`	Date/time of the event related to the threat
`%{app}`	The identifier of the Dr.Web Industrial for Unix File Servers component which processed a threat
`%{event}`	The latest event related to a threat: • `FOUND`—threat was detected; • `CURE`—threat was cured; • `QUARANTINE`—file with a threat was quarantined; • `DELETE`—file with a threat was deleted; • `IGNORE`—threat was ignored; • `RECAPTURED`—threat was detected by another component
`%{err}`	Error message text (if no error has occurred, the text is replaced with an empty string)

3. Specific for `quarantine` command:

Marker	Description
`%{qid}`	The identifier of the quarantined object
`%{qtime}`	Date/time of moving the object to quarantine
`%{curetime}`	Date/time of curing attempt of the quarantined object (if not applicable or the value is unknown, it is replaced with `?`)
`%{cureres}`	The result of the quarantined object curing attempt: • `cured`—threat was cured; • `not cured`—threat was not cured or no curing attempts were made

Example

```
$ drweb-ctl quarantine --Format "{%{n} %{origin}: %{threat_name} - %{qtime}%
{n}}"
```

This command displays quarantine contents as records of the following type:

```
{
 <path to file>: <threat name> - <date of putting in quarantine>
}
...
```

9.2.3.3. Configuration Management Commands

The following commands to manage configuration are available:

Command	Description
`cfset` <*section*>`.`<*parameter*> <*value*>	**Purpose**: Change the active value of the specified parameter in the current configuration of Dr.Web Industrial for Unix File Servers. **Arguments** • <*section*>—name of the configuration file section which provides the parameter. This argument is mandatory. • <*parameter*>—name of the parameter to be changed. This argument is mandatory. • <*value*>—new parameter value. This argument is mandatory. To specify a parameter value, the format <*section*>.<*parameter*> <*value*> is always used, the assignment character = is not used. If you want to indicate several parameter values, you need to repeatedly call the `cfset` command, as many times as the number of parameter values you want to add. To add a new value to the list of parameter values, you need to use the `-a` option (see below). You cannot specify the string <*parameter*> <*value 1*>, <*value 2*> as an argument, because the string "<*value 1*>, <*value 2*>" will be considered one value of <*parameter*>. For description of the configuration file, refer to the section Appendix D. Dr.Web Industrial for Unix File Servers Configuration File, as well as the documentation displayed upon running `man 5 drweb.ini`.

Command	Description
	Options `-a [--Add]`—do not substitute the current parameter value but add the specified value to the list (allowed only for parameters that can accept a list of values). This option should also be used for adding new parameter groups with a tag. `-e [--Erase]`—do not substitute the current parameter value but remove the specified value from the list (allowed only for parameters that can have several values, specified as a list). `-r [--Reset]`—reset the parameter value to the default. At that, <*value*> is not required in the command and is ignored if specified. Options are not mandatory. If they are not specified, then the current parameter value (including a list of values) are substituted with the specified value. If you use the `-r` option for sections that contain individualized parameter settings for shared directories for the SplDer Guard for SMB monitor, the parameter value in the individualized settings section will be changed to the value of its corresponding "parent" parameter in the component settings section. If you need to add parameter section for a Samba shared directory with the <*tag*> tag, run the command `cfset SmbSpider.Share.<tag> -a` for example: `cfset SmbSpider.Share.AccountingFiles -a` This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`cfshow [<section>[.<parameter>]]`	**Purpose**: Display parameters of the current configuration of Dr.Web Industrial for Unix File Servers. The command to display parameters is specified as follows: <*section*>.<*parameter*> = <*value*>. Sections and parameters of non-installed components are not displayed by default. **Arguments** • <*section*>—name of the configuration file section parameters of which are to be displayed. The argument is optional. If not specified, parameters of all configuration file sections are displayed.

Command	Description
	• <*parameter*>—name of the displayed parameter. Optional argument. If not specified, all parameters of the section are displayed. Otherwise, only this parameter is displayed. If a parameter is specified without the section name, all parameters with this name from all of the configuration file sections are displayed. **Options** `--Uncut`—display all configuration parameters, and not only those used with the currently installed set of components. If the option is not specified, only parameters used by the installed components are displayed. `--Changed`—display only those parameters whose values differ from the default ones. `--Ini`—display parameter values in the `.ini` file format: at first, the section name is specified in square brackets, then the section parameters listed as <*parameter*> = <*value*> pairs (one pair per line). `--Value`—output only the value of the specified parameter. The <*parameter*> argument is mandatory in this case.
`reload`	**Purpose**: Reload the configuration of Dr.Web Industrial for Unix File Servers. For that purpose, the Dr.Web ConfigD configuration management daemon performs the following actions: • rereads the configuration and notifies all Dr.Web Industrial for Unix File Servers components about its changes; • reopens the Dr.Web Industrial for Unix File Servers log; • starts the components that use virus databases (including the scanning engine); • attempts to start those components that were shut down abnormally. **Arguments**: None. **Options**: None

9.2.3.4. Advanced Commands

In this section

- Commands to Manage Updates and Operation in Centralized Protection Mode
- Information Commands

9.2.3.4.1. Commands to Manage Updates and Operation in Centralized Protection Mode

The following commands for managing updates are available, as well as commands for operation in the centralized protection mode:

Command	Description
`update`	**Purpose**: Start the updating process of anti-virus components (virus databases, the scan engine and so on, depending on the distribution) from Doctor Web update servers, terminate the updating process if it is already running, or perform rollback of the latest update to the previous versions of updated files. The command has no effect if Dr.Web Industrial for Unix File Servers is connected to the centralized protection server. **Arguments**: None. **Options** `--Stop`—terminate the running update process. `--Rollback`—roll back the last update and restore the previous version of the files that have been updated during the last update. `--From` <*path*>—apply updates offline from a specified directory. `--Path` <*path*>—store files for updating offline in a specified directory. If this directory already has files, then they will be updated.
`esconnect` <*server*>`[:`<*port*>`]`	**Purpose**: Connect Dr.Web Industrial for Unix File Servers to the specified centralized protection server. For details on the operation modes, refer to the Operation Modes section. **Arguments** • <*server*>—IP address or network name of the host on which the centralized protection server is operating. This argument is mandatory. • <*port*>—port number used by the centralized protection server. The argument is optional and should be specified only if the centralized protection server uses a non-standard port. **Options** `--Certificate` <*path*>—file path to a certificate of the centralized protection server, the connection to which will be established. `--Login` <*ID*>—login (workstation identifier) used for connection to the centralized protection server. `--Password` <*password*>—password for connection to the centralized protection server.

Command	Description
	`--Compress` <*On*|*Off*>—enable (`On`) or disable (`Off`) forced compression of transmitted data. If not specified, usage of compression is determined by the server. `--Encrypt` <*On*|*Off*>—enable (`On`) or disable (`Off`) forced encryption of transmitted data. If not specified, usage of encryption is determined by the server. `--Newbie`—connect as a "newbie" (get a new account on the server). `--Group` <*ID*>—identifier of the group to which the workstation is added on connection. Must be specified together with the `--Password` option. `--Rate` <*ID*>—identifier of the tariff group applied to your workstation when it is included in one of the centralized protection server groups. Must be specified together with the `--Group` option. `--CfgFile` <*path*>—connect to the centralized protection server using the configuration file with connection settings. This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`esdisconnect`	**Purpose**: Disconnect Dr.Web Industrial for Unix File Servers from the centralized protection server and switch it to a standalone mode. The command has no effect if Dr.Web Industrial for Unix File Servers already operates in standalone mode. **Arguments**: None. **Options**: None. This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.

9.2.3.4.2. Information Commands

The following information commands are available:

Command	Description
`appinfo`	**Purpose**: Output information about active Dr.Web Industrial for Unix File Servers components. The following information is output for each running component: • internally used name;

Command	Description
	• GNU/Linux process identifier (PID); • state (running, stopped and so on); • error code, if the component has been terminated owing to an error; • additional information (optional); For the configuration daemon (`drweb-configd`), the following is output as additional information: • the list of installed components—*Installed*; • the list of components which must be run by the configuration daemon—*Should run*. **Arguments**: None. **Options** `-f [--Follow]`—wait for new messages on module status change and display them once such a message is received (CTRL+C interrupts waiting)
`baseinfo`	**Purpose**: Display the information on the current version of the scan engine and status of virus databases. The following information is displayed: • version of the scan engine; • release date and time of the virus databases being used; • the number of available threat records; • the time of the last successful update of the virus databases and of the scan engine; • the time of the next scheduled automatic update. **Arguments**: None. **Options** `-l [--List]`—display the full list of loaded files of virus databases and a number of threat records in each file
`events`	**Purpose**: Display Dr.Web Industrial for Unix File Servers events. In addition, this command allows you to manage the events (mark them as read or delete them). **Arguments**: None. **Options** `--Report` <*type*>—type of the event report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report;

Command	Description
	• `JSON`—serialized report in the JSON format. `-f [--Follow]`—wait for new events and display them upon their occurrence (CTRL+C interrupts waiting). `--ShowSeen`—display already read events as well; `-s [--Since]` <*date, time*>—show the events that occurred before the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `-u [--Until]` <*date, time*>—show the events that occurred no later than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `-t [--Types]` <*type list*>—show the events of the specified types only (types are comma-separated). The following event type is available: `UnexpectedAppTermination`—unexpected component shutdown. To view all types of events, use `All`. `--Show` <*list of events*>—display the listed events (event identifiers are comma-separated); `--Delete` <*list of events*>—remove the listed events (event identifiers are comma-separated); `--MarkAsSeen` <*list of events*>—mark the listed events as read (event identifiers are comma-separated). If you want to mark as "read" or delete all events, specify `All` instead of <*events list*>. For example, the command `$ drweb-ctl events --MarkAsSeen All` will mark all existing events as "read"
`report` <*type*>	**Purpose**: Create a report on Dr.Web Industrial for Unix File Servers events in the HTML format (the page body is output to the specified file). **Arguments** <*type*>—event type that required reporting (indicate one type). See allowed values in the `--Types` option description of the `events` command. A mandatory argument. **Options** `-o [--Output]` <*path to file*>—save the report to the specified file. The option is mandatory. `-s [--Since]` <*date, time*>—report events that occurred no earlier than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`).

Command	Description
	`-u [--Until]` <*date, time*>—report events that occurred no later than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `--TemplateDir` <*path to directory*>—path to the directory that contains HTML report templates. Options `-s`, `-u`, and `--TemplateDir` are not mandatory.
`license`	**Purpose**: Display the information about the currently active license or get the key file for a license that has already been registered (for example, that has been registered on the company website). If no options are specified, then the following information is output (if you are using a license for the standalone mode): • a license number, • date and time when the license expires. If you are using a license provided to you by a centralized protection server (for the use of the product in the centralized protection mode or mobile mode), the corresponding message is output. **Arguments**: None. **Options** `--GetRegistered` <*serial number*>—get a license key file for the specified serial number, if the conditions for the provision of a new key file have not been breached (for example, breached by using the product not in centralized protection mode, when the license is managed by a centralized protection server). `--NetworkTimeout` <*time interval*>—timeout in milliseconds for network operations during the use of the `license` command. This parameter is used to continue activation when the connection is temporarily lost. If the connection is re-established before the timeout expires, the activation will be resumed. If `0` is specified, then there is no timeout. Default value: `0`. `--Proxy http://`<*username*>`:`<*password*>`@`<`server address`>`*:*<*port*>—get a license key via the proxy server (used only with the `--GetRegistered` option). *The serial number must be first registered at the company website.* For further information about licensing Dr.Web products, refer to the Licensing section. To register a serial number, an internet connection is required.

Command	Description
`log`	**Purpose**: Display the latest log records of Dr.Web Industrial for Unix File Servers in the console (the *stdout* stream, similar to the `tail` command). **Arguments**: None. **Options** `-s [--Size]` <*number*>—number of the latest log records to be displayed on the screen. `-c [--Components]` <*components list*>—list of component identifiers whose records are displayed. Identifiers are space-separated. If no argument is defined, all available records logged by all components are displayed. Actual identifiers of the installed components (e.g. internal component names displayed in the log) can be displayed using the `appinfo` command. `-f [--Follow]`—wait for new log records and display them once they are received (CTRL+C interrupts waiting). This command requires `drweb-ctl` to be started with superuser (usually the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`stat`	**Purpose:** Display statistics about the operation of components that process files or about the operation of the network data scanning agent Dr.Web Network Checker (press CTRL+C or Q to interrupt displaying the statistics). The statistics output includes: • a name of the component that initiated file scanning; • component PID; • an average number of files processed per second during the last minute, 5 minutes, 15 minutes; • a percentage of using the cache of the scanned files; • an average number of scan errors per second. For the distributed scanning agent, the following information is displayed: • a list of local clients that initiated scanning; • a list of remote hosts that received files for scanning; • a list of remote hosts that sent files for scanning. For local clients of the distributed scanning agent, their PID and name are specified; for remote clients—an address and port of the host. For both clients—local and remote—the following information is displayed:

Command	Description
	• an average number of files scanned per second; • an average number of sent and received bytes per second; • an average number of errors per second. **Arguments**: None. **Options** `-n [--Netcheck]`—display statistics on operation of the network data scanning agent

9.2.4. Usage Examples

In this section

- Scanning objects
  - Simple scanning commands
  - Scanning of files selected by criteria
  - Scanning of additional objects
- Configuration management
- Threat management
- Example of operation in standalone instance mode

Scanning objects

Simple scanning commands

1. Perform scanning of the `/home` directory with default parameters:

```
$ drweb-ctl scan /home
```

2. Scan file paths listed in the `daily_scan` file (one path per line):

```
$ drweb-ctl scan --stdin < daily_scan
```

3. Perform scanning of the boot record on the *sda* drive:

```
$ drweb-ctl bootscan /dev/sda
```

4. Perform scanning of the running processes:

```
$ drweb-ctl procscan
```

Scanning of files selected by criteria

Examples for file selection for scanning are listed below and use the result of the `find` utility operation. The obtained list of files is sent to the `drweb-ctl scan` command with the `--stdin` or `--stdin0` parameter.

1. Scan listed files returned by the `find` utility and separated with the NUL (`\0`) character:

```
$ find -print0 | drweb-ctl scan --stdin0
```

2. Scan all files in all directories, starting from the root directory, on one partition of the file system:

```
$ find / -xdev -type f | drweb-ctl scan --stdin
```

3. Scan all files in all directories, starting from the root directory, with the exception of the `/var/log/messages` and `/var/log/syslog` files:

```
$ find / -type f ! -path /var/log/messages ! -path /var/log/syslog | drweb-ctl scan
--stdin
```

4. Scan all files of the *root* user in all directories, starting from the root directory:

```
$ find / -type f -user root | drweb-ctl scan --stdin
```

5. Scan all files of the *root* and *admin* users in all directories, starting from the root directory:

```
$ find / -type f \( -user root -o -user admin \) | drweb-ctl scan --stdin
```

6. Scan all files of the users with UID within the range of 1000–1005 in all directories, starting from the root directory:

```
$ find / -type f -uid +999 -uid -1006 | drweb-ctl scan --stdin
```

7. Scan files in all directories, starting from the root directory, with a nesting level of no more than five:

```
$ find / -maxdepth 5 -type f | drweb-ctl scan --stdin
```

8. Scan files in a root directory while ignoring files in subdirectories:

```
$ find / -maxdepth 1 -type f | drweb-ctl scan --stdin
```

9. Scan files in all directories, starting from the root directory, while following all symbolic links:

```
$ find -L / -type f | drweb-ctl scan --stdin
```

10. Scan files in all directories, starting from the root directory, without following symbolic links:

```
$ find -P / -type f | drweb-ctl scan --stdin
```

11. Scan files created no later than May 1, 2017 in all directories, starting from the root directory:

```
$ find / -type f -newermt 2017-05-01 | drweb-ctl scan --stdin
```

Scanning of additional objects

Scan objects located in the `/tmp` directory on the remote host *192.168.0.1* by connecting to it via SSH as the user *user* with the password *passw*:

```
$ drweb-ctl remotescan 192.168.0.1 /tmp --Login user --Password passw
```

Configuration management

1. Display information about the running components of Dr.Web Industrial for Unix File Servers:

```
$ drweb-ctl appinfo
```

2. Display all parameters of the `[Root]` section:

```
$ drweb-ctl cfshow Root
```

3. Set `No` as the value of the `Start` parameter in the `[LinuxSpider]` section of the active configuration (this will disable SpIDer Guard):

```
# drweb-ctl cfset LinuxSpider.Start No
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl cfset LinuxSpider.Start No
```

4. Force update of anti-virus components of Dr.Web Industrial for Unix File Servers:

```
$ drweb-ctl update
```

5. Restart the component configuration of Dr.Web Industrial for Unix File Servers:

```
# drweb-ctl reload
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl reload
```

6. Connect Dr.Web Industrial for Unix File Servers to a centralized protection server operating on host *192.168.0.1* if a server certificate is stored in the `/home/user/cscert.pem` file:

```
$ drweb-ctl esconnect 192.168.0.1 --Certificate /home/user/cscert.pem
```

7. Connect Dr.Web Industrial for Unix File Servers to the centralized protection server using the `install.cfg` configuration file:

```
$ drweb-ctl esconnect --CfgFile <path to install.cfg>
```

8. Disconnect Dr.Web Industrial for Unix File Servers from the centralized protection server:

```
# drweb-ctl esdisconnect
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl esdisconnect
```

9. View the last log records made by the `drweb-update` and `drweb-configd` components in the Dr.Web Industrial for Unix File Servers log:

```
# drweb-ctl log -c Update ConfigD
```

Threat management

1. Display information on detected threats:

```
$ drweb-ctl threats
```

2. Quarantine all files containing non-neutralized threats:

```
$ drweb-ctl threats --Quarantine All
```

3. Display the list of quarantined files:

```
$ drweb-ctl quarantine
```

4. Restore all quarantined files:

```
$ drweb-ctl quarantine --Restore All
```

Example of operation in standalone instance mode

Scan files and process quarantine in standalone instance mode:

```
$ drweb-ctl scan /home/user -a --OnKnownVirus=QUARANTINE
$ drweb-ctl quarantine -a --Delete All
```

The first command will scan files in the `/home/user` directory in standalone instance mode. Files containing known threats will be quarantined. The second command will process quarantine content (in standalone instance mode as well) and remove all quarantined objects.

9.2.5. Configuration Parameters

The Dr.Web Ctl command-line management tool does not have its own parameter section in the unified configuration file of Dr.Web Industrial for Unix File Servers. The tool uses the parameters specified in the `[Root]` section.

9.3. Dr.Web Management Web Interface

In this section

- Function
- System requirements of the web interface
- Accessing the web interface
- Main menu

Function

The Dr.Web Industrial for Unix File Servers management web interface allows you to:

- view the current state of Dr.Web Industrial for Unix File Servers components, start or stop some of them;
- view the status of updates and start an updating process manually, if necessary;
- view the status of the product license and load a license key, if necessary;
- view the list of detected threats and manage quarantined objects (only threats detected in the local file system with the Dr.Web File Checker component are displayed);
- edit the settings of Dr.Web Industrial for Unix File Servers components;
- connect Dr.Web Industrial for Unix File Servers to a centralized protection server or switch to the standalone mode;
- start an on-demand scanning of local files (including by dragging and dropping them to the page opened in your browser).

System requirements of the web interface

Correct operation of the web interface is guaranteed for the following web browsers:

- Microsoft Internet Explorer 11 and later;
- Mozilla Firefox 25 and later;
- Google Chrome 30 and later.

Accessing the web interface

To access the web interface, type the following address in the address bar of your browser:

```
https://<drweb_host>:<port>/
```

where <*drweb_host*> is the IP address or the name of the host on which Dr.Web Industrial for Unix File Servers including the Dr.Web HTTPD web interface server operates, and <*port*> is the port (on this host) listened by Dr.Web HTTPD. To access components of Dr.Web Industrial for

Unix File Servers running on the local host, use the IP address `127.0.0.1` or the name `localhost`. By default, the port is `4443`.

Thus, to access the web interface on the local host by default, use the following URL:

```
https://127.0.0.1:4443/
```

After the connection to the management server is established, the startup page opens and displays the authentication form. To access management functions, fill in the authentication form by specifying the login and password of a user who has administrative privileges on the host on which Dr.Web Industrial for Unix File Servers operates.

If needed, you can provide authorization via the web interface using a personal user certificate. To do so:

1. Create a personal certificate signed by a certificate authority certificate.
2. Import the signed certificate as a user authorization certificate in the browser that is used to connect to the management web interface.
3. In the Dr.Web HTTPD settings (the `AdminSslCA` parameter), specify a path to the certificate authority certificate used to sign your personal certificate.

If you use a personal user certificate to authorize on the web interface, the authorization form does not appear, and the user is authorized as *root*.

If necessary, refer to the Appendix E. Generating SSL Certificates section.

Main menu

In the left pane of the web interface, which appears once you have successfully passed authentication, there is a main menu with the following items:

- **Main**—open the main page that displays the list of installed components of Dr.Web Industrial for Unix File Servers and their status.
- **Threats**—open a page that displays all threats detected on the server. In this section, you can manage detected threats, for example, quarantine infected objects, rescan, cure or delete malicious objects.
- **Settings**—open a page with settings of Dr.Web Industrial for Unix File Servers components installed on the server.
- **Information**—open a page that displays brief information about the version of this web interface and about the state of the virus databases.
- **Help**—open a new browser tab with the help information on Dr.Web for Unix products.
- **Scan File**—display a panel for quick file scanning, which will stay available on top of any opened page of the web interface until you close this panel.
- **Log Out**—end the current management web interface session (unavailable if the authentication with a user personal certificate is used).

9.3.1. Managing the Components

You can view the list of components included in Dr.Web Industrial for Unix File Servers and manage their operation on the **Main** page.

The listed Dr.Web Industrial for Unix File Servers components are separated into two groups: main components that monitor threats and service components that are responsible for the overall correct operation of Dr.Web Industrial for Unix File Servers. The list of main components (depends on the distribution) is displayed as a table in the upper part of the page. The following information is provided for each component:

1. **Name**. Click the name to open the settings page for a component.
2. **State**. A state of the component is indicated by a switch and a text label reflecting the current state of the component. To start the component or to suspend its operation, click the switch. The possible states of the switch are:

	Component is disabled and not used.
	Component is enabled and operates correctly.
	Component is enabled but does not operate because of an error.

   If an error occurred in the operation of the component, an error message is displayed instead of a note about a component state. Click to open a window with detailed information about the occurred error and with recommendations for resolving it.
3. **Load**. An average number of files processed by the component per second within the last minute, 5 minutes, 15 minutes respectively are specified (three numbers separated with a slash).
4. **Errors**. An average number of errors encountered in the operation of the component per second within the last minute, 5 minutes, 15 minutes respectively are specified (three numbers separated with a slash).

To display a tooltip, place the cursor over .

Below the table that provides the information about the main components, you will find Dr.Web Industrial for Unix File Servers service components (such as Dr.Web Scanning Engine, Dr.Web File Checker and so on) listed as a set of tiles. Their states and operational statistics are also displayed. To open a settings page of a component, click the name of the required component. As a rule, these components are started and stopped automatically when necessary. If any of them can be started or stopped manually by the user, the tile of the service component displays a switch for starting and stopping the component besides its name and operational statistics.

The bottom of the page displays information about the virus databases and the license. To force updating the virus databases, click **Update**. By clicking **Renew** (or **Activate license**, depending on the current state of your license) you can renew or activate your license by uploading a relevant key file for Dr.Web Industrial for Unix File Servers to the server.

9.3.2. Threat Management

You can view a list of detected threats and manage a reaction to them on the **Threats** page.

This page displays the full list of threats detected by the components of Dr.Web Industrial for Unix File Servers that monitor and scan the file system. At the top of the page, you can see a menu which allows filtration of threats on the basis of their categories:

- **All**—display all detected threats (including both active and quarantined threats);
- **Active**—display only active threats, that is, those that were detected but not neutralized yet;
- **Blocked**—display those threats that were not neutralized, but access to files containing them was blocked for users (only with regard to file storage units monitored by SpIDer Guard for SMB);
- **Quarantined**—display quarantined threats;
- **Errors**—display threats whose processing has finished with an error.

To the right of the name of each category in the menu, a number of detected threats that fall into this category is displayed. To display threats of a required category, click its name in the menu.

For each threat, the following information is listed:

- **File**—name of a file that contains a malicious object (a file path is not specified);
- **Owner**—name of a user who owns the file with a threat;
- **Component**—name of a component that detected the threat;
- **Threat**—name of the threat detected in the file, as classified by the Doctor Web company.

For an object selected in the list, the following detailed information is displayed to the right of the list:

- name of the threat (displayed as a link that opens a page of the Dr.Web virus information library with a threat description);
- file size, in bytes;
- name of the component that detected the threat;
- date and time of threat detection;
- file last modification date and time;
- name of the user who owns the infected file;
- name of the group that includes the file owner;
- name of the user who placed the file in the storage (only for file storage units monitored by SpIDer Guard for SMB);
- identifier that was assigned to the infected file (if the file was quarantined);
- full path to the file original location (at the moment of threat detection).

To select an object, click the corresponding line on the list. To select multiple objects, select the check boxes for the corresponding objects. To select all objects or cancel the selection at once, toggle the check box in the **File** field in the threat list header.

To apply actions to the objects selected on the list of threats, click the corresponding button on the toolbar that is located directly above the threat list. The toolbar contains the following buttons:

	Delete the selected files.
	Restore the selected files from quarantine to their original location.
	Apply an additional action to the selected files (available actions are specified in a drop-down list). The following additional actions are available: • **Quarantine**—quarantine the files with threats; • **Cure**—attempt to cure the threats; • **Ignore**—ignore the threats detected in the selected files and remove the threats from the list.

 Some buttons can be unavailable depending on the type of the selected threats.

The **Threats** page also allows you to filter displayed threats based on a search query. To filter out unnecessary threats and display only those that correspond to the query, use the search box. The box is displayed on the right side of the toolbar and contains . To filter the threat list, enter a word in the search box. At that, all threats that do not have the entered word in their name or description will be hidden (the search is case-insensitive). To clear search results and display the unfiltered list, click  in the search box or erase the word.

9.3.3. Managing Settings

In this section

- Viewing and editing component settings in a tabular form
- Viewing and editing component settings in the .ini editor
- Additional information

You can view and change current configuration parameters of the components included in Dr.Web Industrial for Unix File Servers and listed on the main page. For that, open the **Settings** page. On this page, you can also switch Dr.Web Industrial for Unix File Servers to a *centralized protection* mode or a *standalone* mode (for further information about these modes, refer to the Operation Modes section).

On the left side of the page, a menu is displayed that contains the names of all Dr.Web Industrial for Unix File Servers components whose settings can be viewed and adjusted. To view

and adjust the settings of a component, select its name from the menu by clicking it. The name of the component whose settings you are currently viewing and editing is highlighted in this menu to the left.

- The **Centralized protection** item of the menu takes you to the page for managing the centralized protection mode.
- The **General Settings** item of the menu corresponds to the settings of Dr.Web ConfigD, which is responsible for the overall functioning of Dr.Web Industrial for Unix File Servers.

If a component has additional, specific sections of settings apart from a section with its main settings, then an icon indicating that you can collapse or expand additional (dependent) sections is displayed on the menu to the left of the component name. If the icon looks like ▸, additional sections are hidden. If the icon looks like ▾, additional sections are displayed on the menu, one per line. To collapse or expand the list of additional sections for a component, click the collapse/expand icon next to the name of the required component.

- The additional sections of the component settings are indented to the right. To view or edit parameters of an additional section, click its name.
- To add an additional subsection with settings for a component, if possible, click +. This icon is positioned to the right of the component name and appears when you point to the component name. Then specify a unique name (tag) for a new subsection and click **OK**. To close the window without creating a subsection, click **Cancel**.
- To remove a subsection, click ×. This icon is positioned to the right of the subsection name (tag) and appears when you point to the component name. Then, confirm that you want to remove the subsection by clicking **Yes** or cancel by clicking **No**.

At the top of the settings page, you can see a menu that allows you to change a viewing mode. The following modes are available:

- **All**—display all component configuration parameters that can be viewed and adjusted in a table editor;
- **Changed**—display only those component configuration parameters that have values different from the defaults in the table editor;
- **Ini Editor**—display component configuration parameters that have values different from the defaults in an editor for parameters in the configuration file format (in `parameter = value` pairs).

The settings management page also contains a panel for filtering the list of displayed parameters based on a search query. To filter the list of parameters and keep only those parameters whose description contains the given string, use the search bar. The bar is located on the right side of the menu controlling the display mode and contains . To filter the parameter list, enter a word in the search bar. At that, all parameters whose description does not contain the specified word will be hidden (the search is case-insensitive). To clear search results and return to the initial parameter list, click  in the search bar or delete the search word.

Parameters can be filtered only when they are displayed in tabular form (in **All** mode or **Changed** mode).

Viewing and editing component settings in a tabular form

When viewing parameters in tabular form (the **All** mode or the **Changed** mode), each table row contains a name and a description of a parameter (on the left) and its current value (on the right). For boolean parameters (those that have only two available values: “Yes” and “No”), a check box is displayed instead of a value (the selected check box means “Yes” and the cleared one means “No”).



When you view all parameters (and not only those that were changed), the modified (non-default) values are provided on the list in bold.

The complete parameter list is split into sections, such as **General**, **Advanced** and so on. To collapse or expand a table section, click its title. When the section is collapsed and its parameters are not displayed in the table, › is displayed to the left of the section name. When the section is expanded and its parameters are displayed in the table, ⌄ is displayed to the left of the section name.

To adjust a parameter, click its current value on the table (for a boolean parameter—select or clear the corresponding check box). If the parameter has a set of predefined values, clicking the current value opens a drop-down list in which a required value can be set. If the parameter has a numerical value, clicking the current value opens an editing field directly in the table. In this case, specify the required value and press ENTER. In all these cases the changed parameter value is immediately stored in the component settings.

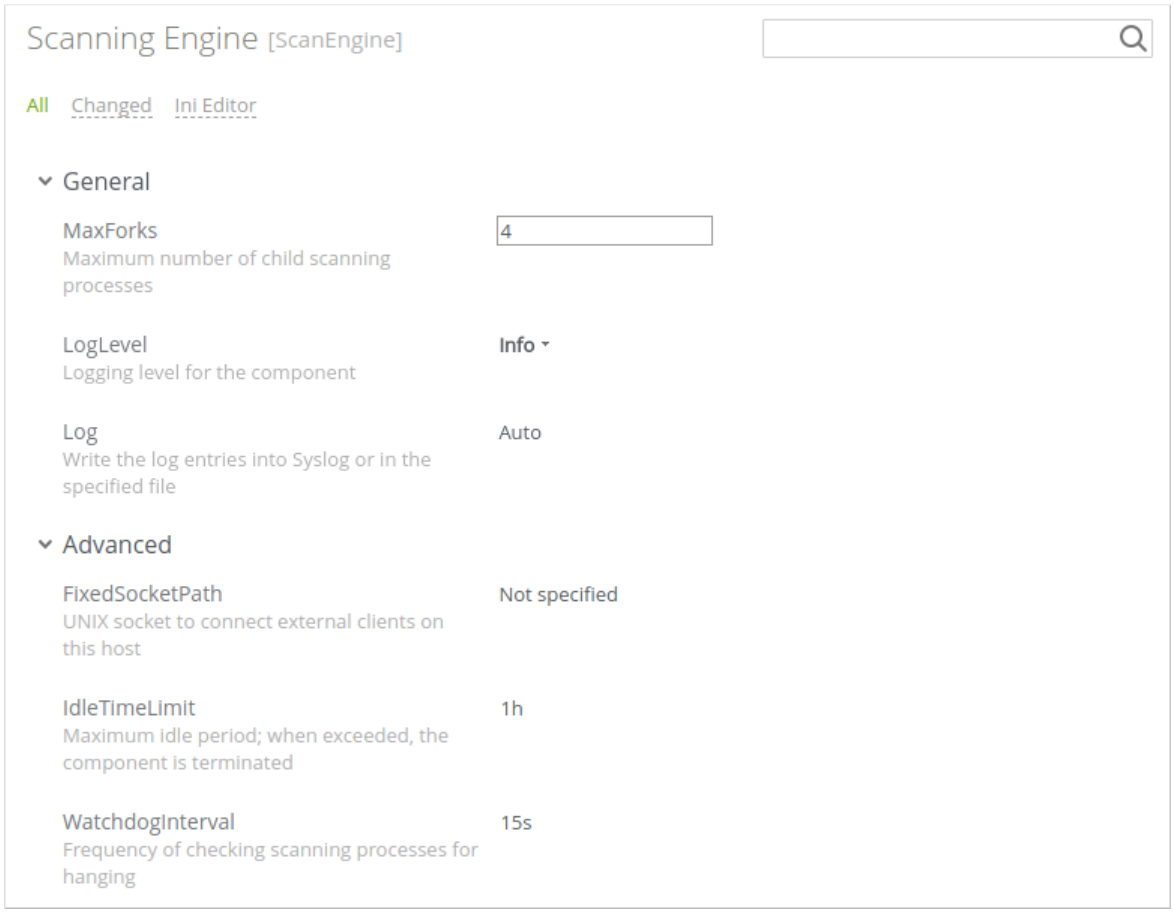


Figure 3. Component settings in tabular form

If the parameter accepts a string value or a list of arbitrary values, a pop-up window will appear once you click the parameter current value to edit it. If the parameter accepts a list of values, they are displayed in a multiline editing field (one value per line) as shown in the figure below. To edit the values of the list, change, delete or add required lines in the editing field.

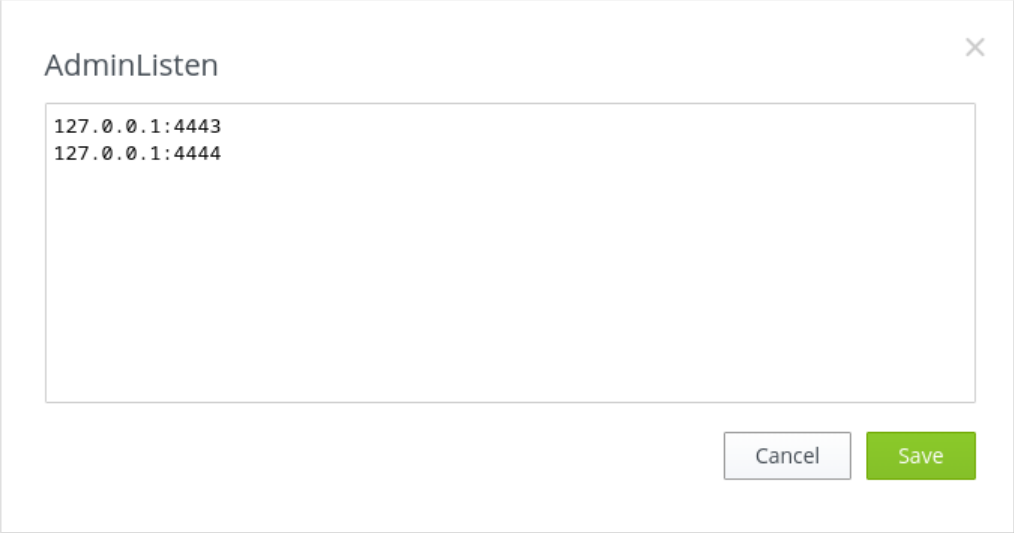


Figure 4. Editing a list of values

After editing a parameter value, click **Save** to save your changes and to close the window. To close the window without saving the changes, click **Cancel** or ✕ in the upper right corner of the pop-up window.

Viewing and editing component settings in the .ini editor

When viewing parameters in **Ini Editor** mode, they are displayed in an editor for parameters in the configuration file format (in `parameter = value` pairs), where `parameter` is a parameter name that is set directly in a component settings section of the configuration file. In this mode, only those parameters are displayed whose values differ from the defaults (that is, the parameters whose values are in bold in the **All** table). The figure below displays parameters in pairs.

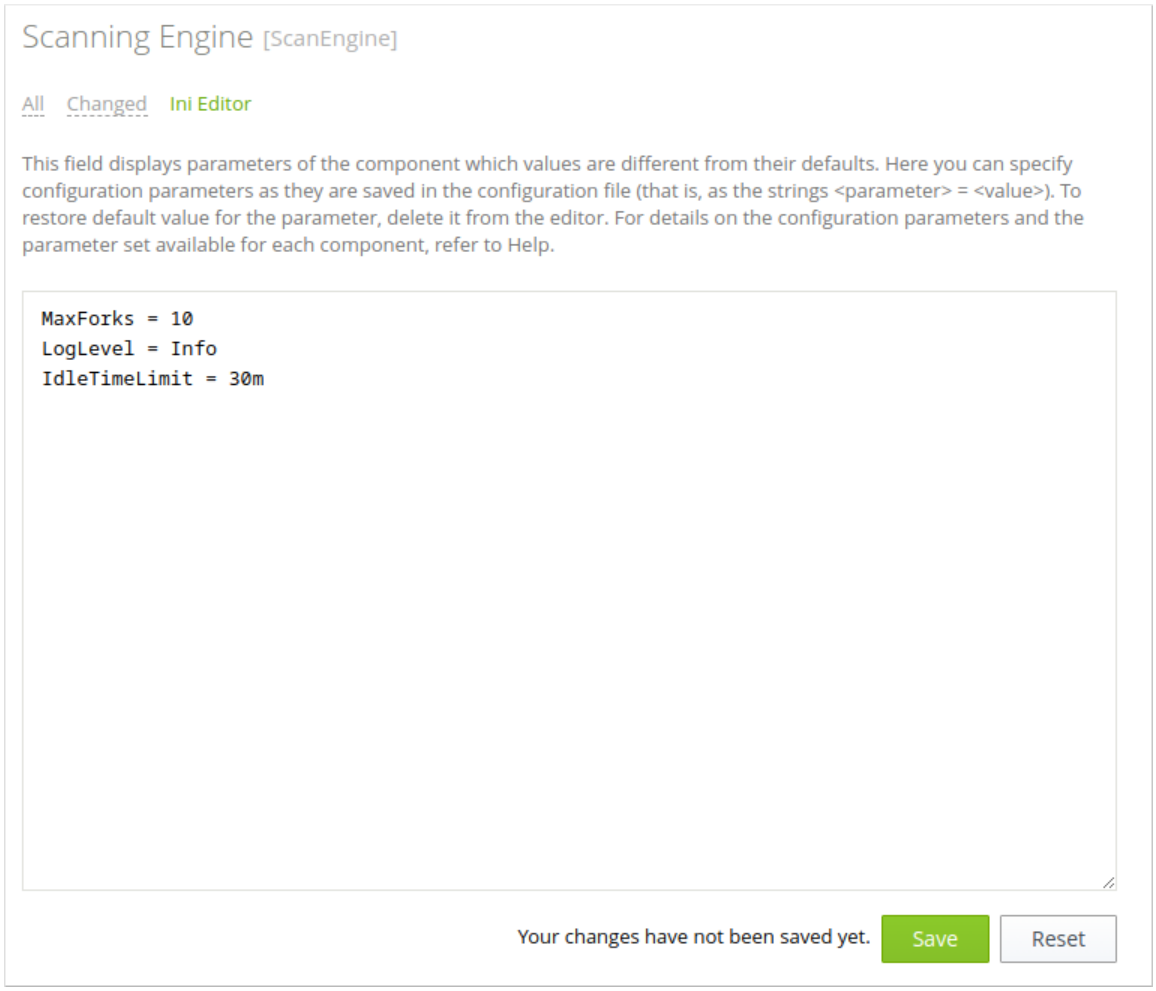


Figure 5. Editing settings in the configuration file format

To make changes, edit the text in accordance with the rules for editing the configuration file (this will modify only the section that contains the settings of the component selected to the left). If necessary, you can specify a new value for any parameter available for the component. In this case, the default value of this parameter changes to the value you enter in the editor. If you want to reset the parameter back to its default value, delete the line containing this parameter in this editor. If you do so, once you save the changes, the parameter is reset to its default value.

Once you have finished editing, click **Save** to save the changes. To discard the changes, click **Cancel**.

If you click **Save**, the text displayed in the editor is validated: the program checks whether all parameters are existent and their set values are valid. In case of an error, a corresponding message is displayed.

For details on the configuration file, its structure and aspects of specifying parameter values, refer to the Appendix D. Dr.Web Industrial for Unix File Servers Configuration File section.

Additional information

- Configuration parameters of the Dr.Web ConfigD component (general settings).
- Configuration parameters of the SpIDer Guard component.
- Configuration parameters of the SpIDer Guard for SMB component.
- Configuration parameters of the Dr.Web ES Agent component.
- Configuration parameters of the Dr.Web Updater component.
- Configuration parameters of the Dr.Web File Checker component.
- Configuration parameters of the Dr.Web Scanning Engine component.
- Configuration parameters of the Dr.Web Network Checker component.
- Configuration parameters of the Dr.Web SNMPD component.
- Configuration parameters of the Dr.Web StatD component.
- Managing Centralized Protection.

9.3.3.1. Managing Centralized Protection

You can connect Dr.Web Industrial for Unix File Servers to a centralized protection server or disconnect from it, thereby switching the product to the standalone mode. To open a page where you can manage centralized protection, select **Centralized protection** from the settings menu on the **Settings** page.

To connect Dr.Web Industrial for Unix File Servers to a centralized protection server or disconnect from it, use the corresponding check box on this page.

Connection to a centralized protection server

At an attempt to connect to a centralized protection server a pop-up window appears on the screen; in this window you need to specify the parameters for connecting to the centralized protection server.

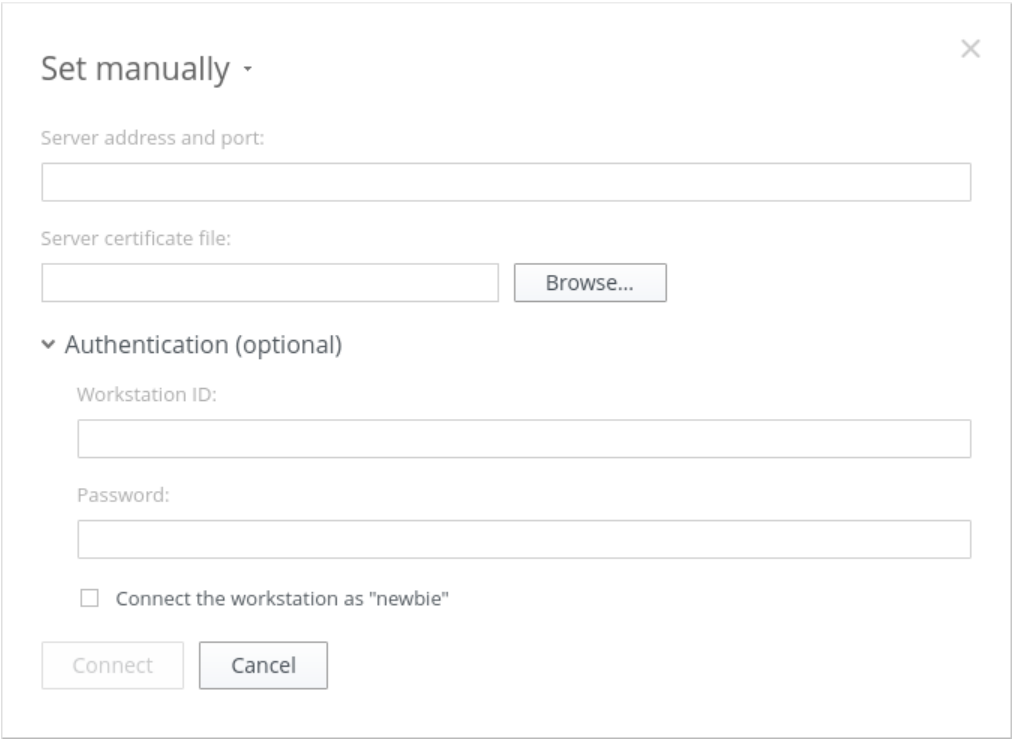


Figure 6. Connection to the centralized protection server

Select a method for connecting to the server from the drop-down list located at the top of the window. Three methods are available:

- *Load from file*,
- *Set manually*,
- *Detect automatically*.

If you select the *Load from file* option, specify a path to a file with connection settings in the corresponding field of the window. The file is provided by an antivirus network administrator. If you select the *Set manually* option, specify an address and a port to connect to the centralized protection server. In addition, for the *Set manually* or *Detect automatically* options, you can also specify a path to a file containing a server public key if you have one (usually, this file is provided by an antivirus network administrator or an internet service provider).

Furthermore, in the **Authentication (optional)** section you can specify a workstation identifier and a password for authentication on the server, if you know them. If these fields are filled in, your connection to the server will be established only if a correct identifier/password pair was provided. If you leave these fields empty, connection to the server is established only if it is approved by the server (either automatically or by the antivirus network administrator depending on the server settings).

Furthermore, you can select the **Connect the workstation as "newbie"** check box. If the newbie option is allowed on the server and the connection is approved, the server automatically generates a unique identifier/password pair to be further used for connecting your computer to this server.

If you connect as a newbie, the centralized protection server generates a new account for your computer even if it already had an account on this server.

Connection parameters must be specified in strict conformity with instructions provided by the administrator of your antivirus network or service provider.

To connect to the server, specify all the parameters, click **Connect** and wait for the connection to be established. To close the window without establishing the connection to the server, click **Cancel**.

Once you have connected Dr.Web Industrial for Unix File Servers to the centralized protection server, its operation is managed by the centralized protection server until you switch back to the standalone mode. Connection to the server is established automatically every time Dr.Web Industrial for Unix File Servers is started.

9.3.4. Scanning Local Files

In this section

- Opening a panel to scan local files and adjusting scanning parameters
- Starting a scan of local files

The web interface allows you to quickly scan files stored on your local computer used to access the web interface. The scanning engine included in Dr.Web Industrial for Unix File Servers is used for scanning. The files selected for scanning are uploaded via HTTP to a server, but after the scanning, even if threats have been found, the files are not stored on the server and are not quarantined. The user who sent the files for scanning is only informed about its result.

Opening a panel to scan local files and adjusting scanning parameters

You can select and upload the files for scanning via a panel for scanning local files, which is displayed when you select the **Scan File** item on the main menu of the web interface. The activated panel is displayed in the bottom right corner of the web interface. The figure below shows what the panel for scanning local files looks like.

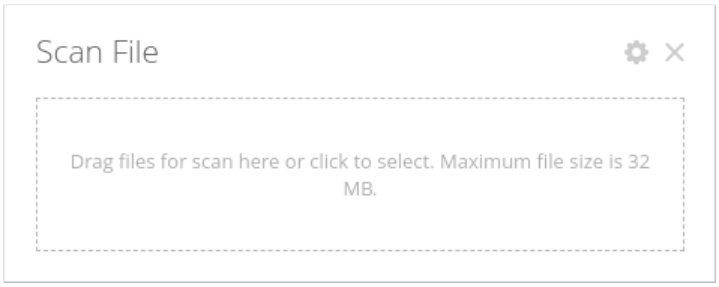


Figure 7. Panel for scanning local files

To close this panel, click ✕ in the top right corner of the panel. Click ⚙ to display the settings for the scanning of local files, where you can set the maximum time to scan a file (which does not include the time for uploading the file to the server from the local computer), enable or disable the heuristic analysis, and also set the maximum compression ratio for compressed objects and the maximum nesting level for objects packed into containers (such as archives).

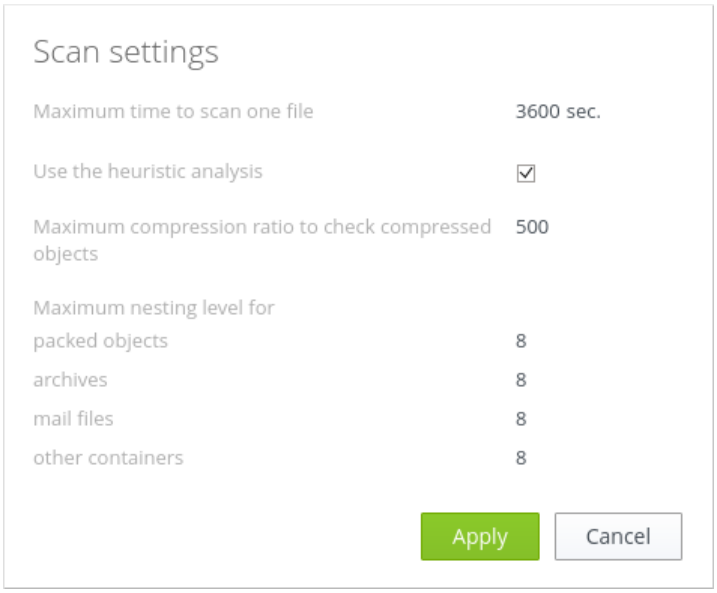


Figure 8. Adjusting the parameters for the scanning of local files

To apply the adjusted settings and to return to the file selection mode, click **Apply**. To go back to file selection without applying changes, click **Cancel**.

Starting a scan of local files

To start file scanning, click the target area **Drag files for scan here or click to select**; at that, a standard file chooser of your file manager opens. You can select multiple files at once for scanning.



You cannot select directories for scanning.

In addition, you can drag selected files with your mouse directly to the target area from the file manager window. Once the files to be scanned have been specified, they are uploaded to the server running Dr.Web Industrial for Unix File Servers and scanned after uploading. During the uploading and scanning of the files, the file scanning panel displays the overall scanning progress.

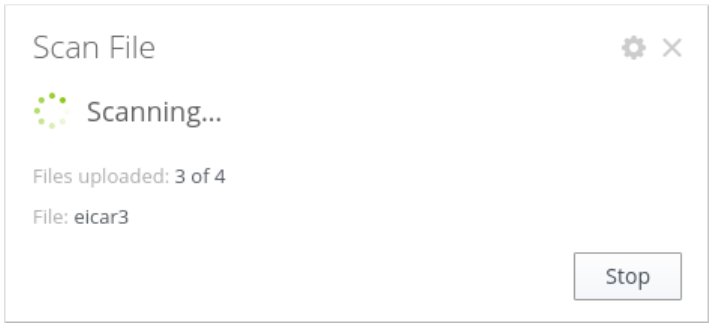


Figure 9. Scanning local files

If necessary, you can abort uploading and scanning by clicking **Stop**. Once the scanning is complete, a report about the scanning of the uploaded files is displayed on the panel.

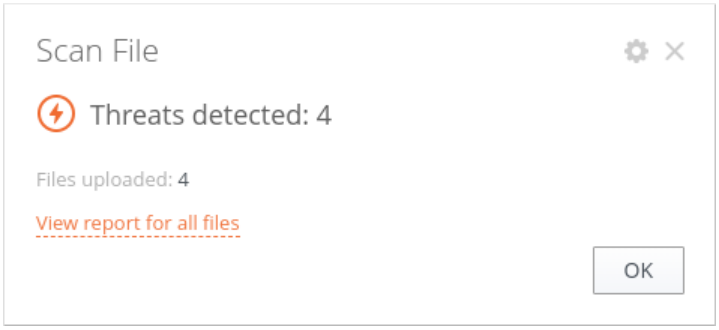


Figure 10. Result of the scanning of local files

If multiple files have been uploaded, an extended report about the scanning will be available. To view the extended report, click **View report for all files**.

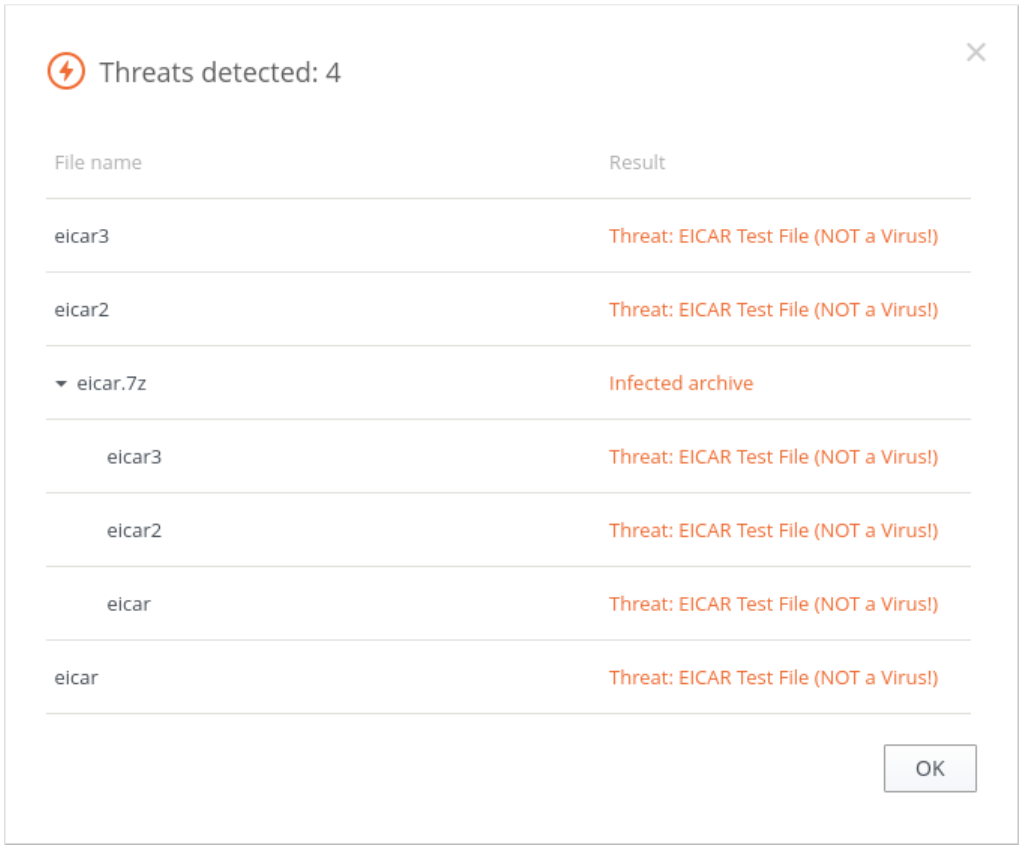


Figure 11. Extended report on scanned local files

To close the report and return to selecting new files for scanning, click **OK**.

You can start scanning local files with the current settings even if the file scanning panel is closed. To start uploading and scanning local files, just drag them from the file manager window to the web interface page opened in your browser.

9.4. SpIDer Guard



This component is included only in the distributions designed for the OSes of the GNU/Linux family.

The SpIDer Guard file system monitor is designed for monitoring file activity on GNU/Linux file system volumes. The component operates as a resident monitor and controls main file system events related to file modification (creating, opening, closing). When such an event is intercepted, the monitor checks whether the file was modified and, if so, the module generates a task for the Dr.Web File Checker file scanning component to scan the modified file with Dr.Web Scanning Engine.

Moreover, the SpIDer Guard file system monitor detects attempts to run executable files. If a program in an executable file is considered malicious during scanning, all processes started from this file will be forcibly terminated.

9.4.1. Operating Principles

In this section

- General information
- Defining file system areas to be monitored
- Enhanced file monitoring mode

General information

The SpIDer Guard file system monitor operates in user space (*user mode*) using the fanotify system mechanism. It is recommended that you use the *automatic mode* (`Auto`), which will allow the component to determine and use the best operation mode at startup, because not all Linux kernel versions support fanotify used by the monitor. If the component cannot support the specified integration mode, the components exits after startup. If the auto mode is selected, the component attempts to use the fanotify mode. If this mode can be used, the component shuts down.

Once new or modified files are detected, the monitor sends a task to the Dr.Web File Checker component to scan these files. The component then initiates their scanning by Dr.Web Scanning Engine. The operation scheme is shown in the picture below. When operating via the fanotify system mechanism, the monitor can block access to files (of all types of files or only executable files—PE, ELF, scripts with the `#!` preamble) that are not scanned yet until they are scanned (see below).

SpIDer Guard automatically detects mounting and unmounting new file system volumes (for example, on USB flash drives, CD/DVD, RAID arrays, and so on) and adjusts the list of monitored objects, if necessary.

Defining file system areas to be monitored

To optimize file system scanning, SpIDer Guard controls requests only to those files that are in the file system scopes specified in the configuration. Each scope is defined as a path to some directory of the file system tree and is called a *protected space*. A set of all protected spaces forms a single *monitoring scope* controlled by the monitor. Besides the monitoring scope, you can specify a set of file system directories to be excluded from monitoring in the component settings (*exclusion scope*). If you did not specify any protected space in the component settings, the monitoring scope covers the entire file system tree. Thus, only those files are monitored which paths are covered by the monitoring scope and are not covered by the exclusion scope.

Specifying exclusion can be useful when, for example, some files are frequently modified, which results in constant repeated scanning of these files thus increasing the system load. If it is known with certainty that frequent modification of files in a directory is not caused by a malicious program but is due to operation of a trusted program, you can add the path of this directory or files modified in it to the list of exclusions. In this case, the SpIDer Guard file system monitor stops responding to modification of these objects, even if they are covered by the monitoring scope. Moreover, you can add the program processing the files to the list of trusted programs (the `ExcludedProc` configuration parameter). In this case, file operations of this program will not result in scanning even if they are covered by the monitoring scope. Similarly, if necessary, you can disable monitoring and scanning of files from other file systems that are mounted on the local file system (for example, external file server directories mounted via CIFS). To specify file systems which files should not be scanned, use the `ExcludedFilesystem` parameter.

Protected spaces, as parts of the monitoring scope with scan parameters specified for them, are set in the component settings as named sections, which names contain a unique identifier assigned to the protected space. Each section describing a space contains the `Path` parameter that defines a path in the file system storing directories of this protected space (i.e. the file system tree fragment that is monitored within this space) and the `ExcludedPath` parameter that defines a local (with respect to `Path`) exclusion scope inside the protected space. The `ExcludedPath` parameter can contain standard file masks (i.e. `*` and `?` characters). Besides local exclusion scopes, you can also specify a global exclusion scope by using the `ExcludedPath` parameter specified outside the sections describing protected spaces. All directories covered by this scope, including the directories of protected spaces, are excluded from monitoring. Only global and individual exclusion scopes can be applied to each protected space: if a space is nested in another, the exclusion settings specified for the enclosing space are not applied to the nested space. In addition, each protected scope settings have the `Enable` boolean parameter which determines whether the files covered by this space are monitored. If this parameter is set to `No`, the contents of this space is not monitored. Moreover, the protected space is not monitored if the `Path` parameter has an empty value.

If all protected spaces specified in the monitor settings are not monitored or their paths are not specified, SpIDer Guard is running idle because none of the files of the file system tree are monitored. If you want to monitor the file system as a single protected space, remove all named protected space sections from the settings.

Consider an example of configuring monitor and exclusion scopes. Suppose the following parameters are set in the component settings:

```
[LinuxSpider]
ExcludedPath = /directory1/tmp
...

[LinuxSpider.Space.space1]
Path = /directory1
ExcludedPath = "*.tmp"
...

[LinuxSpider.Space.space2]
Path = /directory1/directory2
...

[LinuxSpider.Space.space3]
Path = /directory3
Enable = No
...
```

This means that the files in the `/directory1` directory and its subdirectories, except the `/directory1/tmp` directory, are monitored. Moreover, the files which full names match the `/directory1/*.tmp` mask are not monitored (except the `/directory1/directory2` nested scope to which this mask is not applied despite the fact that the scope is nested in the *space1* protected space). Files in the `/directory3` directory are not monitored.

Enhanced file monitoring mode

SpIDer Guard can use three monitoring modes:

- *Regular* (set by default)—monitors file access operations (creation, opening, closing, and running). Scanning of a file, access to which has been allowed, is requested. If a threat is detected upon the scan, the action to neutralize the threat can be applied to the file. Applications are allowed to access the file until the file scanning is finished.
- *Enhanced control of executable files*—monitor files considered non-executable as in regular mode. SpIDer Guard blocks access to an executable file until its scanning is finished.

  Executable files are binary files of PE and ELF formats, as well as text script files containing the "`#!`" preamble.

- *"Paranoid" mode*—SpIDer Guard blocks access to any file until its scanning is finished.

Dr.Web File Checker stores file scan results in specialized cache for a certain time, so reaccessing the same file does not lead to rescanning it if there is information in the cache; the

result extracted from the cache is therefore displayed as the scan result. Despite this, the use of the "paranoid" monitoring mode leads to a significant slowdown in accessing files.

To configure the monitoring mode, change the `BlockBeforeScan` parameter value in the component settings.



See the Configuring File System Monitoring section for more information about configuring SpIDer Guard and file monitoring modes.

9.4.2. Command-Line Arguments

To start the SpIDer Guard component from the command line, run the command:

```
$ /opt/drweb.com/bin/drweb-spider [<parameters>]
```

SpIDer Guard accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-spider --help
```

This command outputs short help information about the SpIDer Guard component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon at the startup of the operating system. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-spider` command.

9.4.3. Configuration Parameters

In this section

- Component parameters
- Customizing protected space individual monitoring settings

The component uses configuration parameters specified in the `[LinuxSpider]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: `/opt/drweb.com/bin/drweb-spider`
`Start` *{boolean}*	The component is started by the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration management daemon to start the component immediately, and setting this parameter to `No`—to shut down the component immediately. Default value: Depends on the Dr.Web product as part of which the component is supplied.
`[*] ExcludedPath` *{path to file or directory}*	Path to an object (file or directory) to be excluded from file monitoring. Either an individual file or an entire directory can be specified. If a directory is specified, all files and

Parameter	Description
	subdirectories (including nested ones) will be skipped. You can use file masks (containing characters `?` and `*` as well as character classes `[ ]`, `[! ]` and `[^ ]`). Multiple values can be specified as a list. List values must be comma-separated and put in quotation marks. The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add the `/etc/file1` file and the `/usr/bin` directory to the list. • Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset LinuxSpider.ExcludedPath -a /etc/file1` `# drweb-ctl cfset LinuxSpider.ExcludedPath -a /usr/bin` • Adding values to the configuration file. ○ Two values per line: `[LinuxSpider]` `ExcludedPath = "/etc/file1", "/usr/bin"` ○ Two lines (one value per line): `[LinuxSpider]` `ExcludedPath = /etc/file1` `ExcludedPath = /usr/bin` To apply the changes, reload the Dr.Web Industrial for Unix File Servers configuration by running the command: `# drweb-ctl reload` There is no point in providing paths to symbolic links here as only a direct path to a file is analyzed while scanning it. Default value: `/proc, /sys`
`Mode` *{FANOTIFY | AUTO}*	SplDer Guard operation mode. Allowed values: • `FANOTIFY`—use the *fanotify* monitoring interface; • `AUTO`—select an optimal operation mode automatically. Default value: `AUTO`
`ExcludedProc` *{path to file or path list}*	List of processes which file activity is not monitored. If a file operation was initiated by one of the processes specified in

Parameter	Description
	the parameter value, the modified or created file will not be scanned. Multiple values can be specified as a list. List values must be comma-separated and put in quotation marks. The parameter can be specified more than once in the section (in this case, all its values are combined into one list). You can use file masks (containing characters `?` and `*` as well as character classes `[ ]`, `[! ]` and `[^ ]`). Example: Add the `wget` and `curl` processes to the list. • Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset LinuxSpider.ExcludedProc -a /usr/bin/wget` `# drweb-ctl cfset LinuxSpider.ExcludedProc -a /usr/bin/curl` • Adding values to the configuration file. ○ Two values per line: `[LinuxSpider]` `ExcludedProc = "/usr/bin/wget", "/usr/bin/curl"` ○ Two lines (one value per line): `[LinuxSpider]` `ExcludedProc = /usr/bin/wget` `ExcludedProc = /usr/bin/curl` To apply the changes, reload the Dr.Web Industrial for Unix File Servers configuration by running the command: `# drweb-ctl reload` Default value: *(not specified)*
`BlockBeforeScan` *{Off | Executables | All}*	Block files while being accessed until they are scanned by the monitor (an enhanced or “paranoid” monitoring mode). Allowed values: • `Off`—do not block access to files even if they were not scanned. • `Executables`—block access to executable files (PE and ELF files and scripts containing the `#!` preamble) not scanned by the monitor. • `All`—block access to all files not scanned by the monitor. Files are blocked only in `FANOTIFY` mode. Default value: `Off`
`ExcludedFilesystem`	File system accessing the files of which will not be monitored.

Parameter	Description
{file system name}	This option is available only in `FANOTIFY` mode. Multiple values can be specified as a list. List values must be comma-separated and put in quotation marks. The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add the `cifs` and `nfs` file systems to the list. • Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset LinuxSpider.ExcludedFilesystem -a cifs` `# drweb-ctl cfset LinuxSpider.ExcludedFilesystem -a nfs` • Adding values to the configuration file. ○ Two values per line: `[LinuxSpider]` `ExcludedFilesystem = "cifs", "nfs"` ○ Two lines (one value per line): `[LinuxSpider]` `ExcludedFilesystem = cifs` `ExcludedFilesystem = nfs` To apply the changes, reload the Dr.Web Industrial for Unix File Servers configuration by running the command: `# drweb-ctl reload` Default value: `cifs`
`[*] OnKnownVirus` *{action}*	Action to be applied upon detection of a known threat (a virus and so on) in the scanned file. Allowed values: `REPORT`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnIncurable` *{action}*	Action to be applied upon detection of an incurable threat. Allowed values: `QUARANTINE`, `DELETE`. Default value: `QUARANTINE`
`[*] OnSuspicious` *{action}*	Action to be applied upon detection of an unknown threat (or a suspicious object) in the scanned file by using heuristic analysis. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`

Parameter	Description
`[*] OnAdware` *{action}*	Action to be applied upon detection of adware in the scanned file. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnDialers` *{action}*	Action to be applied upon detection of a dialer in the scanned file. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnJokes` *{action}*	Action to be applied upon detection of a joke program in the scanned file. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnRiskware` *{action}*	Action to be applied upon detection of riskware in the scanned file. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnHacktools` *{action}*	Action to be applied upon detection of a hacktool in the scanned file. Allowed values: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] ScanTimeout` *{time interval}*	Timeout for scanning one file. Allowed values: from 1 second (`1s`) to 1 hour (`1h`). Default value: `30s`
`[*] HeuristicAnalysis` *{On | Off}*	Enable or disable the heuristic analysis for detection of unknown threats. The heuristic analysis provides higher detection reliability but increases the duration of scanning. Action applied to threats detected by the heuristic analyzer is specified by the `OnSuspicious` parameter. Allowed values: • `On`—enable the heuristic analysis while scanning. • `Off`—disable the heuristic analysis. Default value: `On`

Parameter	Description
`[*] PackerMaxLevel` *{integer}*	Maximum nesting level for packed objects. A packed object is executable code compressed with special software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects that may also include packed objects and so on. The value of this parameter specifies a nesting limit beyond which packed objects inside other packed objects are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[*] ArchiveMaxLevel` *{integer}*	Maximum nesting level for archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `0`
`[*] MailMaxLevel` *{integer}*	Maximum nesting level for files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `0`
`[*] ContainerMaxLevel` *{integer}*	Maximum nesting level while scanning other types of objects containing nested objects (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects will not be scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[*] MaxCompressionRatio` *{integer}*	Maximum compression ratio of scanned objects (a ratio of an uncompressed size to a compressed size). If the ratio of an object exceeds the limit, this object is skipped during the scan. The compression ratio must be no less than `2`. Default value: `500`

Parameter	Description
`DebugAccess` *{boolean}*	Log or do not log detailed information on file access attempts at the debug level (with `LogLevel = DEBUG`). Default value: `No`

Customizing protected space individual monitoring settings

For each protected space of a file system, a separate section containing the path to a monitored file system area and monitoring parameters is specified in the configuration file together with the `[LinuxSpider]` section, which stores all the monitor parameters. Each section must be named as `[LinuxSpider.Space.`<*space name*>`]`, where <*space name*> is a unique identifier of the protected space.

The space individual section must contain the following parameters absent in the `[LinuxSpider]` general section:

Parameter	Description
`Path` *{path to directory}*	Path to a directory with files to be monitored (including nested directories). By default, this parameter has an empty value; therefore, you must specify a value when adding the protected space to the monitoring scope. Default value: *(not specified)*
`Enable` *{boolean}*	Contents of the protected space located at `Path` must be monitored. To stop monitoring the contents of this protected space, set the parameter to `No`. Default value: `Yes`



If all protected spaces specified in the monitor settings are not monitored or their paths are not specified, SpIDer Guard is running idle because none of the files of the file system tree are monitored. If you want to monitor the file system as a single protected space, remove all named protected space sections from the settings.

Except for those mentioned above, separate sections of protected spaces can include a list of parameters from the general section of the component settings that are marked with the `[*]` designation in the table above and redefine a parameter specified for the protected space (for example, an action to be applied upon threat detection, the maximum archive nesting level and so on). If the parameter is not specified for the protected space, file monitoring for this space is adjusted with the corresponding parameter values from the `[LinuxSpider]` section.

To add a new section of parameters for the protected space with the <*space name*> tag using the Dr.Web Ctl management tool (started by running the `drweb-ctl` command), run the command:

```
# drweb-ctl cfset LinuxSpider.Space -a <space name>
```

Example:

```
# drweb-ctl cfset LinuxSpider.Space -a Space1
# drweb-ctl cfset LinuxSpider.Space.Space1.Path /home/user1
```

The first command adds the `[LinuxSpider.Space.Space1]` section to the configuration file; the second one sets a value of the `Path` parameter for the section by specifying the path to the monitored file system area. Other parameters of this section will be the same as in the `[LinuxSpider]` general section.

9.5. SpIDer Guard for SMB

SpIDer Guard for SMB is a monitor of file system directories used by the SMB file server Samba as shared directories. This component is designed to monitor actions applied to files in Samba shared directories. It operates as a resident monitor and controls basic actions in the protected file system (creation, opening, closing, as well as read and write operations). Once the component intercepts such operation, it checks whether the file was modified and if so, a task to scan the file is created and sent to the Dr.Web File Checker file scanning component. If the file requires scanning, Dr.Web File Checker, in its turn, initiates scanning of the file contents by Dr.Web Scanning Engine. If the file contains a threat, it is blocked for access for a period specified in settings until the threat is neutralized. Moreover, the component can be set up to block the file in case of a scanning error (including cases when there is no valid license).



To avoid potential conflicts between SpIDer Guard and SpIDer Guard for SMB, which may occur in the process of scanning files in Samba shared directories, it is recommended that you additionally configure SpIDer Guard by performing one of the following actions:

- exclude the Samba shared directories (specify them in the `ExcludedPath` parameter);
- add the Samba process (`smbd`) to the list of ignored processes (specify `smbd` in the `ExcludedProc` parameter).



The SpIDer Guard for SMB monitor uses a custom VFS SMB module to integrate with Samba. Several versions of the VFS SMB module built for different versions of Samba are supplied with the SpIDer Guard for SMB component; however, they may be incompatible with the version of Samba installed on your file server, for example, if your Samba server uses the `CLUSTER_SUPPORT` option.

If VFS SMB modules are incompatible with your Samba server, build the VFS SMB module for your Samba server with the `CLUSTER_SUPPORT` option if necessary. The procedure of building the VFS SMB module from source code is described in the Building the VFS SMB Module section.

9.5.1. Operating Principles

In this section

- General information
- Specifying file system areas to be monitored

General information

The SpIDer Guard for SMB monitor operates in daemon mode (usually it is started by the Dr.Web ConfigD configuration management daemon on system startup). After the startup, the component operates as a server to which custom plugins (VFS SMB modules) are connected that operate on the Samba server side and monitor user activity in shared directories. Once new

or modified files are found, the monitor requests their scanning with the Dr.Web File Checker component.

If a file scanned at request of the monitor is infected with an incurable threat or with a threat for which the "Block" (`BLOCK`) action is specified, the monitor instructs the VFS SMB module controlling the shared directory to block this file (that is, to prevent users from reading, writing and executing the file). Furthermore, a text file describing a reason for blocking is created next to the blocked file, if this setting is enabled. This is done to avoid the "unexpected disappearance" of the file to which the "Delete" (`DELETE`) or "Quarantine" (`QUARANTINE`) action was applied. Furthermore, this prevents the user (or the worm that infected the computer) from multiple attempts to recreate the moved or deleted file. Moreover, this text file also notifies the user that the computer is probably infected with a malicious program. If informed of this, the user can start anti-virus scanning of the computer to detect and neutralize local threats. In addition, the file (depending on a value of the corresponding configuration parameter) can be blocked upon a scanning error, including the absence of a valid license that enables operation of SpIDer Guard for SMB.

Specifying file system areas to be monitored

You can disable monitoring of specified files and directories stored in controlled shared directories of the Samba server. This can be useful when some files are frequently modified, which results in constant repeated scanning of these files and thus can cause high system load. If it is known with certainty that frequent modification is typical for some files in the storage, it is recommended that you add them to the list of exclusions. In this case, the monitor ignores modification of these files, and their scanning with the file scanning component is not initiated.

To distinguish between directories to be monitored and those to be skipped, the file storage monitor for Samba—SpIDer Guard for SMB—uses two configuration parameters:

- `IncludedPath`—list of paths to be monitored (monitoring scope);
- `ExcludedPath`—list of paths to be excluded from monitoring (exclusion scope).

Normally, the monitoring scope covers the entire shared directory. If you specify both monitoring and exclusion scopes, only those files in the shared directory are monitored whose paths are not specified in the `ExcludedPath` parameter or are specified in the `IncludedPath` parameter. If the same path is specified in both parameters, the `IncludedPath` parameter has a priority over the other one: the objects whose paths are included will be controlled by the SpIDer Guard for SMB monitor. Thus, you can use the `IncludedPath` parameter to add some files and directories to monitoring even if they are covered by the exclusion scope.

You can specify various protection parameters for different Samba shared directories protected by the SpIDer Guard for SMB monitor, including different monitoring and exclusion scopes as well as reactions to detected threats. For that purpose, in the SpIDer Guard for SMB configuration section, specify individual settings for VFS SMB modules that control these shared directories.

Refer to the Integration with a Samba File Server section for information about integrating Dr.Web Industrial for Unix File Servers with the file service.

9.5.2. Command-Line Arguments

To start the SpIDer Guard for SMB component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-smbspider-daemon [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-smbspider-daemon [<parameters>]
```

SpIDer Guard for SMB accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-smbspider-daemon --help
```

This command outputs short help information about the SpIDer Guard for SMB component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon at the startup of the operating system. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-smbspider-daemon` command.

9.5.3. Configuration Parameters

In this section

- Component parameters
- Customizing monitoring settings

The component uses configuration parameters specified in the `[SMBSpider]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-smbspider-daemon` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-smbspider-daemon`
`Start` *{boolean}*	The component is started by the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration daemon to start the component immediately, whereas setting this parameter to `No` instructs the configuration daemon to terminate the component immediately.

Parameter	Description
	Default value: `No`
`[*] ExcludedPath` *{path to file or directory}*	Path to a shared directory object to be skipped with all nested directories and files. A path relative to the root directory of the SMB file storage (see the `SambaChrootDir` parameter) can be specified for either an individual file or an entire directory. You can use file masks containing `?` and `*` characters as well as `[ ]`, `[! ]` and `[^ ]` character classes. Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add the `etc/file1` file and the `usr/bin` directory to the list. 1. Adding values to the configuration file. • Two values per line: `[SMBSpider]` `ExcludedPath = "etc/file1", "usr/bin"` • Two lines (one value per line): `[SMBSpider]` `ExcludedPath = etc/file1` `ExcludedPath = usr/bin` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset SMBSpider.ExcludedPath -a etc/file1` `# drweb-ctl cfset SMBSpider.ExcludedPath -a usr/bin` If a directory is specified, all directory contents will be skipped. Default value: *(not specified)*
`[*] IncludedPath` *{path to file or directory}*	Path to a shared directory object that must be scanned. A path relative to the root directory of the SMB file storage (see the `SambaChrootDir` parameter) can be specified for either an individual file or an entire directory. You can use file masks containing `?` and `*` characters as well as `[ ]`, `[! ]` and `[^ ]` character classes. Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add the `etc/file1` file and the `usr/bin` directory to the list. 1. Adding values to the configuration file.

Parameter	Description
	• Two values per line: `[SMBSpider]` `IncludedPath = "etc/file1", "usr/bin"` • Two lines (one value per line): `[SMBSpider]` `IncludedPath = etc/file1` `IncludedPath = usr/bin` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset SMBSpider.IncludedPath -a etc/file1` `# drweb-ctl cfset SMBSpider.IncludedPath -a usr/bin` If a directory is specified, all directory contents including nested files and directories will be scanned. This parameter has priority over the `ExcludedPath` parameter; that is, if the same object (a file or a directory) is specified in both parameter values, this object will be scanned. Default value: *(not specified)*
`[*] AlertFiles` *{boolean}*	Create or do not create a text file with the explanation of the reason for blocking for each blocked object. The created file will be named *<name of the blocked file>*`.drweb.alert.txt`. Allowed values: • `Yes`—create files describing the reasons why the object was blocked; • `No`—do not create such files. Default value: `Yes`
`[*] OnKnownVirus` *{action}*	Action to be applied upon detection of a known threat. Allowed values: `REPORT`, `BLOCK`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`
`[*] OnIncurable` *{action}*	Action to be applied upon detection of an incurable threat. Allowed values: `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `QUARANTINE`
`[*] OnSuspicious` *{action}*	Action to be applied upon detection of a suspicious object. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `REPORT`

Parameter	Description
`[*] OnAdware` *{action}*	Action to be applied upon detection of adware. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `PASS`
`[*] OnDialers` *{action}*	Action to be applied upon detection of a dialer. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `PASS`
`[*] OnJokes` *{action}*	Action to be applied upon detection of a joke program. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `PASS`
`[*] OnRiskware` *{action}*	Action to be applied upon detection of riskware. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `PASS`
`[*] OnHacktools` *{action}*	Action to be applied upon detection of a hacktool. Allowed values: `PASS`, `REPORT`, `BLOCK`, `QUARANTINE`, `DELETE`. Default value: `PASS`
`[*] BlockOnError` *{boolean}*	Block or do not block access to a file if an attempt to scan it has failed or there is no license allowing to scan files with SpIDer Guard for SMB. If this parameter is set to `Yes` and there is no valid license, SpIDer Guard for SMB will block all files moved to a shared directory it protects. Allowed values: • `Yes`—block access to the file; • `No`—do not block access to the file. Default value: `Yes`
`[*] ScanTimeout` *{time interval}*	Timeout for scanning one file. Allowed values: from 1 second (`1s`) to 1 hour (`1h`). Default value: `30s`
`[*] HeuristicAnalysis` *{On | Off}*	Enable or disable the heuristic analysis for detection of unknown threats. The heuristic analysis provides higher detection reliability but increases the duration of scanning.

Parameter	Description
	The `OnSuspicious` parameter defines an action applied to threats detected by the heuristic analyzer. Allowed values: • `On`—enable the heuristic analysis during a scan; • `Off`—disable the heuristic analysis. Default value: `On`
`[*] PackerMaxLevel` *{integer}*	Maximum nesting level for packed objects. A packed object is executable code compressed with special software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects that may also include packed objects and so on. The value of this parameter specifies a nesting limit beyond which packed objects inside other packed objects are not scanned. All objects at a deeper nesting level are skipped during a scan initiated by SpIDer Guard for SMB. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[*] ArchiveMaxLevel` *{integer}*	Maximum nesting level for archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `0`
`[*] MailMaxLevel` *{integer}*	Maximum nesting level for files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[*] ContainerMaxLevel` *{integer}*	Maximum nesting level while scanning other types of objects containing nested objects (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects will not be scanned.

Parameter	Description
	The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[*] MaxCompressionRatio` *{integer}*	Maximum compression ratio of scanned objects (ratio between the compressed size and uncompressed size). If the ratio of an object exceeds the limit, this object is skipped during a scan initiated by SpIDer Guard for SMB. The compression ratio must be no less than `2`. Default value: `500`
`SmbSocketPath` *{path to file}*	Path to the socket file which enables interaction between SpIDer Guard for SMB and VFS SMB modules. The path is always relative and supplements the path specified as the `SambaChrootDir` parameter value (if the value is empty, then the `/` path to the file system root is supplemented). Default value: `var/run/.com.drweb.smb_spider_vfs`
`SambaChrootDir` *{path to directory}*	Path to the root directory of the SMB file storage (can be redefined by the file server using `chroot`). Used as a prefix inserted at the beginning of all paths to files and directories in the file server storage and describes a path to them relative to the local file system root. If not specified, the `/` path to the file system root is used. Default value: *(not specified)*
`ActionDelay` *{time interval}*	Delay time between the moment when a threat is detected and the moment when SpIDer Guard for SMB applies the action specified for this threat type. The file is blocked during this time period. Default value: `1d`
`MaxCacheSize` *{size}*	Size of cache used by VFS SMB modules to store information about scanned files in monitored SMB directories. If `0` is specified, the cache is not used. Default value: `10MB`

Customizing monitoring settings

You can specify a different tag for each VFS SMB module which monitors each shared directory (file storage). You can do that in the configuration file (typically, `smb.conf`) of the Samba SMB server. Unique tags for VFS SMB modules in the `smb.conf` file are specified as follows:

```
smb_spider:tag = <name>
```

where <*name*> is a unique tag assigned to a VFS SMB module, which controls some shared directory, by the Samba server.

If some VFS SMB module has a unique tag <*name*>, you can create a separate section in the configuration file of Dr.Web Industrial for Unix File Servers in addition to the `[SMBSpider]` section storing all SMB parameters. The created section will cover only parameters for scanning a particular file storage protected by the VFS SMB module to which the tag was assigned. This section should be named as follows: `[SMBSpider.Share.<name>]`.

Invididual sections for VFS SMB modules can contain parameters indicated with the character `[*]` in the table above. Other parameters cannot be specified in individual sections because these parameters are defined simultaneously for all VFS SMB modules operating with SMB directory monitor SpIDer Guard for SMB.

A VFS SMB module using an individual section `[SMBSpider.Share.<name>]` gets the values of all parameters not provided in this section from the general section `[SMBSpider]`. Thus, if there are no tagged individual sections, all VFS SMB modules will use the same parameters for protection of shared directories being monitored. If you delete some parameter from the `[SMBSpider.Share.<name>]` section, the parameter value for this section (and for the corresponding shared directory with the <*name*> tag) will be inherited from the corresponding "parent" parameter with the same name from the general `[SMBSpider]` section rather than from the default parameter.

To add a new section for the shared Samba directory with the <*name*> tag using the Dr.Web Ctl command-line tool for Dr.Web Industrial for Unix File Servers management (started using the `drweb-ctl` command), run the command:

```
# drweb-ctl cfset SmbSpider.Share -a <name>
```

Example:

```
# drweb-ctl cfset SmbSpider.Share -a AccountingFiles
# drweb-ctl cfset SmbSpider.Share.AccountingFiles.OnAdware QUARANTINE
```

The first command adds the `[SMBSpider.Share.AccountingFiles]` section to the configuration file, and the second command changes the `OnAdware` parameter value in this section. Thus, the added section will contain all parameters marked with the `[*]` character in the table above, moreover, the values of all parameters except for the `OnAdware` parameter specified in the command will coincide with the parameter values from the `[SMBSpider]` general section.

9.5.4. Building the VFS SMB Module

In this section

- General information
- Building the VFS SMB module

General information

If the Samba version used on your file server is incompatible with all supplied versions of the VFS SMB module used by SpIDer Guard for SMB, you will have to build this module manually from source code.

The source code of the VFS SMB module for SpIDer Guard for SMB is supplied in the additional package named `drweb-smbspider-modules-src` and is packed to the `tar.gz` archive. When installing the `drweb-smbspider-modules-src` package, an archive with source code is located in the `/usr/src/` directory and is named `drweb-smbspider-11.1.src.tar.gz`. If the archive is absent at the specified path, install the package (from the repository or by custom installation from the universal package, depending on the Dr.Web Industrial for Unix File Servers installation method).

Besides the source code of the VFS SMB module used by SpIDer Guard for SMB, you will also need the source code of the Samba version installed on your file server. If you do not have the source code, download it. To determine which Samba version is installed on your file server, run the command:

```
$ smbd -V
```



To build the VFS SMB module used by SpIDer Guard for SMB, the source code of the actual version of Samba installed on your file server must be used. Otherwise, correct operation of the SpIDer Guard for SMB is not guaranteed.

To build the module manually from the source code, superuser (usually the *root* user) privileges are required. For that purpose, use the `su` command to switch to another user or the `sudo` command to build the module as a different user.

Building the VFS SMB module

1. Unpack the archive with the module source code to any directory. For example, the command

   ```
   # tar -xf drweb-smbspider-11.1.src.tar.gz
   ```

   unpacks the source code to the directory containing the archive and creates a subdirectory with the archive file name.
2. Determine the version of Samba installed on your server and download its source code if necessary.
3. Determine if the version of Samba installed on your server uses the `CLUSTER_SUPPORT` option. To do that, run the command:

   ```
   $ smbd -b | grep CLUSTER_SUPPORT
   ```

   If your Samba server uses the `CLUSTER_SUPPORT` option, this command displays the `CLUSTER_SUPPORT` string.

4. Go to the directory with the Samba source code, configure (`./configure`) and build (`make`) the server. When configuring, specify the correct value of the `CLUSTER_SUPPORT` option. In case of issues with configuring and building the source code of Samba, refer to developer documentation.

   Building Samba from the source code is required only for the further correct building of the VFS SMB module used by SpIDer Guard for SMB. There is no need to replace an already installed Samba server with the one built from the source code.

5. When you have successfully built Samba, go to the directory with the VFS SMB module source code and run the command:

   ```
   # ./configure --with-samba-source=<path to Samba source directory> && make
   ```

   where *<path to Samba source directory>* is the path to the directory where Samba was built in the previous step.

6. When the VFS SMB module is successfully built, copy the `libsmb_spider.so` resulting file from the `.libs` subdirectory (created while building) to the directory with VFS modules of the Samba server (by default, for GNU/Linux, it is `/usr/lib/<architecture>-linux-gnu/samba/`) and rename it to `smb_spider.so`. To do that, run the command:

   ```
   # cp ./.libs/libsmb_spider.so /usr/lib/x86_64-linux-gnu/samba/smb_spider.so
   ```

7. After copying of the built VFS SMB module, you can delete the directories where the module and the Samba server were built.

After that, it is necessary to integrate Dr.Web Industrial for Unix File Servers with the Samba server the way it is described in the corresponding section of the Administrator Manual. In this case, at the first step of the integration, it is not necessary to create the `smb_spider.so` symbolic link in the directory of the VFS modules of the Samba server.

9.6. Dr.Web File Checker

The Dr.Web File Checker file scanning component is designed for scanning files and directories in the file system. It is used by other components of Dr.Web Industrial for Unix File Servers to scan file system objects. In addition, this component permanently registers all threats detected in the file system and also functions as a quarantine manager, as it manages the contents of the directories where isolated files are kept.

9.6.1. Operating Principles

The Dr.Web File Checker component is started with superuser (the *root* user) privileges and scans file system objects (files, directories and boot records).

Dr.Web File Checker indexes all scanned files and directories and stores data about scanned objects in a special cache to avoid repeated scanning of the objects that have been already scanned and have not been modified since that (in this case, if a request to scan such an object is received, the previous scan result retrieved from the cache is returned).

When requests to scan file system objects are received from other components, Dr.Web File Checker checks whether the requested object requires scanning. If so, a scanning task is generated for Dr.Web Scanning Engine. If the scanned object contains a threat, Dr.Web File Checker puts it in the registry of detected threats and applies an action to neutralize it (cure, delete or quarantine), if this action has been specified by the client component that initiated the scanning as a reaction to the threat. The scanning can be initiated by various components of Dr.Web Industrial for Unix File Servers (for example, by the SpIDer Guard for SMB monitor).

During scanning of the requested file system objects, the file-checking component generates a report detailing scanning results and applied actions to neutralize threats, if any, and sends this report to the client component that requested scanning.

Apart from the standard file scanning method, the following special methods are available for internal use:

- *The "flow" method*—a method for scanning files in stream. A component, which uses this method, initializes parameters of scanning and threat neutralization only once. These parameters are further applied to all file scanning requests from this component. This method is used by the SpIDer Guard monitor.
- *The "proxy" method*—a method for file scanning consisting in that the file-checking component only scans files for threats without applying any actions to them and without registering the detected threats (these actions are fully delegated to the component that initiated the scanning). This method is used by the SpIDer Guard for SMB monitor.

During its operation, the file-checking component not only maintains threat registry and manages quarantine, but also collects overall file scan statistics, averaging the number of files scanned per second for the last minute, last 5 minutes, last 15 minutes.

9.6.2. Command-Line Arguments

To start the Dr.Web File Checker component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-filecheck [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-filecheck [<parameters>]
```

Dr.Web File Checker accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-filecheck --help
```

This command outputs short help information about the Dr.Web File Checker component.

Startup notes

The component cannot be started directly from the command line in standalone mode. It is started automatically by the Dr.Web ConfigD configuration management daemon upon receiving requests for file system scanning from other components of Dr.Web Industrial for Unix File Servers. To manage the operation parameters of the component as well as to scan files on demand, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory using Dr.Web File Checker, use the `scan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl scan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-filecheck` command.

9.6.3. Configuration Parameters

The component uses configuration parameters specified in the `[FileCheck]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

This section stores the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-filecheck` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-filecheck`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`DebugClientIpc` *{boolean}*	Log or do not log detailed IPC messages at the debug level (with `LogLevel = DEBUG`). Default value: `No`

Parameter	Description
`DebugScan` *{boolean}*	Log or do not log detailed messages received during file scanning at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`DebugFlowScan` *{boolean}*	Log or do not log detailed messages about file scanning using the *"flow"* method at the debug level (with `LogLevel = DEBUG`). The *"flow"* method is usually used by the SpIDer Guard monitor. Default value: `No`
`DebugProxyScan` *{boolean}*	Log or do not log detailed messages about file scanning using the *"proxy"* method at the debug level (with `LogLevel = DEBUG`). The *"proxy"* method is usually used by the SpIDer Guard for SMB monitor. Default value: `No`
`DebugCache` *{boolean}*	Log or do not log detailed messages about the cache status of scanned files at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`MaxCacheSize` *{size}*	Maximum allowed size of cache to store data about scanned files. If `0` is specified, caching is disabled. Default value: `50mb`
`RescanInterval` *{time interval}*	Period of time during which a file will not be rescanned if the results of its previous scan are available in the cache (the period during which the stored information is considered up-to-date). Allowed values: from 0 seconds (`0s`) to 1 minute (`1m`). If the set interval is less than `1s`, there will be no delay—the file will be scanned upon any request. Default value: `1s`

9.7. Dr.Web Network Checker

The Dr.Web Network Checker component is designed to scan data received over the network, with the scanning engine, as well as for distributed file scanning for threats. The component allows to establish a connection between network hosts with Dr.Web Industrial for Unix File Servers installed on them for receiving and sending data (for example, file contents) via the network hosts for scanning. The component manages automatic distribution of scanning tasks (by sending and receiving them over the network) to all available network hosts with which the connection is configured. The component balances the load caused by the scanning tasks between the hosts. If there are no configured connections with remote hosts, the component sends all the data only to the local instance of Dr.Web Scanning Engine.



This component is always used to scan the data received over network connections. Thus, if the component is absent or unavailable, the operation of the components that send data for scanning over a network connection will be hindered.

This component is not designed to manage distributed scanning of files located in a local file system, since it cannot replace the Dr.Web File Checker component. To manage distributed scanning of local files, use the Dr.Web MeshD component.

If data scanning over the network is highly intensive, issues with scanning may arise when available file descriptors are depleted. In this case, it is necessary to increase the limit by the number of file descriptors available to Dr.Web Industrial for Unix File Servers.

Data can be shared either over an insecure channel or over a secure channel via SSL/TLS. To use a secure connection it is required to provide hosts that share files with valid certificates and SSL keys. To generate keys and certificates, you can use the `openssl` utility. An example of how to use the `openssl` utility to generate certificates and private keys is given in the Appendix E. Generating SSL Certificates section.

9.7.1. Operating Principles

The component allows sending the data not represented by files of a local file system for scanning to Dr.Web Scanning Engine located on a local or remote host. This data is processed by the components that send data for scanning over a network connection.



These components always use Dr.Web Network Checker (even if it is located on a local host) to send files for scanning to Dr.Web Scanning Engine. Thus, if Dr.Web Network Checker is unavailable, these components *cannot operate correctly*.

In addition, Dr.Web Network Checker allows to connect Dr.Web Industrial for Unix File Servers to a given set of hosts on the network that run Dr.Web Industrial for Unix File Servers (or any other Dr.Web for Unix solution 10.1 or later) in order to manage a distributed search for data not represented by files of the local file system. Thereby, this component allows to create and

configure a *scanning cluster*, which is a set of network hosts that exchange data for scanning (each host must run its own instance of the Dr.Web Network Checker distributed scanning agent). On each network host comprised by the scanning cluster, Dr.Web Network Checker performs automatic distribution of tasks for scanning data, sending them over the network to all available hosts for which the connection is configured. At the same time, the host load balancing caused by data scanning is performed depending on the amount of resources available on remote hosts. The number of child scanning processes generated by Dr.Web Scanning Engine on a host comprised by the cluster indicates the amount of resources available for load. The lengths of the queues of the files for scanning on each host in use are also considered.

In this case, any network host comprised by the scanning cluster can act both as a scanning client that sends data for remote scanning and as a scanning server that receives data from the specified network hosts for scanning. If necessary, the distributed scanning agent can be configured so that the host acts only as a scanning server or only as a scanning client.

The data received over the network for scanning is stored on the local file system as temporary files and sent to Dr.Web Scanning Engine or, in case it is unavailable or heavily loaded, to another host of the scanning cluster.

The `InternalOnly` parameter of the component settings allows to manage the Dr.Web Network Checker operation mode. The parameter defines if the component is used for including Dr.Web Industrial for Unix File Servers in the scanning cluster or only for internal purposes of the Dr.Web Industrial for Unix File Servers components operating locally.



You can create your own component (external application) which will use Dr.Web Network Checker to scan files (including distributing scanning tasks between the hosts of the scanning cluster). For this, the Dr.Web Network Checker component provides a custom API based on the Google Protobuf technology. The description of the Dr.Web Network Checker API , as well as client application sample code that uses Dr.Web Network Checker, are supplied as part of the `drweb-netcheck` package.

An example of creating a scanning cluster can be found in the Creating a Scanning Cluster section.

9.7.2. Command-Line Arguments

To start the Dr.Web Network Checker component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-netcheck [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-netcheck [<parameters>]
```

Dr.Web Network Checker accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-netcheck --help
```

This command outputs short help information about the Dr.Web Network Checker component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). At the same time, if a `FixedSocket` parameter value is specified in the component settings and the `InternalOnly` parameter is set to `No`, the component will be started by the configuration management daemon and always available to clients via a Unix socket. To manage component operation parameters and to start network scanning (if there is a configured connection to other network hosts), use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line. If there is no configured connection to other network hosts, normal scanning will be started using the local scanning engine instead of network scanning.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory with Dr.Web Network Checker, use the `netscan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl netscan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-netcheck` command.

9.7.3. Configuration Parameters

The component uses configuration parameters specified in the `[NetCheck]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-netcheck` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-netcheck`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. If the `LoadBalanceAllowFrom` or `FixedSocket` parameter is set, this setting is ignored (the component does not finish its operation after the time interval expires). Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`

Parameter	Description
`FixedSocket` *{path to file | address}*	Socket of the Dr.Web Network Checker agent fixed instance. If this parameter is specified, the Dr.Web ConfigD configuration management daemon ensures that there is always a running instance of the distributed scanning agent available to clients via this socket. Allowed values: • *<path to file>*—path to a local Unix socket; • *<address>*—network socket set as an *<IP address>*:*<port>* pair. ⚠ Ensure that the following permissions are assigned to the directory with the socket file: `drwxr-xr-x` To view the permissions, run the command: `$ ls -ld <path to directory>` Default value: *(not specified)*
`InternalOnly` *{boolean}*	Component operation mode. If the value is set to `Yes`, the component is used for internal purposes of Dr.Web Industrial for Unix File Servers components only, but not for servicing a scanning cluster or external (to Dr.Web Industrial for Unix File Servers) client applications regardless of `LoadBalance*` settings and the specified value of the `FixedSocket` parameter. Default value: `No`
`LoadBalanceUseSsl` *{boolean}*	Use or do not use SSL/TLS to connect to other hosts. Allowed values: • `Yes`—use SSL/TLS; • `No`—do not use SSL/TLS. If the parameter is set to `Yes`, a certificate and a private key must be specified for this host and for all hosts with which it interacts (the `LoadBalanceSslCertificate` and `LoadBalanceSslKey` parameters). Default value: `No`
`LoadBalanceSslCertificate` *{path to file}*	Path to the SSL certificate used by Dr.Web Network Checker on the current host for communication with other hosts via a secure SSL/TLS connection.

Parameter	Description
	 The certificate file and the private key file (specified by the `LoadBalanceSslKey` parameter) must match each other. Default value: *(not specified)*
`LoadBalanceSslKey` *{path to file}*	Path to the private key file used by Dr.Web Network Checker on the current host for communication with other hosts via a secure SSL/TLS connection.  The certificate file (specified by the `LoadBalanceSslCertificate` parameter) and the private key file must match each other. Default value: *(not specified)*
`LoadBalanceSslCa` *{path}*	Path to the directory or file with the list of trusted root certificates. Among these certificates, there must be a certificate that certifies the authenticity of the certificates used by agents within the scanning cluster when exchanging data via SSL/TLS protocols. If the parameter value is empty, Dr.Web Network Checker operating on the current host does not authenticate certificates of interacting agents; however, depending on the settings, these agents can verify authenticity of the certificate used by the agent operating on this host. Default value: *(not specified)*
`LoadBalanceServerSocket` *{address}*	Network socket (an IP address and a port) listened by Dr.Web Network Checker on the current host for receiving files sent by remote hosts for scanning (in case of operating as a network scanning server). Default value: *(not specified)*
`LoadBalanceAllowFrom` *{IP address}*	IP address of a remote network host from which the Dr.Web Network Checker operating on the current host can receive files for scanning (as a network scanning server). Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add host addresses 192.168.0.1 and 10.20.30.45 to the list. 1. Adding values to the configuration file.

Parameter	Description
	• Two values per line: `[NetCheck]` `LoadBalanceAllowFrom = "192.168.0.1", "10.20.30.45"` • Two lines (one value per line): `[NetCheck]` `LoadBalanceAllowFrom = 192.168.0.1` `LoadBalanceAllowFrom = 10.20.30.45` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset NetCheck.LoadBalanceAllowFrom -a 192.168.0.1` `# drweb-ctl cfset NetCheck.LoadBalanceAllowFrom -a 10.20.30.45` If the parameter is empty, remote files are not accepted for scanning (the host does not operate as a scanning server). Default value: *(not specified)*
`LoadBalanceSourceAddress` *{IP address}*	IP address of a network interface used by Dr.Web Network Checker on the current host to transfer files for remote scanning (if the host operates as a network scanning client and has several network interfaces). If an empty value is specified, the network interface is automatically selected by the OS kernel. Default value: *(not specified)*
`LoadBalanceTo` *{address}*	Socket (an IP address and a port) of a remote host to which Dr.Web Network Checker operating on the current host can send files for remote scanning (as a network scanning client). Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list. 1. Adding values to the configuration file. • Two values per line: `[NetCheck]` `LoadBalanceTo = "192.168.0.1:1234", "10.20.30.45:5678"`

Parameter	Description
	• Two lines (one value per line): `[NetCheck]` `LoadBalanceTo = 192.168.0.1:1234` `LoadBalanceTo = 10.20.30.45:5678` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset NetCheck.LoadBalanceTo -a 192.168.0.1:1234` `# drweb-ctl cfset NetCheck.LoadBalanceTo -a 10.20.30.45:5678` If the parameter value is empty, local files cannot be transferred for a remote scanning (the host does not operate as a network scanning client). Default value: *(not specified)*
`LoadBalanceStatusInterval` *{time interval}*	Time interval the current host waits to inform all distributed scanning agents specified in the `LoadBalanceAllowFrom` parameter about its workload. Default value: `1s`
`SpoolDir` *{path to directory}*	Local file system directory used to store files received from clients by Dr.Web Network Checker over the network for scanning. Default value: `/tmp/com.drweb.ncheck`
`LocalScanPreference` *{fractional number}*	Relative weight (priority) of the host upon selecting a server to scan a file (a local file or a file received over the network). If at some instant a relative weight of the local host is greater than the total weight of all hosts available as scanning servers, the file is kept by the agent for local scanning. Minimal value: `1`. Default value: `1`
`LoadBalanceSslCrl` *{path}*	Path to the directory or file with a list of revoked certificates. If a parameter value is not specified, Dr.Web Network Checker running on the current host does not verify certificates of interacting agents; however, depending on the settings, they may verify relevance of the certificate used by the agent running on the current host. Default value: *(not specified)*

9.7.4. Creating a Scanning Cluster

In this section

- Introductory remarks
- How to create a scanning cluster
- Configuring cluster hosts
- Verifying cluster operability

Introductory remarks

To create a scanning cluster that allows performing distributed scans (while scanning files or other objects), you need to have a set of network hosts with the Dr.Web Network Checker component installed on each host. To ensure that a cluster host not only receives and transmits data to be scanned, Dr.Web Scanning Engine must be installed on the host. Thus, to create a scanning cluster host, install at least the following components on the server (other components of Dr.Web Industrial for Unix File Servers that are installed automatically to ensure operability of the components listed here are not mentioned):

1. Dr.Web Network Checker (the `drweb-netcheck` package) is a component for receiving and sending data between hosts over the network.
2. Dr.Web Scanning Engine (the `drweb-se` package) is an engine for scanning data received over the network. The component may be absent; in this case, the host only transmits data to be scanned to other hosts of the scanning cluster.

The hosts that constitute the scanning cluster form a *peer to peer* network, i.e. each of the hosts, depending on the settings defined for the Dr.Web Network Checker component on this host, is capable of acting either as a *scanning client* (which transmits data for scanning to other hosts) or as a *scanning server* (which receives data for scanning from other hosts). With the appropriate settings, a cluster host can be both a scanning client and a scanning server at the same time.

Dr.Web Network Checker parameters related to scanning cluster configuration have names starting with `LoadBalance`.

How to create a scanning cluster

Let us consider the scanning cluster example displayed in the figure below.

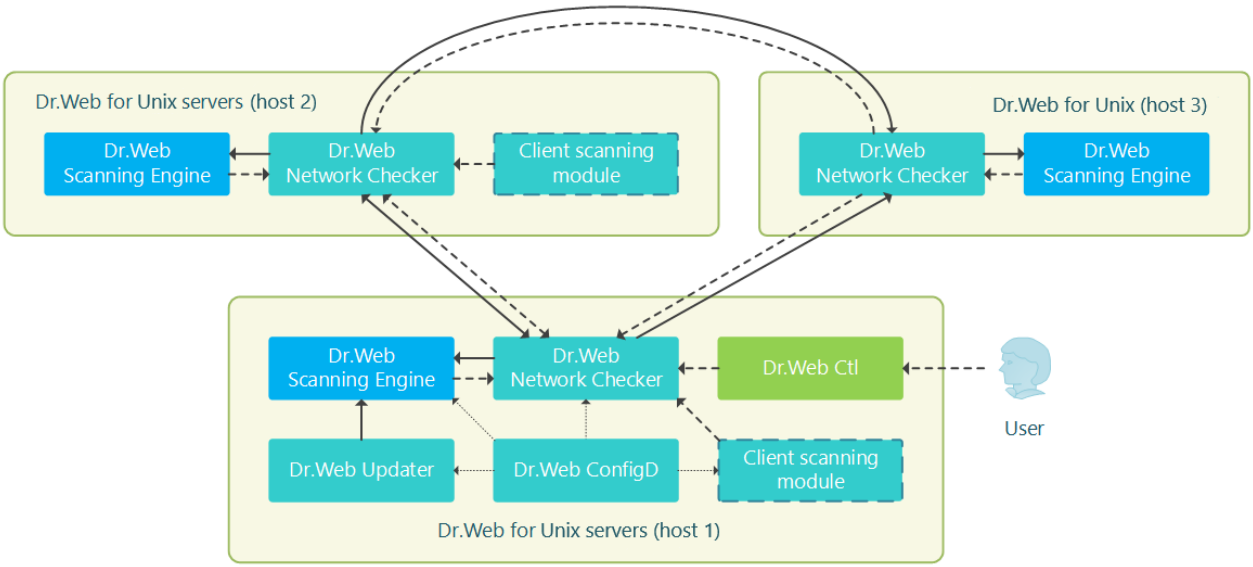


Figure 12. Scanning cluster structure

In this case, it is assumed that the cluster consists of three hosts (displayed in the figure as *host 1*, *host 2* and *host 3*). Host 1 and host 2 are servers with a fully-fledged Dr.Web product for Unix servers installed (for example, Dr.Web Gateway Security Suite or Dr.Web Mail Security Suite, the product type does not matter), and host 3 only assists in scanning files transferred from hosts 1 and 2. Therefore, only the minimal required component set is installed (Dr.Web Network Checker and Dr.Web Scanning Engine, other components that are automatically installed to ensure host operability, such as Dr.Web ConfigD, are not displayed in the figure). Hosts 1 and 2 can operate both as servers and scanning clients with regard to each other (perform mutual distribution of scanning load), and host 3—only as a server receiving tasks from hosts 1 and 2.

In the scheme, "Client scanning module" designates components of the servers that create scanning tasks for the Dr.Web Network Checker component. These tasks will be distributed (depending on the load balance) between local Dr.Web Scanning Engine and cluster partner hosts acting as scanning servers.

Only components that scan data not represented as a file in the local file system can act as the client scanning module. This means that the scanning cluster cannot be used for distributed scanning of files by the SpIDer Guard file system monitor and the Dr.Web File Checker component.

Configuring cluster hosts

To set up the specified cluster configuration, adjust Dr.Web Network Checker settings on all three cluster hosts. All of the following settings are provided in the `.ini` format (refer to configuration file format description).

Host 1

```
[NetCheck]
InternalOnly = No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 1 IP address>:<Host 1 port>
LoadBalanceAllowFrom = <Host 2 IP address>
LoadBalanceSourceAddress = <Host 1 IP address>
LoadBalanceTo = <Host 2 IP address>:<Host 2 port>
LoadBalanceTo = <Host 3 IP address>:<Host 3 port>
```

Host 2

```
[NetCheck]
InternalOnly = No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 2 IP address>:<Host 2 port>
LoadBalanceAllowFrom = <Host 1 IP address>
LoadBalanceSourceAddress = <Host 2 IP address>
LoadBalanceTo = <Host 1 IP address>:<Host 1 port>
LoadBalanceTo = <Host 3 IP address>:<Host 3 port>
```

Host 3

```
[NetCheck]
InternalOnly=No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 3 IP address>:<Host 3 port>
LoadBalanceAllowFrom = <Host 1 IP address>
LoadBalanceAllowFrom = <Host 2 IP address>
```

Notes:

- Other Dr.Web Network Checker parameters (not mentioned here) are left unchanged.
- Change IP addresses and port numbers to relevant ones.
- This example does not use SSL for data exchange between hosts. To use SSL, set the `LoadBalanceUseSsl` to `Yes`, as well as set relevant values for parameters `LoadBalanceSslCertificate`, `LoadBalanceSslKey` and `LoadBalanceSslCa`.

Verifying cluster operability

To verify cluster operability in data distribution mode, run the command on hosts 1 and 2:

```
$ drweb-ctl netscan <path to file or directory>
```

While this command is being run, Dr.Web Network Checker scans files from the specified directory having distributed load between cluster hosts. To view statistics on network scanning for each host, output statistics of Dr.Web Network Checker by running the command before scanning:

```
$ drweb-ctl stat -n
```

To interrupt outputting statistics, press CTRL+C or Q.

9.8. Dr.Web Scanning Engine

Dr.Web Scanning Engine is designed to search for viruses and other malicious objects in files and boot records (*MBR—Master Boot Record*, *VBR—Volume Boot Record*) of disk devices. The component loads Dr.Web Virus-Finding Engine into memory and starts it as well as loads Dr.Web virus databases used by the engine to detect threats.

The scanning engine operates in daemon mode, as a service that receives requests for scanning file system objects from other Dr.Web Industrial for Unix File Servers components (these are Dr.Web File Checker and Dr.Web Network Checker and, potentially, Dr.Web MeshD). If Dr.Web Scanning Engine and Dr.Web Virus-Finding Engine are absent or unavailable, no antivirus scanning is performed on this host (except when Dr.Web Industrial for Unix File Servers contains the Dr.Web MeshD component, whose settings define a connection to local cloud hosts providing scanning engine services).

9.8.1. Operating Principles

The Dr.Web Scanning Engine component operating in daemon mode receives requests from other Dr.Web Industrial for Unix File Servers components to scan file system objects (files and boot records) for embedded threats, queues scanning tasks and scans requested objects with Dr.Web Virus-Finding Engine. If a threat is detected in a scanned object that must be cured according to the scanning task, the scanning engine attempts to cure it if this action is applicable.

The scanning engine, Dr.Web Virus-Finding Engine and virus databases form a single entity and cannot be separated. The scanning engine downloads the virus databases and provides an operation environment for cross-platform Dr.Web Virus-Finding Engine. The virus databases and the scanning engine are updated by the Dr.Web Updater component included in Dr.Web Industrial for Unix File Servers but not being a part of the scanning engine. The update component is run by the Dr.Web ConfigD configuration management daemon periodically or forcibly in response to a user command. In addition, if Dr.Web Industrial for Unix File Servers operates in centralized protection mode, the virus databases and the scanning engine are updated by Dr.Web ES Agent, which interacts with a centralized protection server and receives updates from it.

Dr.Web Scanning Engine can be controlled by the Dr.Web ConfigD configuration management daemon or operate in standalone mode. In the former case, the daemon starts the engine and ensures that the virus databases are up to date. In the latter case, the engine is started and the virus databases are updated by an external application that uses the engine. Both the Dr.Web Industrial for Unix File Servers components that make requests to the scanning engine for scanning files and external applications use the same API.

You can create your own component (an external application) using Dr.Web Scanning Engine for scanning files. For this purpose, Dr.Web Scanning Engine provides a custom API based on the Google Protobuf technology. To obtain the Dr.Web Scanning Engine API guide and examples of client application code using Dr.Web Scanning Engine, contact the partner relations department of the Doctor Web company.

Received scanning tasks are automatically distributed in queues with different priorities (high, normal and low). Selection of a queue for a task depends on the component that created the task. For example, tasks received from file system monitors are placed in a high-priority queue, because response time is important while monitoring file system objects. The scanning engine collects statistics on its usage, including the number of all tasks received for scanning and queue lengths. As an average load rate, the scanning engine uses an average length of queues per second. This rate is averaged for the last minute, last 5 minutes and last 15 minutes.

Dr.Web Virus-Finding Engine supports a signature analysis (signature-based detection of threats covered by virus databases) and other methods of heuristic and behavioral analyses designed for detection of potentially dangerous objects based on machine instructions and other attributes of executable code.

Heuristic analyzer cannot guarantee highly reliable results and may cause the following errors:

- *Errors of the first type*. These errors occur when a safe object is detected as malicious (false positive detections).
- *Errors of the second type*. These errors occur when a malicious object is detected as safe.

Thus, objects detected by the heuristic analyzer are treated as *Suspicious*.

It is recommended that you quarantine suspicious objects. After virus databases are updated, such files can be scanned using the signature analysis. Keep the virus databases up to date in order to avoid errors of the second type.

Dr.Web Virus-Finding Engine allows scanning both unpacked and packed files and objects inside various containers, such as archives, email messages and so on.

9.8.2. Command-Line Arguments

To start Dr.Web Scanning Engine from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-se <socket> [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-se <socket> [<parameters>]
```

where the mandatory <*socket*> argument indicates an address of a socket used by Dr.Web Scanning Engine for processing requests of client components. It can be set only as a file path (a Unix socket).

Dr.Web Scanning Engine accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None
Startup options (identical to configuration file parameters and substitute them when necessary):	
`--CoreEnginePath`	Function: Path to the library of Dr.Web Virus-Finding Engine. Short form: None. Arguments: <*path to file*>—full path to the library in use.
`--VirusBaseDir`	Function: Path to a directory with virus database files. Short form: None. Arguments: <*path to directory*>—full path to the virus database directory.
`--TempDir`	Function: Path to a directory with temporary files. Short form: None. Arguments: <*path to directory*>—full path to the directory with temporary files.
`--KeyPath`	Function: Path to the key file in use. Short form: None. Arguments: <*path to file*>—full path to the key file.
`--MaxForks`	Function: Maximum allowed number of child processes that can be spawned by Dr.Web Scanning Engine during scanning. Short form: None. Arguments: <*number*>—maximum allowed number of child processes.
`--WatchdogInterval`	Description: Frequency with which Dr.Web Scanning Engine checks whether child processes are operable and ends those that stopped responding. Short form: None. Arguments: <*time interval*>—frequency of checking child processes.
`--AbortForkOnTimeout`	Function: Terminate scan processes by sending them SIGABRT on timeout.

	Short form: None. Arguments: None
`--ShellTrace`	Function: Shell tracing (log detailed information about file scanning performed by Dr.Web Virus-Finding Engine). Short form: None. Arguments: None
`--LogLevel`	Description: Logging level of Dr.Web Scanning Engine during its operation. Short form: None. Arguments: <*logging level*>. Allowed values: • `DEBUG`—the most detailed logging level. All messages and debug information are logged. • `INFO`—all messages are logged. • `NOTICE`—all error messages, warnings and notifications are logged. • `WARNING`—all error messages and warnings are logged. • `ERROR`—only error messages are logged.
`--Log`	Description: Logging method of the component. Short form: None. Arguments: <*log type*>. Allowed values: • `Stderr[:ShowTimestamp]`—output messages to the standard error stream *stderr*. The `ShowTimestamp` option is used to add a timestamp to every message. • `Syslog[:`<*facility*>`]`—pass messages to the `syslog` system logging service. The additional <*facility*> option is used to specify a type of a log to store messages logged by `syslog`. Allowed values: o `DAEMON`—messages from daemons; o `USER`—messages from user processes; o `MAIL`—messages from mailers; o `LOCAL0`—messages from local processes 0; ... o `LOCAL7`—messages from local processes 7. • <*path*>—path to a file for storing log messages. Examples: `--Log /var/opt/drweb.com/log/se.log` `--Log Stderr:ShowTimestamp` `--Log Syslog:DAEMON`
`--ForkMemoryLimit`	Description: Maximum amount of operating memory allocated for operation of a scanning engine instance. Short form: None. Arguments: <*size in bytes*>—maximum amount of operating memory.

	Allowed values: From 1073741824 bytes (1 Gb) to 4294967296 bytes (4 Gb). Default value: `4294967296` (4 Gb) ⚠ This parameter has no effect in 32-bit distributions of the product.
`--EngineDallocLimit`	Description: Maximum amount of operating memory allocated for storing temporary files. Short form: None. Arguments: <*size in bytes*>—maximum amount of operating memory. Allowed values: From 52428800 bytes (50 Mb) to 1073741824 bytes (1 Gb). Default value: `524288000` (500 Mb) ⚠ It is strongly recommended to change the default value of this parameter only if you are sure that this is necessary.

Example:

```
$ /opt/drweb.com/bin/drweb-se /tmp/drweb.ipc/.se --MaxForks=5
```

This command starts an instance of Dr.Web Scanning Engine and instructs it to create the `/tmp/drweb.ipc/.se` Unix socket to interact with client components and limit the number of child scanning processes to 5.

Startup notes

If necessary, any number of instances of Dr.Web Scanning Engine can be started, which provide the scanning service for client applications (not only for Dr.Web Industrial for Unix File Servers components). At that, if a value of the `FixedSocketPath` parameter is specified in the component settings, one instance of the scanning engine is always run by the Dr.Web ConfigD configuration management daemon and available to the clients via the Unix socket. Instances of the scanning engine started directly from the command line will operate in standalone mode without establishing connection to the configuration management daemon, even if it is running. To manage the operation of the component, as well as to scan files upon request, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory with Dr.Web Scanning Engine, use the `rawscan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl rawscan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-se` command.

9.8.3. Configuration Parameters

The component uses configuration parameters specified in the `[ScanEngine]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

This section stores the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-se` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-se`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. If the `FixedSocketPath` parameter is set, this setting is ignored (the component does not finish its operation after the time interval expires). Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `1h`

Parameter	Description
`FixedSocketPath` *{path to file}*	Path to the Unix socket of the fixed instance of Dr.Web Scanning Engine. If this parameter is specified, the Dr.Web ConfigD configuration management daemon ensures that there is always a running scanning engine instance available to clients via this socket. ⚠ Ensure that the following permissions are assigned to the directory with the socket file: `drwxr-xr-x` To view the permissions, run the command: `$ ls -ld <path to directory>` Default value: *(not specified)*
`MaxForks` *{integer}*	Maximum allowed number of child scanning processes that can run simultaneously spawned by Dr.Web Scanning Engine. Default value: *Automatically determined at startup* as twice the number of available CPU cores; or `4`, if the resulting number is less than 4.
`ForkMemoryLimit` *{integer}*	Maximum amount of operating memory allocated for operation of a scanning engine instance. The value is specified as an integer with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Allowed values: from 1 gigabyte (`1GB`) to 4 gigabytes (`4GB`). Default value: `4GB` ⚠ This parameter has no effect in 32-bit distributions of Dr.Web Industrial for Unix File Servers.
`EngineDallocLimit` *{integer}*	Maximum amount of operating memory allocated for storing temporary files. If the specified value is less than the size of a temporary file, then it will be created on the storage device, which will slow down the operation of Dr.Web Industrial for Unix File Servers. The value is specified as an integer with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Allowed values: from 50 megabytes (`50MB`) to 1 gigabyte (`1GB`). Default value: `500MB` ⚠ It is strongly recommended to change the default value of this parameter only if you are sure that this is necessary.

Parameter	Description
`WatchdogInterval` *{time interval}*	Rate at which Dr.Web Scanning Engine checks whether child scanning processes spawned by it are operable to detect deadlocks ("watchdog timer"). Default value: `15s`
`BufferedIo` *{boolean}*	Enable or disable buffered input/output while scanning files. Using buffered input/output on OSes of the GNU/Linux family and on FreeBSD can increase speed of scanning files on slow disks. Allowed values: • `On`, `Yes`, `True`—enable buffered input/output; • `Off`, `No`, `False`—disable buffered input/output. Default value: `Off`
`AbortForkOnTimeout` *{boolean}*	Terminate or do not terminate scan processes by sending them SIGABRT on timeout. Allowed values: • `On`, `Yes`, `True`—terminate scan processes on timeout; • `Off`, `No`, `False`—do not terminate scan processes on timeout. Default value: `Off`

9.9. Dr.Web Updater

The Dr.Web Updater component is designed for receiving all available updates for virus databases and Dr.Web Virus-Finding Engine from Doctor Web update servers.

If Dr.Web Industrial for Unix File Servers operates in centralized protection mode, updates are received from a centralized protection server; at that, all updates are received from the server via Dr.Web ES Agent, and Dr.Web Updater is not used for downloading updates.

9.9.1. Operating Principles

The component is designed to establish connections to Doctor Web update servers to check for updates for virus databases and Dr.Web Virus-Finding Engine. The lists of the servers that constitute an available update zone are stored in a special file signed to prevent its modification. Only basic and digest authentication are supported for connection to the update servers using a proxy server.

If Dr.Web Industrial for Unix File Servers is not connected to a centralized protection server or is connected to it in mobile mode, Dr.Web Updater is automatically started by the Dr.Web ConfigD configuration management daemon at intervals specified in the settings. The component can also be started by Dr.Web ConfigD upon receiving a corresponding command from the user (unscheduled update).

When updates become available on update servers, they are downloaded to the directory `/var/opt/drweb.com/cache/` for GNU/Linux or `/var/drweb.com/cache/` for FreeBSD; after that they are moved to the working directories of Dr.Web Industrial for Unix File Servers.

By default, all updates are downloaded from the update zone that is common for all Dr.Web products. The list of the servers used by default and included in the update zone is specified in the files located in directories defined by `*DrlDir` parameters. Upon a client request a custom update zone can be created (for each update type), the server list of which is specified in a separate file (named `update.drl` by default). This file is located in the directory specified by the corresponding `*CustomDrlDir` parameter. In this case, the update component will download updates only from these servers without using the servers from the default zone.

If you do not want to use the custom update zone, clear the value of the corresponding `*CustomDrlDir` parameter in the component settings.



The content of the files with server lists is signed so that the files cannot be modified. If you need to create a custom list of update servers, contact our technical support.

The component can back up the files to be updated to further roll back updates at user request. You can specify a backup location and an update history depth in the component settings. To

roll back updates, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line (run the tool using the `drweb-ctl` command).

9.9.2. Command-Line Arguments

To start the Dr.Web Updater component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-update [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-update [<parameters>]
```

Dr.Web Updater accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-update --help
```

This command outputs short help information about the Dr.Web Updater component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration daemon when necessary. To manage the operation parameters of the component, as well as to update virus databases and the scan engine, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-update` command.

9.9.3. Configuration Parameters

The component uses configuration parameters specified in the `[Update]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-update` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-update`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. The component is started upon the next update on schedule. When the update is completed, it is waiting for the specified time interval, and, if there are no new requests, it shuts down until the next update attempt. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the `SIGTERM` signal will not be sent if the component goes idle. Default value: `10m`

Parameter	Description
`Start` *{boolean}*	Enable or disable the component at the startup of Dr.Web Industrial for Unix File Servers. This parameter has priority over the `DwsUpdateEnabled` parameter. Allowed values: • `Yes`—enable the component at the startup of Dr.Web Industrial for Unix File Servers; • `No`—disable the component at the startup of Dr.Web Industrial for Unix File Servers. Default value: `Yes`
`UpdateInterval` *{time interval}*	The frequency to check for updates on Dr.Web update servers. This is a time period between a previous successful attempt to connect to the update servers (initiated automatically or manually) and the next attempt to perform an update. Default value: `30m`
`NetworkTimeout` *{time interval}*	A time-out period imposed on the network-related operations of the component while downloading updates from Dr.Web servers. This parameter is used when a connection is temporarily lost: if the connection is established again before the time-out expires, the interrupted updating process will be continued. Specifying the time-out value greater than `75s` has no effect as the connection is closed by the TCP timeout. Minimal value: `5s`. Default value: `60s`
`RetryInterval` *{time interval}*	Frequency of repeated attempts to perform an update using the update servers if the previous attempt failed. Allowed values: from 1 minute (`1m`) to 30 minutes (`30m`). Default value: `3m`
`MaxRetries` *{integer}*	Number of repeated attempts to perform an update using Dr.Web update servers (the frequency is specified in the `RetryInterval` parameter) if the previous attempt failed. If the value is set to `0`, repeated attempts are not made (the next update will be performed after the time period specified in the `UpdateInterval` parameter). Default value: `3`
`UseHttps` *{Always | ResListOnly | Never}*	Use or do not use HTTPS while downloading updates.

Parameter	Description
	Allowed values: • `Always`—always use HTTPS while downloading updates. • `ResListOnly`—use HTTPS only while downloading an `.lst` file containing a list of update files. At that, update files will be downloaded via HTTP. • `Never`—always use HTTP while downloading updates. Default value: `ResListOnly`
`Proxy` *{connection string}*	Parameters for connecting to a proxy server that is used by the Dr.Web Updater component when it is connecting to Dr.Web update servers (for example, if connecting directly to external servers is prohibited by network security policies). If a parameter value is not specified, the proxy server is not used. Allowed values: <*connection string*>—proxy server connection string. The string has the following format (URL): `[`<*protocol*>`://][`<*user*>`:`<*password*>`@]`<*host*>`:`<*port*> where: • <*protocol*> is a type of the protocol in use (in the current version, only `http` is available); • <*user*> is a user name for connecting to the proxy server; • <*password*> is a password for connecting to the proxy server; • <*host*> is the host address of the proxy server (an IP address or a domain name, i.e. FQDN); • <*port*> is a port to be used. The URL parts <*protocol*> and <*user*>*:*<*password*> may be absent. The proxy server address <*host*>*:*<*port*> is mandatory. If the user name or the password contains characters `@`, `%` or `:`, these characters must be replaced with the corresponding HEX codes: `%40`, `%25` and `%3A` respectively. Examples: 1. In the configuration file: • Connection to a proxy server hosted at `proxyhost.company.org` using port `123`: `Proxy = proxyhost.company.org:123` • Connection to a proxy server hosted at `10.26.127.0` using port `3336` via HTTP protocol as the `legaluser` user with the `passw` password: `Proxy = http://legaluser:passw@10.26.127.0:3336`

Parameter	Description
	• Connection to the proxy server hosted at `10.26.127.0` using port `3336` as the `user@company.com` user with the `passw%123` password: `Proxy = user%40company.com:passw%25123%3A@10.26.127.0:3336` 2. Setting the same values using the `drweb-ctl cfset` command: `# drweb-ctl cfset Update.Proxy proxyhost.company.org:123` `# drweb-ctl cfset Update.Proxy http://legaluser:passw@10.26.127.0:3336` `# drweb-ctl cfset Update.Proxy user%40company.com:passw%25123%3A@10.26.127.0:3336` Default value: *(not specified)*
`ExcludedFiles` *{file name}*	Name of a file that will not be updated by the Dr.Web Updater component. Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add the `123.vdb` and `456.dws` files to the list. 1. Adding values to the configuration file. • Two values per line: `[Update]` `ExcludedFiles = "123.vdb", "456.dws"` • Two lines (one value per line): `[Update]` `ExcludedFiles = 123.vdb` `ExcludedFiles = 456.dws` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset Update.ExcludedFiles -a 123.vdb` `# drweb-ctl cfset Update.ExcludedFiles -a 456.dws` Default value: `drweb32.lst`
`BaseUpdateEnabled` *{boolean}*	Allow or do not allow updating the virus databases and the scan engine. Allowed values: • `Yes`—updating is allowed and will be performed; • `No`—updating is not allowed and will not be performed. Default value: `Yes`
`BaseDrlDir` *{path to directory}*	Path to a directory that contains files used for connection to servers of a standard update zone, which are used to update

Parameter	Description
	virus databases and the scan engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/drl/bases` • for FreeBSD: `/var/drweb.com/drl/bases`
`BaseCustomDrlDir` *{path to directory}*	Path to a directory that contains files used for connection to a special ("customized") update zone, which are used to update virus databases and the scan engine. If the directory defined in this parameter contains a non-empty signed server list file (the `.drl` file), the update is performed only from these servers, and the main zone servers (see above) are not used to update the virus databases and the scanning engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/custom-drl/bases` • for FreeBSD: `/var/drweb.com/custom-drl/bases`
`VersionUpdateEnabled` *{boolean}*	Allow or do not allow updating versions of Dr.Web Industrial for Unix File Servers components. Allowed values: • `Yes`—updating is allowed and will be performed; • `No`—updating is not allowed and will not be performed. Default value: `Yes`
`VersionDrlDir` *{path to directory}*	Path to a directory that contains files for connection to servers that are used for updating versions of Dr.Web Industrial for Unix File Servers components. Default value: • for GNU/Linux: `/var/opt/drweb.com/drl/version` • for FreeBSD: `/var/drweb.com/drl/version`
`BackupDir` *{path to directory}*	Path to a directory where old versions of updated files are saved for possible rollback. Only updated files are backed up upon every update. Default value: • for GNU/Linux: `/var/opt/drweb.com/backup` • for FreeBSD: `/var/drweb.com/backup`
`MaxBackups` *{integer}*	The maximum number of the previous versions of updated files, which are saved. If this number is exceeded, the oldest copy is removed upon the next update.

Parameter	Description
	If the parameter value is `0`, the previous versions of backup files are not stored. Default value: `0`

9.10. Dr.Web ES Agent

The centralized protection agent Dr.Web ES Agent is designed for connecting Dr.Web Industrial for Unix File Servers to a centralized protection server.

When Dr.Web Industrial for Unix File Servers is connected to the centralized protection server, Dr.Web ES Agent synchronizes the license key file with the key files stored on the centralized protection server. In addition, Dr.Web ES Agent sends statistics on virus events, the list of running components and their status to the centralized protection server.

Furthermore, Dr.Web ES Agent updates Dr.Web Industrial for Unix File Servers virus databases directly from the connected centralized protection server without using the Dr.Web Updater component.

9.10.1. Operating Principles

The Dr.Web ES Agent component connects to a centralized protection server, which allows a network administrator to implement a common security policy within the entire network, in particular, to configure the same scanning settings and reaction on threat detection for all workstations and servers of the network. In addition, the centralized protection server also acts as an internal update server on the protected network and stores up-to-date virus databases (in this case, updating is performed via Dr.Web ES Agent, Dr.Web Updater is not used).

When Dr.Web ES Agent connects to the centralized protection server, the agent receives up-to-date settings of the program components and the license key file from the server, which are then passed to the Dr.Web ConfigD configuration management daemon for applying them to the managed components. In addition, the component can also receive tasks from the centralized protection server for scanning file system objects on the station (including on schedule).

Dr.Web ES Agent collects statistics on detected threats and applied actions and sends it to the server to which the agent is connected.

To connect Dr.Web ES Agent to the centralized protection server, the password and identifier of the host (“station” in terms of the centralized protection server) are required as well as a public encryption key file, which is used by the server for authentication. Instead of the station identifier, while connecting, you can specify the identifier of the main and tariff groups in which the station is to be included. For the required identifiers and public key file, contact the administrator of your anti-virus network.

In addition, you can connect a protected host (“station”) to the centralized protection server as a “newbie”, if this is allowed by the centralized protection server. In this case, after the administrator confirms the request to connect the station, the centralized protection server automatically generates new identifier and password for the host and sends them to the agent for future connections.

9.10.2. Command-Line Arguments

To start the Dr.Web ES Agent component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-esagent [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-esagent [<parameters>]
```

Dr.Web ES Agent accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-esagent --help
```

This command outputs short help information about the Dr.Web ES Agent component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration daemon at the startup of the operating system. To manage the operation parameters of the component as well as to connect Dr.Web Industrial for Unix File Servers to a centralized protection server, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-esagent` command.

9.10.3. Configuration Parameters

The component uses configuration parameters specified in the `[ESAgent]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-esagent` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-esagent`
`MobileMode` *{On | Off | Auto}*	Enable or disable the mobile mode of Dr.Web Industrial for Unix File Servers while connected to a centralized protection server. Allowed values: • `On`—enable the mobile mode if it is allowed by the centralized protection server (install updates from the update servers of Doctor Web using Dr.Web Updater); • `Off`—disable the mobile mode and continue operation in centralized protection mode (always receive updates from the centralized protection server); • `Auto`—enable the mobile mode if allowed by the centralized protection server and install updates both from the update servers of Doctor Web using Dr.Web Updater and from the centralized protection server depending on which connection is available and which connection quality is higher. Behavior of this parameter depends on server permissions: if the mobile mode is not allowed on the server in use, this parameter has no effect. Default value: `Auto`
`Discovery` *{On | Off}*	Allow or do not allow the agent to receive *discovery* requests from a network inspector built in the centralized protection server (*discovery* requests are

Parameter	Description
	used by the inspector to check the structure and state of the anti-virus network). Allowed values: • `On`—allow the agent to receive and process *discovery* requests; • `Off`—do not allow the agent to receive and process *discovery* requests.  This parameter has priority over the settings of the centralized protection server: if the parameter value is set to `Off`, the agent does not receive discovery requests even if this feature is enabled on the server. Default value: `On`
`UpdatePlatform` *{platform name}*	Designation of a platform for which the agent will receive updates for the scanning engine from the centralized protection server. Allowed values: • for GNU/Linux: `unix-linux-32`, `unix-linux-64-engine64`, `unix-linux-aarch64`, `unix-linux-e2k`, `unix-linux-e2k-engine64`, `unix-linux-mips`, `unix-linux-ppc64le`; • for FreeBSD: `unix-freebsd-32`, `unix-freebsd-64-engine64`; • for Darwin: `unix-darwin-32`, `unix-darwin-64-engine64`, `unix-darwin-aarch64`.  It is strongly recommended to change the parameter value only if you are sure that this is necessary. Default value: *Depends on the platform in use*
`DebugIpc` *{boolean}*	Log or do not log IPC messages at the debug level (with `LogLevel = DEBUG`) (interaction of Dr.Web ES Agent with the Dr.Web ConfigD configuration management daemon). Default value: `No`
`DebugConfig` *{boolean}*	Log or do not log information about settings received from the centralized protection server. Default value: `No`
`SrvMsgAutoremove` *{integer}*	Storage period after which the messages from the centralized protection server are removed automatically. Allowed values: from 1 week (`1w`) to 365 days (`365d`). The storage period is specified as an integer with a suffix (`s`, `m`, `h`, `d`, `w`). Default value: `1w`

9.11. Dr.Web HTTPD

The Dr.Web HTTPD component provides infrastructure for local and remote interaction with Dr.Web Industrial for Unix File Servers via HTTP (for example, using a web browser).

In addition to managing Dr.Web Industrial for Unix File Servers via the Dr.Web web interface, you can also use a command-line interface (HTTP API) of Dr.Web HTTPD directly to interact with Dr.Web Industrial for Unix File Servers components via HTTPS. This capability allows you to create your own interface to manage Dr.Web Industrial for Unix File Servers.

The HTTP API of Dr.Web HTTPD is described in the corresponding section.

To use a secure HTTPS connection, it is required to provide a valid SSL server certificate and private key for Dr.Web HTTPD. By default, a server certificate and an SSL private key are generated for Dr.Web HTTPD automatically during the installation procedure, but, if necessary, you can generate your own certificate and key. In addition, a user personal authorization certificate signed by a certificate issued by a certificate authority and trusted by Dr.Web HTTPD can be used for automatic client authorization when connecting to Dr.Web HTTPD.

To generate keys and certificates, you can use the `openssl` utility. Examples of using the `openssl` utility to generate certificates and private keys are provided in the Appendix E. Generating SSL Certificates section.

9.11.1. Operating Principles

Dr.Web HTTPD is a web server designed for managing Dr.Web Industrial for Unix File Servers without a need for either third-party servers (for example, Apache HTTP Server and Nginx) or management services, such as Webmin. Furthermore, the component can function simultaneously with them on the same host without impeding their operation.

The Dr.Web HTTPD server processes requests received via HTTP and HTTPS to sockets specified in the settings. For this reason, the server does not have conflicts with other web servers used on the same host. Dr.Web Industrial for Unix File Servers is managed via the secure protocol, HTTPS.



The Dr.Web management web interface is not required for the operation of Dr.Web Industrial for Unix File Servers and can be absent; therefore, the corresponding block is circled with a dashed line in the scheme.

The Dr.Web HTTPD component issues commands to the Dr.Web ConfigD configuration management daemon, to the Dr.Web File Checker component designed for file scanning and to other components on the basis of commands received via the HTTP API.

The management web interface, which uses Dr.Web HTTPD, is described in the corresponding section.

If the Dr.Web management web interface is not included in Dr.Web Industrial for Unix File Servers, you can connect the product to any third-party management interface that uses the HTTP API (described in the Description of the HTTP API section) of the Dr.Web HTTPD component.

9.11.2. Command-Line Arguments

To start the Dr.Web HTTPD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-httpd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-httpd [<parameters>]
```

Dr.Web HTTPD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-httpd --help
```

This command outputs short help information about the Dr.Web HTTPD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). If the component is running, to manage the components of Dr.Web Industrial for Unix File Servers, you can use any standard web browser to access, via HTTPS, any of the addresses at which the web interface is served. To manage the operation of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-httpd` command.

9.11.3. Configuration Parameters

The component uses configuration parameters specified in the `[HTTPD]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-httpd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-httpd`
`Start` *{boolean}*	Start or stop the component with the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration management daemon to start the component immediately, and setting this parameter to `No`—to shut down the component immediately. Default value: *Depends on whether the management web interface is installed*
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`

Parameter	Description
`AdminListen` *{address, ...}*	List of network sockets (every network socket is defined by the *<IP address>:<port>* pair) listened by Dr.Web HTTPD for HTTPS connections from clients with administrative privileges. These sockets are used both to connect to the management web interface and to access the HTTP API. The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list). Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list. 1. Adding the values to the configuration file. • Two values per line: `[HTTPD]` `AdminListen = "192.168.0.1:1234", "10.20.30.45:5678"` • Two lines (one value per line): `[HTTPD]` `AdminListen = 192.168.0.1:1234` `AdminListen = 10.20.30.45:5678` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset HTTPD.AdminListen -a 192.168.0.1:1234` `# drweb-ctl cfset HTTPD.AdminListen -a 10.20.30.45:5678` If no value is specified, it is impossible to use HTTP API and the management web interface. Default value: `127.0.0.1:4443`
`AdminSslCertificate` *{path to file}*	Path to the server certificate file used by the management web interface server to communicate with clients that establish HTTPS connections to the administrative socket. This file is generated automatically during the installation of the component. ⚠ The certificate file and the private key file (specified by the `AdminSslKey` parameter) must match each other. Default value: • for GNU/Linux: `/etc/opt/drweb.com/certs/serv.crt` • for FreeBSD: `/usr/local/etc/drweb.com/certs/serv.crt`
`AdminSslKey` *{path to file}*	Path to the private key file used by the management web interface server to communicate with clients that establish HTTPS connections to the administrative socket. The private key file is generated automatically during the installation of the component.

Parameter	Description
	 The certificate file (specified by the `AdminSslCertificate` parameter) and the private key file must match each other. Default value: • for GNU/Linux: `/etc/opt/drweb.com/certs/serv.key` • for FreeBSD: `/usr/local/etc/drweb.com/certs/serv.key`
`AdminSslCA` *{path to file}*	Path to a Certification Authority (CA) certificate file for certificates used by clients that establish HTTPS connections to the administrative socket. If a client certificate is signed with the certificate specified by this parameter, client authentication by providing the login/password pair is not performed. Furthermore, this authentication method is prohibited for clients that use client certificates signed with this certificate. A client who passed the certificate-based authentication is always assumed to be a superuser (*root*). Default value: *(not specified)*
`PublicListen` *{address, ...}*	List of network sockets (every network socket is defined by the *<IP address>:<port>* pair) listened by Dr.Web HTTPD for HTTP connections from clients with limited privileges. The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list). Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list. 1. Adding the values to the configuration file. • Two values per line: `[HTTPD]` `PublicListen = "192.168.0.1:1234", "10.20.30.45:5678"` • Two lines (one value per line): `[HTTPD]` `PublicListen = 192.168.0.1:1234` `PublicListen = 10.20.30.45:5678` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset HTTPD.PublicListen -a 192.168.0.1:1234` `# drweb-ctl cfset HTTPD.PublicListen -a 10.20.30.45:5678` You cannot access the full scope of the HTTP API commands or access the management web interface at these addresses. Default value: *(not specified)*

Parameter	Description
`WebconsoleRoot` *{path to directory}*	Path to a directory with the files used by the management web interface (similar to the `htdocs` directory of Apache HTTP Server). Default value: • for GNU/Linux: `/opt/drweb.com/share/drweb-httpd/webconsole` • for FreeBSD: `/usr/local/libexec/drweb.com/share/drweb-httpd/webconsole`
`HelpRoot` *{path to directory}*	Path to a directory with the files of the documentation accessible via the management web interface. Default value: • for GNU/Linux: `/opt/drweb.com/share/doc/html` • for FreeBSD: `/usr/local/libexec/drweb.com/share/doc/html`
`AccessLogPath` *{path to file}*	Path to the file in which all HTTP/HTTPS requests from clients to the management web interface server are logged. If the parameter is not specified, requests are not logged. Default value: *(not specified)*

9.11.4. Description of the HTTP API

In this section

- General information
  - Format of an HTTP response to an HTTP request
  - Encoding of strings in JSON objects
  - General restrictions on transmitting data
- User authentication and authorization
  - By specifying a login and a password (SCS)
  - Authorization using a personal certificate
- Managing Dr.Web Industrial for Unix File Servers
- Scanning objects
- Managing the list of threats
- Managing quarantine
- HTTP API usage examples

General information

The HTTP API is provided as a means to integrate Dr.Web Industrial for Unix File Servers with third-party applications via HTTP (to ensure security, HTTPS is used).

HTTP 1.0 is used to interact with the API. All requests use standard methods of HTTP: `GET` and `POST`. All data is transferred in the form of JSON objects, except as otherwise specified. If you are sending a JSON object in the body of your HTTP POST request, set the `Content-Type:` header to have the value of `application/json`. For clarity, the examples of JSON objects provided below have comments (starting with `//` characters). The comments are not supported by the JSON format and must be deleted upon pasting into code.

Format of an HTTP response to an HTTP request

- In response to all requests, except as otherwise specified, the API always returns a JSON object. If an error occurs, the API returns an Error JSON object.
- If the JSON object sent as a response has a field of an Array type, but this array does not contain any elements, this field is omitted in the server response.
- In all responses, except as otherwise specified, the `Content-Type:` header field has the `application/json` value.
- If the client requests an endpoint that does not exist, the Error JSON object in which the `code` field with the `EC_UNEXPECTED_MESSAGE` value is returned.
- If SCS (*Secure Cookie Sessions for HTTP*) is used, the responses contain a cookie proving the client authorization (hereinafter referred to as the *SCS cookie*).

Encoding of strings in JSON objects

- The strings are passed in the UTF-8 encoding (without BOM). Characters that are not part of the ASCII table are not escaped with sequences of the form `\uXXXX`, but are passed in the UTF-8 encoding.
- JSON objects can contain both UTF-8 encoded characters and escaped sequences of the form `\uXXXX`.

General restrictions on transmitting data

- In POST requests with JSON objects in their body any characters complying with RFC 7159 are allowed.
- In GET requests any characters complying with RFC 1945 are allowed in the URI.
- Characters complying with RFC 1945 can be used in any other part of the request (either in the headers or in the body).

User authentication and authorization

To start using the API, you must be authenticated by the server. Two methods of authorization are provided:

1. Using SCS in accordance with RFC 6896.

2. Using client SSL certificates attested by a trusted CA's certificate on behalf of Dr.Web HTTPD. In this case the client is considered authorized as a superuser (X.509 client certificates are used).

If SCS is used, SCS cookies are passed in the following HTTP headers: `Cookie:`—in the request and `Set-Cookie:`—in the response.
When authorizing with SSL certificates, SCS cookies are not required.

When authorizing with SCS, using the API starts with sending the `login` command. If this command is run successfully, the client receives an SCS cookie confirming the authorization. When authorizing with a client certificate, you do not need to run the `login` command. If you try to run it, the Error JSON object is returned.

By specifying a login and a password (SCS)

User authentication and authorization commands:

API command	Description
`login`	Action: Authorize the client based on the specified user name and password to further use the HTTP API. If the authentication is successful, an SCS cookie will be returned. URI: `/api/10.2/login` HTTP method: POST Input parameters: AuthOptions object Result of successful execution: an empty object, SCS cookie
`logout`	Action: Revoke (deactivate) the SCS cookie provided earlier. The Error object with the `EC_NOT_AUTHORIZED` error code is returned in response to a request with the revoked SCS cookie. URI: `/api/10.2/logout` HTTP method: GET Input parameters: SCS cookie Result of successful execution: an empty object
`whoami`	Action: Get the name of the authorized user. URI: `/api/10.2/whoami` HTTP method: GET Input parameters: (SCS cookie)* Result of successful execution: whoami object, (SCS cookie)

*) Here and below the SCS cookie is put in parentheses, because it is required only when authorizing via SCS.



The `login` and `logout` commands are used only when authorizing via SCS.

Description of used objects

1) `AuthOptions`—object containing the user login and password:

```
{
 "user": <string>, //User name
 "password": <string> //User password
}
```



You can specify any user who is a member of the admin group (`sudoers` in Debian and Ubuntu, `wheel` in CentOS and Fedora, `astra-admin` in Astra Linux, etc). If the user is not a member of the `sudoers` group, the `EC_NOT_AUTHORIZED` error is returned in the response.

2) `whoami`—object containing the name of the user who was authorized to use the HTTP API:

```
{
 "whoami":
 {
 "user": <string>, //User name
 }
}
```

3) `Error`—object containing information about the occurred error:

```
{
 "error":
 {
 "code": <string>, //String specifying an error code of the form EC_XXX
 *"what": <string> //Error description
 }
}
```

The parameter marked with `*` is optional.



The Error JSON object, that is returned in response to HTTP API commands if an error occurs while processing the requests, has the `code` field containing an internal string-type code used by the components of Dr.Web Industrial for Unix File Servers rather than a numeric error code. This code is a string in the form of `EC_XXX`. To find out the numeric code and get detailed information about the error, refer to the Appendix F. Known Errors section.

Authorization using a personal certificate

This authorization method implies using a personal certificate signed by a certificate authority certificate specified as trusted in the Dr.Web HTTPD settings. If you have authorized with a certicate, all your requests are considered executed as a superuser (the *root* user).

To authorize with a personal user certificate

1. Create a personal certificate signed by a certificate authority certificate.
2. In the Dr.Web HTTPD settings (the `AdminSslCA` parameter), specify a path to the certificate authority certificate used to sign your personal certificate.
3. Every time you connect to Dr.Web HTTPD, use a signed certificate.

If necessary, refer to the Appendix E. Generating SSL Certificates section.

Managing Dr.Web Industrial for Unix File Servers

API commands for viewing and modifying the current values of configuration parameters:

API command	Description
Configuration management commands	
`get_lexmap`	Action: Get the parameter values of the current configuration (a “lexical map” of parameters). URI: `/api/10.2/get_lexmap` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: LexMaps object, (SCS cookie)
`set_lexmap`	Action: Set or reset the specified parameter values of the current configuration (sent as a “lexical map” of parameters). URI: `/api/10.2/set_lexmap` HTTP method: POST Input parameters: (SCS cookie), LexMap object Result of successful execution: SetOptionResult object, (SCS cookie)
Updating commands	
`start_update`	Action: Start updating.

API command	Description
	URI: `/api/10.2/start_update` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: the StartUpdate object, (SCS cookie)
`stop_update`	Action: Stop an active update process. URI: `/api/10.2/stop_update` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: an empty object, (SCS cookie)
`baseinfo`	Action: View information about the loaded virus bases URI: `/api/10.2/baseinfo` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: the BaseInfoResult object containing the VirusBaseInfo object (SCS cookie)
Licence management commands	
`install_license`	Action: Install the specified key file. URI: `/api/10.2/install_license` HTTP method: POST Input parameters: (SCS cookie), body of the key file (or of an archive with the key file) Result of successful execution: an empty object, (SCS cookie)
Commands for connection to the centralized protection server	
`esconnect`	Action: Enable the centralized protection mode. URI: `/api/10.2/esconnect` HTTP method: POST Input parameters: (SCS cookie), the ESConnection object

API command	Description
	Result of successful execution: an empty object, (SCS cookie)
`esdisconnect`	Action: Disable the centralized protection mode. URI: `/api/10.2/esdisconnect` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: an empty object, (SCS cookie)

Configuration of the product components is returned and set as the so-called *lexical map*, i.e. as a sequence of parameter-value pairs. A LexMaps object always contains three nested LexMap objects containing the following lexical maps:

- *active*—active (valid) parameter values;
- *hardcoded*—default values (automatically assigned to the parameters which values are not specified or invalid);
- *master*—map of configuration parameter values specified by the client.



The `get_lexmap` command always returns all three sets of configuration parameter values for all the components that can be included in Dr.Web Industrial for Unix File Servers, and not only for those that are installed and running.

Description of used JSON objects

1) `LexMaps`—object containing an active, a predefined and a user-defined lexical maps of parameter values:

```
{
 "active": <LexMap>, //Active values of configuration parameters
 "hardcoded": <LexMap>, //Predefined values of configuration parameters
 "master": <LexMap> //Parameter values set by the user
}
```

   Each of these fields is a LexMap object, which in turn contains an array of LexOption objects.

2) `LexMap`—object containing a lexical map of parameters:

```
{
 "option": [<LexOption>] //Array of configuration options
}
```

3) `LexOption`—object containing a parameter or a configuration section (a group of parameters):

```
{
 "key": <string>, //Option name (a parameter name or a section name)
 *"value": <LexValue>, //If this option is a parameter
```

```
 *"map": <LexMap> //If this option is a section
}
```

The parameters marked with * are optional.

The LexOption object represents a section or a parameter of the Dr.Web Industrial for Unix File Servers configuration. It always has a `key` field corresponding to the name of the section or of the parameter, as well as a `value` field (if the object is a parameter), or a `map` field (in case it describes a section). A section is also an object of the LexMap type; whereas the parameter value is an object of the LexValue type containing an `item` field specifying the parameter value in the string format.

4) `LexValue`—object containing an array of parameter values:

```
{
 "item": [<string>] //Array of parameter values
}
```

The `set_lexmap` command accepts a LexMap object that must contain all the parameters which values you want to change to new ones or reset to their defaults. Those parameters that you want to reset to their default values must not contain the `value` field. Parameters that are not mentioned in the lexical map passed to the `set_lexmap` command will not be changed. As a result of its execution, the `set_lexmap` command returns a SetOptionResult object containing changed values of every parameter passed to the command.

5) `SetOptionResult`—object with an `item` field containing an array of modified parameter values:

```
{
 "item": [<SetOptionResultItem>] //Array of results
}
```

The object contains an array of SetOptionResultItem objects containing changed values of every parameter passed to the command.

6) `SetOptionResultItem`—object containing information about a change of a parameter value:

```
{
 "option": <string>, //Parameter name
 "result": <string>, //Result of changing the value (an error code)
 *"lower_limit": <string>, //Lowest allowed value
 *"upper_limit": <string> //Highest allowed value
}
```

The parameters marked with * are optional.

The `option` field contains the name of the parameter to which your action was applied, and the `result` field contains the result of an attempt to change the value of this parameter. If a new value was successfully assigned, the field has the `EC_OK` value. In case of an error, this object may contain a `lower_limit` field and an `upper_limit` field that store the maximum and the minimum allowed value for this parameter.

7) The `StartUpdate` object contains data on the started update process:

```
{
 "start_update":
  {
    "attempt_id": <number> //Identifier of the started update process
```

```
  }
}
```

8) The `ESConnection` object contains data on connection to a centralized protection server:

```
{
 *"server": <string>, //<Host address>:<port>(without the http/https prefix)
 "certificate": <string>, //Base64 server key
 *"newbie": <boolean>, //Default value: false
 *"login": <string>, //User name
 *"password": <string> //Password
}
```

The parameters marked with `*` are optional.

The `login` and `password` parameters are only specified if `newbie = true`.
Before connecting, download the certificate file from the centralized protection server and run the command:

```
$ cat certificate.pem | base64
```

The string obtained from running this command is used as the `certificate` parameter value.

9) The `BaseInfoResult` object contains data on the loaded virus bases:

```
{
 "vdb_base_stamp": <number>, //Base timestamp
 "vdb_bases": [<VirusBaseInfo>] //Detailed information about the base
}
```

10) The `VirusBaseInfo` object contains the detailed information about the virus base:

```
{
  "path": <string>, //Path to the base file
  "virus_records": <number>, //Number of records in the base
  "version": <number>, //Base version
  "timestamp": <number>, //Base timestamp
  "md5": <string>, //Base MD5 hash
  "load_result": <string>, //Result of loading the base (EC_OK if the base has been
loaded successfully)
  *"sha1": <string> //Base SHA1 hash
}
```

The parameter marked with `*` is optional.

Scanning objects

API commands for scanning objects:

API command	Description
Data scanning (calling the Dr.Web Network Checker component)	
`scan_request`	Action: Request to create a connection (*endpoint*) to scan data with the necessary parameters. URI: `/api/10.2/scan_request`

API command	Description
	HTTP method: POST Input parameters: (SCS cookie), the ScanOptions object Result of successful execution: the ScanEndpoint object, (SCS cookie)
`scan_endpoint`	Action: Start data scanning (for instance, file body) at the created *endpoint* connection. URI: `/api/10.2/scan_endpoint/<`*endpoint*`>` HTTP method: POST Input parameters: (SCS cookie), verifiable data Result of successful execution: the ScanResult object, (SCS cookie)
`scan_path`	Action: Scan a file or a directory at the specified path. URI: `/api/10.2/scan_path` HTTP method: POST Input parameters: (SCS cookie), ScanPathOptions object Result of successful execution: the ScanPathResult object, (SCS cookie)
`scan_stat`	Action: Get scan statistics. URI: `/api/10.2/scan_stat` HTTP method: GET Input parameters: (SCS cookie), statistics format (JSON or CSV) Result of successful execution: the ScanStat object (if the JSON format is selected), (SCS cookie) If the CSV format is selected, a table whose columns correspond to the fields of the ScanStat object is returned.

Description of used JSON objects

1) `ScanOptions` is an object containing parameters used for creating *endpoint* for file scanning:

```
{
 "scan_timeout_ms": <number>, //Timeout for scanning one file, in ms
 "cure": <boolean>, //Cure or do not cure an infected file
 "heuristic_analysis": <boolean>, //Enable or disable the heuristic analysis
 "packer_max_level": <number>, //Maximum nesting level for packed objects
 "archive_max_level": <number>, //Maximum nesting level for archives
```

```
 "mail_max_level": <number>, //Maximum nesting level for email objects
 "container_max_level": <number>, //Maximum nesting level for other compound objects
(containters)
 "max_compression_ratio": <number>, //Maximum archive compression ratio
 "min_size_to_scan": <number>, //Minimal size of an object to be scanned
 "max_size_to_scan": <number>, //Maximal size of an object to be scanned
 "threat_hash": <boolean> //Get hashes of detected threats
}
```

2) `ScanPathOptions` is the object containing parameters for scanning a file or a directory at the specified path:

```
{
 "path": <string>, //Absolute path to a file or a directory to be scanned
 *"exclude_path": [<string>], //List of paths excluded from scanning (masks are
allowed)
 *"scan_timeout_ms": <number>, //Timeout for scanning one object
 *"archive_max_level": <number>, //Maximum nesting level for archived objects
 *"packer_max_level": <number>, //Maximum nesting level for packed objects
 *"mail_max_level": <number>, //Maximum nesting level for email messages
 *"container_max_level": <number>, //Maximum nesting level for other compound objects
(containters)
 *"max_compression_ratio": <number>, //Maximum archive compression ratio
 *"heuristic_analysis": <boolean>, //Enable or disable the heuristic analysis (default
value: true)
 *"follow_symlinks": <boolean>, //Follow or do not follow symbolic links
 *"min_size_to_scan": <number>, //Minimal size of an object to be scanned
 *"max_size_to_scan": <number>, //Maximal size of an object to be scanned
 *"timeout_ms": <number>, //Timeout for scanning all objects
 *"threat_hash": <boolean> //Return or do not return SHA1 and SHA256 hashes of threats
}
```

The parameters marked with `*` are optional.

3) `ScanPathResult` is an object containing results of scanning a file or a directory at the specified path:

```
{
  //ScanPathResult:
  "results": [<ScanResult>], //Scan results
  *"error": <string> //Error if the process has finished with an error (for example, by
timeout)
}
```

The parameter marked with `*` is optional.

If the scanning is successful, the `error` string will be absent in the response.

4) `ScanResult` is an object containing scan results:

```
{
 //ScanResult:
 "scan_report": <ScanReport>, //Scan report
 *"sha1": <string>, //SHA1 hash of the threat
 *"sha256": <string> //SHA256 hash of the threat
}
```

The parameters marked with `*` are optional.

5) `ScanReport` is an object containing information about the file with a threat:

```
{
 //ScanReport:
 "object": <string>, //Name of the scanned object
```

```
     // Absolute path for a file, the file name for a nested object
     // Always refers to a temporary file when calling scan_endpoint
 *"size": <number>, //Object size
 *"compressed_size": <number>, //Compressed object size
 *"core_fingerprint": <string>, //Engine fingerprint
 *"packer": [<string>], //List of packers used to pack the object
 *"compression_ratio": <number>, //Archive compression ratio
 *"archive": <Archive>, //Information about the archive (container) type, if the
object was identified as such
 *"virus": [<Virus>], //Threats detected in the object (if found)
 *"item": [<ScanReport>], //Reports on scanning nested objects (if any)
 *"error": <string>, //Scanning error (if occurred)
 *"heuristic_analysis": <boolean>, //Whether the heuristic analysis has been enabled
(true—enabled, false—disabled)
 *"cured": <boolean>, //Whether the object has been cured (true—cured, false—not
cured)
 *"cured_by_deletion": <boolean>, //Whether the object has been deleted while curing
(true—deleted, false—not deleted)
 *"new_path": <string>, //New path to the object renamed while curing
 *"user_time": <number>, //Time spent in the userspace
 *"system_time": <number> //Time spent on system calls while scanning
}
```

The parameters marked with `*` are optional.

The fields `virus` and `error` may be absent in the response if no threats were detected and no errors occurred during the scan. To call `scan_endpoint`, the `scan_endpoint` field always specifies a temporary file created by the Dr.Web Network Checker component in the server local file system and containing the data for scanning sent in the body of the `scan_endpoint` request).

6) `ScanEndpoint` is an object containing information about *endpoint* created for file scanning:

```
{
 "endpoint": <string> //Unique identifier of the created endpoint
}
```

The `endpoint` identifier returned in the object body. The identifier is used to start file scanning with the `scan_endpoint` command (as part of URI).

7) `VirusInfo` is an object containing information about the detected threat:

```
{
 "type": <string>, //Detected threat type
 "name": <string> //Name of the threat
}
```

The `type` field (threat type) is a string in the form of `SE_XXX`:

- `SE_KNOWN_VIRUS`—known threat,
- `SE_VIRUS_MODIFICATION`—modification of a known threat,
- `SE_UNKNOWN_VIRUS`—unknown threat (a suspicious object),
- `SE_ADWARE`—adware,
- `SE_DIALER`—dialer program,
- `SE_JOKE`—joke program,
- `SE_RISKWARE`—potentially dangerous software,

- SE_HACKTOOL—hacktool.

8) Archive is an object containing information about an archive, packed objects, email messages and other containers:

```
{
 "type": <string> //Archive type, can be one of the following:
         "SE_ARCHIVE" //Archive
         "SE_MAIL" //Email message
         "SE_CONTAINER" //Another container type
,"name": <string> //Archive format
}
```

9) ScanStat is an object containing scanning statistics:

```
{
 "origin": <string>, //Application at the request of which the scanning has been
performed
 // Counters for infected objects:
 "known_virus": <number>, //Number of objects infected by known threats
 "virus_modification": <number>, //Number of objects infected by a modification of a
known threat
 "unknown_virus": <number>, //Number of objects infected by unknown threats
 "adware": <number>, //Number of objects with SE_ADWARE
 "dialer": <number>, //Number of objects with SE_DIALER
 "joke": <number>, //Number of objects with SE_JOKE
 "riskware": <number>, //Number of objects with SE_RISKWARE
 "hacktool": <number>, //Number of objects with SE_HACKTOOL
 "cured": <number>, //Number of cured objects
 "quarantined": <number>, //Number of quarantined objects
 "deleted": <number> //Number of deleted objects
}
```

Managing the list of threats

To manage the list of threats that have been detected while scanning, the following HTTP API commands are available:

API command	Description
threats	Action: Get a list of identifiers of all detected threats. URI: /api/10.2/threats/ HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: an array of threat identifiers
threat_info	Action: Get information about a threat having the <*threat ID*> identifier. URI: /api/10.2/threat_info/<*threat ID*> HTTP method: GET

API command	Description
	Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), FileThreat object
`cure_threat`	Action: Attempt to cure a threat having the <*threat ID*> identifier. URI: `/api/10.2/cure_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`delete_threat`	Action: Delete the file containing the threat having the <*threat ID*> identifier. URI: `/api/10.2/delete_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`ignore_threat`	Action: Ignore the threat having the <*threat ID*> identifier. URI: `/api/10.2/ignore_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`quarantine_threat`	Action: Quarantine a threat having the <*threat ID*> identifier. URI: `/api/10.2/quarantine_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object

Each found threat has a unique integer identifier <*threat ID*>. The list of identifiers for all detected threats is returned by the `threats` command. The `threat_info`, `cure_threat`, `delete_threat`, `ignore_threat`, and `quarantine_threat` commands allow the threat identifiers from this list only.

All information about a specific threat, including action history, can be obtained using the `threat_info` request. The information is returned as the FileThreat object.

Description of used JSON objects

1) `FileThreat` is an object containing the following information:

```
{
 "threat_id": <number>, //Threat identifier
 "detection_time": <Unix time>, //Date and time of threat detection
 "report": <ScanReport>, //File scan report
 "stat": <FileStat>, //Information about the file
 "origin": <string>, //Name of a component that detected the threat
 "origin_pid": <number>, //PID of the component that detected the threat
 "task_id": <number>, //Identifier of the scanning task for the scanning engine
 "history": [<ActionResult>] //History of actions applied to the threat (an array)
}
```

The `report` field contains a ScanReport object; the `stat` field contains a FileStat object, and the `history` field contains an array of ActionResult objects (the history of actions applied to the file).

2) `ScanReport` is an object containing information about the file with a threat:

```
{
 "object": <string>, //File system object containing the threat
 "size": <number>, //Size (in bytes) of the file containing the threat
 "virus": [<VirusInfo>], //List of details about the detected threats
 *"error": <string>, //Error message
 "heuristic_analysis": <boolean> //Whether the heuristic analysis has been enabled
(true—enabled, false—disabled)
}
```

The parameter marked with `*` is optional.

The `virus` field is an array of VirusInfo objects containing information about all detected threats. The `error` field is only present if an error has occurred.

3) `FileStat` is an object containing file statistics:

```
{
 "dev": <number>, //Device
 "ino": <number>, //Index node (inode)
 *"size": <number>, //File size
 *"uid": <User>, //User/owner identifier
 *"gid": <Group>, //Owner group identifier
 *"mode": <number>, //File access mode
 *"mtime": <Unix time>, //File modification date and time
 *"ctime": <Unix time>, //File creation date and time
 *"rsrc_size": <number>,
 *"finder_info": <string>,
 *"ext_finder_info": <string>,
 *"uchg": <string>,
 *"volume_name": <string>, //Volume name
 *"volume_root": <string>, //Volume root (mount point)
 *"xattr": [<Xattr>] //Extended information about the file
}
```

The parameters marked with `*` are optional.

The `xattr` field contains an array of `Xattr` objects. This object contains two string-type fields: `name` and `value`. The `uid` and `gid` fields represent `User` and `Group` objects that store information about a user/owner and an owner group respectively. These objects contain two fields each:

- `uid` (`gid`)—numeric ID of the user (group);
- `username` (`groupname`)—name of the user (group) as a string.

4) `ActionResult` is an object containing information about the action applied to the file and its result:

```
{
 "action": <string>, //Applied action
 "action_time": <Unix time>, //Date and time of applying the action
 "result": <string>, //Result of applying the action
 "cure_report": <ScanReport> //Report on applying the action
}
```

The `cure_threat`, `delete_threat`, `ignore_threat` and `quarantine_threat` commands return an empty object when run successfully. If the requested action to be applied to the threat fails (for example, the threat cannot be neutralized), an Error object is returned instead of an empty object.

Managing quarantine

To manage quarantined threats, HTTP API provides the following commands:

API command	Description
`quarantine`	Action: List identifiers of quarantined objects. URI: `/api/10.2/quarantine/` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), array of QuarantineId objects (quarantined objects)
`qentry_info`	Action: Get information about the quarantined object having the *<entry ID>* identifier. URI: `/api/10.2/qentry_info/`*<entry ID>* HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), QEntry object
`cure_qentry`	Action: Attempt to cure the quarantined object with the *<entry ID>* identifier.

API command	Description
	URI: `/api/10.2/cure_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`delete_qentry`	Action: Delete the quarantined object with the *<entry ID>* identifier. URI: `/api/10.2/delete_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`restore_qentry`	Action: Restore the quarantined object with the *<entry ID>* identifier to its original location. URI: `/api/10.2/restore_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object

The `quarantine` command returns the array of QuarantineId objects, each of which contains the identifier of a quarantined object. The identifier consists of two parts: `chunk_id` and `entry_id`.

Description of used JSON objects

1) `QuarantineId` is an object containing parts of the compound identifier of a quarantined object:

```
{
 "chunk_id": <string>,
 "entry_id": <string>
}
```

A combination of these two fields in the form of *<entry_id>@<chunk_id>* represents the identifier of a quarantined object. This identifier is provided in all requests for performing actions under any quarantined object using the `qentry_info`, `cure_qentry`, `delete_qentry` and `restore_qentry` commands. The `qentry_info` command allows to get the detailed information about a quarantined object with a specified identifier. This command returns a QEntry object.

2) `QEntry` is an object containing information about a quarantined object:

```
{
 "entry_id": <string>, //Parts of the identifier
 *"chunk_id": <string>, //of the quarantined object
 *"quarantine_dir": <string>, //Quarantine directory
 "restore_path": <string>, //Path to restore
 "creation_time": <Unix time>, //Date and time when the object was quarantined
 "report": <ScanReport>, //Object scan report (see ScanReport above)
 "stat": <FileStat>, //File statistics (see FileStat above)
 *"history": [<QEntryOperation>], //History of operations performed on the object
 *"who": <RemoteUser>, //Remote owner of the file (if the file was quarantined from a
file server storage having a network access)
 *"detection_time": <Unix time>, //Date and time of threat detection
 *"origin": <string> //Component that detected the threat
}
```

The parameters marked with `*` are optional.

The `report` field contains a ScanReport object, the `stat` field contains a FileStat object, and the `history` field contains the history of actions applied to the quarantined object. Each action record is described by a QEntryOperation object. The optional `who` field contains information about the remote user in the form of a RemoteUser object.

3) `QEntryOperation` is an object containing information about an operation applied to the quarantined object:

```
{
 "action": <string>, //Action applied to the object (see the designations below)
 "action_time": <Unix time>, //Date and time of applying the action
 "result": <string>, //Error of applying the action (a code in the form of EC_XXX)
 *"restore_path": <string>, //Path for restoring the quarantined object (with action =
"QENTRY_ACTION_RESTORE")
 *"rescan_report": <ScanReport> //Report on rescanning (with action =
"QENTRY_ACTION_RESCAN")
}
```

The parameters marked with `*` are optional.

The `action` field can have the following values:

- `QENTRY_ACTION_DELETE`—attempt to delete the quarantined object;
- `QENTRY_ACTION_RESTORE`—attempt to restore the quarantined object;
- `QENTRY_ACTION_RESCAN`—attempt to rescan the quarantined object;
- `QENTRY_ACTION_CURE`—attempt to cure the quarantined object.

4) `RemoteUser` is an object containing information about the remote user who owns the file (if the file was relocated to quarantine from the server storage):

```
{
 *"ip": <string>, //User IP address
 *"user": <string>, //User name
 *"domain": <string> //User domain
}
```

The parameters marked with `*` are optional.

The `cure_qentry`, `delete_qentry` and `restore_qentry` commands return an empty object if the command execution was successful. In case the command failed, an Error object will be returned in the response.

HTTP API usage examples

To test the HTTP API, you can use the `curl` utility. The structure and format of the request can be provided as follows:

```
$ curl https://<HTTPD.AdminListen>/<HTTP API URI> -k -X <HTTP method>
 [-H 'Content-Type: application/json' --data-binary '@<JSON object file>']
 [-c <cookie file> [-b <cookie file>]] [> <result file>]
```

- the `-k` option indicates that `curl` does not have to verify the used SSL certificate;
- the `-X` option specifies the HTTP method being used (GET or POST);
- the `-H` option is used to add the `Content-Type: application/json` header;
- the `--data-binary` (or `-d`) option is used to add a JSON object read from a text file to the body request;
- if SCS is used for authorization, the files containing the sent and received SCS cookies must be specified in the `-b` and `-c` parameters respectively.

The detailed description of the `curl` utility options can be displayed using the `curl --help` and `man curl` commands.

1. **Authorize a client** by specifying a user name and a password (for SCS).

   An AuthOptions object in the JSON format must be stored in advance in the `user.json` file, for example:

   ```
   {"user":"<username>","password":"<passphrase>"}
   ```

   Request:

   ```
   $ curl https://127.0.0.1:4443/api/10.2/login -k -X POST -H 'Content-Type:
   application/json' --data-binary '@user.json' -c cookie.file
   ```

   Response:

   ```
   HTTP/1.0 200 OK
   Content-Type: application/json
   Content-Length: 2
   Set-Cookie: DWToken=6QXy4wn_JGov9A1GohWP_kvMK3dN6ccKegjNgKcmHpb_AqSrHg9cNX_yFJhxPDgr|
   MTQ2Mjg3Mzg4NQ==|cWd4Ow==|GywBUVOhU4w2LF_BKT5frg==|kR_rip5nrpxWjJ2dfZ7Xfmvi3rE=;
   Secure; HttpOnly; Max-Age: 900; Path=/
   Pragma: no-cache

   {}
   ```

   The `Set-Cookie` header contains an SCS cookie to be used in all further requests to the HTTP API. If the authorization was successful, the body of the response will contain an empty object. If the user has not been authorized, the Error object is returned:

   ```
   HTTP/1.0 403 Forbidden
   Content-Type: application/json
   Content-Length: 35
   ```

```
Pragma: no-cache

{"error":{"code":"EC_AUTH_FAILED"}}
```

2. **Get information about the threat** having the *ID = 1* identifier:

   Request:

```
$ curl https://127.0.0.1:4443/api/10.2/threat_info/1 -k -X GET -c cookie.file -b
cookie.file
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 574
Set-Cookie: DWToken=<…>;
 Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{"threat_id":1,"detection_time":1462881660,
"report":{"object":"/sites/site1/eicar.com.txt","size":68,"packer":[],
"virus":[{"type":"SE_KNOWN_VIRUS","name":"EICAR Test File (NOT a Virus!)"}],
"heuristic_analysis":true,"core_fingerprint":"0D2DD5A869DAB7AE354153A4D5F70F45",
"item":[],"log":[],"user_time":0,"system_time":0},"stat":{"dev":2049,"ino":898,
"size":68,"uid":{"uid":1000,"username":"user"},"gid":{"gid":1000,"groupname":"user"},
"mode":33204,"mtime":1441028214,"ctime":1460738554,"xattr":[]},
"origin":"APP_COMMAND_LINE_TOOL","origin_pid":2726,"task_id":1,"history":[]}
```

3. **Quarantine** the threat having the *ID = 1* identifier:

   Request:

```
$ curl -v -c cookie.jar -b cookie.jar -k -X POST -H 'Content-Type:application/json'
https://127.0.0.1:4443/api/10.2/quarantine_threat/1
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 2
Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{}
```

4. **View information about the quarantined object**:

   Request:

```
$ curl -v -k -X GET -c cookie.jar -b cookie.jar
https://127.0.0.1:4443/api/10.2/qentry_info/3070d3ce-7b6e-4143-9d9f-
89ba3473a781@801:2108d
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 781
Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{"entry_id":"3070d3ce-7b6e-4143-9d9f-89ba3473a781","chunk_id":"3830313A3231303864",
"quarantine_dir":"2F686F6D652F757365722F2E636F6D2E64727765622E71756172616E74696E65",
"restore_path":"2E2E2F7473742F65696361722E636F6D2E747874","creation_time":1462888884,
```

```
"report":{"object":"/home/user/tst/eicar.com.txt","size":68,"packer":[],
"virus":[{"type":"SE_KNOWN_VIRUS","name":"EICAR Test File (NOT a Virus!)"}],
"heuristic_analysis":true,"core_fingerprint":"467CD4C6D423C55448B71CD5B8152776",
"item":[],"log":[],"user_time":0,"system_time":0},"stat":{"dev":2049,"ino":898,
"size":68,"uid":{"uid":1000,"username":"user"},"gid":{"gid":1000,"groupname":"user"},
"mode":33204,"mtime":1441028214,"ctime":1462888421,"xattr":[]},"history":[],
"detection_time":1462888667,"origin":"APP_COMMAND_LINE_TOOL"}
```

9.12. Dr.Web SNMPD

Dr.Web SNMPD is an SNMP agent designed to integrate Dr.Web Industrial for Unix File Servers with monitoring systems running the SNMP protocol. This integration allows you to track the status of the Dr.Web Industrial for Unix File Servers components, as well as collect statistics on the detection and neutralization of threats. The agent supports providing monitoring systems or any SNMP managers with the following information:

- Status of any Dr.Web Industrial for Unix File Servers component.
- Number of detected threats of various types (according to the Dr.Web classification).

Moreover, the agent sends SNMP trap notifications upon detection of a threat and upon failures in neutralization of detected threats. The agent supports SNMP protocol of version 2c and 3.

Description of the information which can be sent by the agent is stored in a special section of MIB (*Management Information Base*) created by Doctor Web. In the MIB section, defined by Dr.Web for Unix-like operating systems, the following information is specified:

1. Formats of SNMP trap notifications about detection and neutralizing of threats and about errors related to the Dr.Web Industrial for Unix File Servers components.
2. Dr.Web Industrial for Unix File Servers operation statistics.
3. Dr.Web Industrial for Unix File Servers component status.

For more details about information that can be obtained over the SNMP protocol, refer to the corresponding section.

9.12.1. Operating Principles

In this section

- General information
- Integration with the system SNMP agent

General information

By default, the component is run automatically upon Dr.Web Industrial for Unix File Servers startup. When run, the component structures data according to the structure described in Dr.Web SNMP MIB and waits for requests to receive data from external SNMP managers. The component receives information on the status of the Dr.Web Industrial for Unix File Servers components and notifications on detected threats directly from the Dr.Web ConfigD configuration daemon.

Threats can be detected by the scan engine during scanning initiated by Dr.Web Industrial for Unix File Servers components. Once any of threats is detected, the appropriate count (of this

threat type) is incremented by one and all SNMP managers that can receive notifications get an SNMP trap notifying on the detected threat.



Collected values of counters (for example, counters of detected threats) are not saved between launches of Dr.Web SNMPD. Thus, once Dr.Web SNMPD is relaunched for any reason (including general restart of Dr.Web Industrial for Unix File Servers), the collected values of counters are reset to zero.

Integration with the system SNMP agent

To enable correct operation of Dr.Web SNMP agent if the main system SNMP agent `snmpd` (`net-snmp`) already operates on the server, configure transmission of SNMP requests through the Dr.Web MIB branch from `snmpd` to Dr.Web SNMPD. For that purpose, edit the `snmpd` configuration file (usually, `/etc/snmp/snmpd.conf` for GNU/Linux or `/etc/snmpd.config` for FreeBSD), by adding the following line to it:

```
proxy -v <version> -c <community> <address>:<port> <MIB branch>
```

Where:

- <*version*>—SNMP version in use (2c, 3);
- <*community*>—"community string" used by Dr.Web SNMPD;
- <*address*>:<*port*>—network socket listened by Dr.Web SNMPD;
- <*MIB branch*>—OID of the MIB branch containing descriptions of variables and SNMP notifications (*trap*) used by Dr.Web (the OID equals `.1.3.6.1.4.1.29690`).

If you are using the default settings of Dr.Web SNMP agent, then the added line should look like this:

```
proxy -v 2c -c public localhost:50000 .1.3.6.1.4.1.29690
```



Since port 161 in this case will be used by the system standard `snmpd`, it is required to specify another port for Dr.Web SNMPD in the `ListenAddress` parameter (in this example, `50000`).

9.12.2. Command-Line Arguments

To start the Dr.Web SNMPD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-snmpd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-snmpd [<parameters>]
```

Dr.Web SNMPD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-snmpd --help
```

This command outputs short help information about the Dr.Web SNMPD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-snmpd` command.

9.12.3. Configuration Parameters

The component uses configuration parameters specified in the `[SNMPD]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component.

Parameter	Description
	If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-snmpd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-snmpd`
`Start` *{boolean}*	Launch/do not launch the component by the Dr.Web ConfigD configuration daemon. When you specify the `Yes` value for this parameter, it the configuration daemon will start the component immediately; and when you specify the `No` value, the configuration daemon will terminate the component immediately. Default value: `No`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`SnmpVersion` *{V2c | V3}*	Current version of SNMP protocol (*SNMPv2c* or *SNMPv3*). Default value: `V2c`
`ListenAddress` *{address}*	Address (IP address and port) listened by Dr.Web SNMPD, which is waiting for client connections (SNMP managers).  Interoperability with `snmpd` requires specifying a port different from the standard port (`161`). In addition, `snmpd` must be configured for proxying. Default value: `127.0.0.1:161`

Parameter	Description
`TrapReceiver` *{address list}*	List of addresses (IP address and port) to which Dr.Web SNMPD sends *SNMP trap* notifications when Dr.Web Industrial for Unix File Servers components detect a threat. Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list. 1. Adding values to the configuration file. • Two values per line: `[SNMPD]` `TrapReceiver = "192.168.0.1:1234", "10.20.30.45:5678"` • Two lines (one value per line): `[SNMPD]` `TrapReceiver = 192.168.0.1:1234` `TrapReceiver = 10.20.30.45:5678` 2. Adding the values using the `drweb-ctl cfset` command: `# drweb-ctl cfset SNMPD.TrapReceiver -a 192.168.0.1:1234` `# drweb-ctl cfset SNMPD.TrapReceiver -a 10.20.30.45:5678` Default value: *(not specified)*
`V2cCommunity` *{string}*	"SNMP read community" string for authentication of SNMP managers (*SNMPv2c* protocol) when Dr.Web MIB variables are accessed for reading. The parameter is used if `SnmpVersion = V2c`. Default value: `public`
`V3EngineId` *{string}*	Identifier (string) of *Engine ID* for *SNMPv3* (according to RFC 3411). Default value: `800073FA044452574542`
`V3UserName` *{string}*	User name for authentication of SNMP managers (*SNMPv3* protocol) when Dr.Web MIB variables are accessed for reading. The parameter is used if `SnmpVersion = V3`. Default value: `noAuthUser`

Parameter	Description
`V3Auth` *{SHA(<pwd>) | MD5(<pwd>) | None}*	Method to authenticate SNMP managers (*SNMPv3* protocol) when Dr.Web MIB variables are accessed for reading. Allowed values: • `SHA(<PWD>)`—SHA hash of the password is used (<*PWD*> strings); • `MD5(<PWD>)`—MD5 hash of the password is used (<*PWD*> strings); • `None`—authentication is disabled; where <*PWD*> is a plain text password. When specifying the parameter value from the command line, you may need to escape the brackets by using the slash mark `\` in some shells. Examples: 1. Parameter value in the configuration file: `V3Auth = MD5(123456)` 2. Specifying the same parameter value from the command line using the `drweb-ctl cfset` command: `drweb-ctl cfset SNMPD.V3Auth MD5\(123456\)` The parameter is used if `SnmpVersion = V3`. Default value: `None`
`V3Privacy` *{DES(<secret>) | AES128(<secret>) | None}*	Encryption method for SNMP messages (*SNMPv3* protocol). Allowed values: • `DES(<secret>)`—DES encryption algorithm; • `AES128(<secret>)`—AES128 encryption algorithm; • `None`—SNMP-messages are not encrypted; where <*secret*> is a secret key shared by the manager and the agent (*plain text*). When specifying the parameter value from the command line, you may need to escape the brackets by using the slash mark `\` in some shells. Examples: 1. Parameter value in the configuration file: `V3Privacy = AES128(supersecret)`

Parameter	Description
	2. Specifying the same parameter value from the command line using the `drweb-ctl cfset` command: `drweb-ctl cfset SNMPD.V3Privacy AES128\(supersecret\)` The parameter is used if `SnmpVersion = V3`. Default value: `None`

9.12.4. Integration with Monitoring Systems

In this section

- Integration with the Munin monitoring system
  - Connecting a host to Munin
- Integration with the Zabbix monitoring system
  - Connecting a host to Zabbix
  - Configuring receipt of SNMP trap notifications for Zabbix
- Integration with the Nagios monitoring system
  - Connecting a host to Nagios
  - Configuring Nagios
  - Configuring receipt of SNMP trap notifications for Nagios

The Dr.Web SNMP agent can act as a data provider for any monitoring system that uses SNMP *2c* or *3*. The list of data available for monitoring and their structure are provided in the Dr.Web MIB description file `DRWEB-SNMPD-MIB.txt` supplied together with Dr.Web Industrial for Unix File Servers. This file is located in the directory `/opt/drweb.com/share/drweb-snmpd/mibs` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/mibs` for FreeBSD.

For ease of configuration, the component is supplied with the required configuration templates for popular monitoring systems such as Munin, Zabbix and Nagios.

Configuration templates for monitoring systems are located in the directory `/opt/drweb.com/share/drweb-snmpd/connectors` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors` for FreeBSD.

Integration with the Munin monitoring system

The Munin monitoring system includes a munin centralized server (master), which collects statistics from munin-node clients installed locally on hosts to be monitored. At the request of the server, each monitoring client collects data about monitored host operation by starting *plug-ins* that provide data to be sent to the server.

To connect Dr.Web SNMPD to the Munin monitoring system, ready-to-use Dr.Web data collection plug-ins used by munin-node are supplied. These plug-ins are located in the directory `/opt/drweb.com/share/drweb-snmpd/connectors/munin/plugins` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/munin/plugins` for FreeBSD and collect data required for drawing the following graphs:

- a number of detected threats;
- file scan statistics.

These plug-ins support SNMP 1, 2c and 3. Based on these template plug-ins, you can create any other plug-ins to poll the status of Dr.Web Industrial for Unix File Servers components via Dr.Web SNMPD.

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/munin` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/munin` for FreeBSD contains the following files.

File	Description
`plugins/snmp__drweb_malware`	`munin-node` plug-in for polling Dr.Web SNMPD via SNMP to get a number of threats detected by Dr.Web Industrial for Unix File Servers on the host.
`plugins/snmp__drweb_filecheck`	`munin-node` plug-in for polling Dr.Web SNMPD via SNMP to gather statistics on scanning files by Dr.Web Industrial for Unix File Servers on the host.
`plugin-conf.d/drweb.cfg`	Example of the `munin-node` configuration for the runtime environment variables of the Dr.Web plug-ins.

Connecting a host to Munin

This instruction assumes that the Munin monitoring system is already deployed on the monitoring server, and the monitored host features installed and functioning Dr.Web SNMPD (optionally, in proxy mode together with snmpd) and `munin-node`.

1. Monitored host configuration
   - Copy the `snmp__drweb_*` files to the directory containing `munin-node` plug-in libraries. This path depends on the OS, for example, the path is `/usr/share/munin/plugins` on Debian and Ubuntu.
   - Configure `munin-node` by connecting the supplied Dr.Web plug-ins to it. To do this, use the `munin-node-configure` utility, which is distributed with `munin-node`.

     For example, the command:

     ```
     $ munin-node-configure --shell --snmp localhost
     ```

will display a list of commands for creation of required symbolic links to plug-ins on a terminal screen. Copy and run them in the command line. The specified command presumes that:

1) `munin-node` is installed on the same host as Dr.Web SNMPD. Otherwise, specify a valid FQDN or IP address of the monitored host instead of `localhost`.
2) Dr.Web SNMPD uses SNMP 2c. Otherwise, specify the correct SNMP version for the `munin-node-configure` command. This command accepts a set of parameters for flexible configuration of plug-ins, for example, you can specify the SNMP version in use, a port that is used by an SNMP agent at the monitored host, a value of *community string* and so on. If required, refer to the manual for the `munin-node-configure` command.

- If necessary, define or redefine parameter values of the environment where the installed Dr.Web plug-ins for `munin-node` must be run. As the environment parameters, the value of *community string*, the port used by the SNMP agent and so on are used. These parameters must be defined in the `/etc/munin/plugin-conf.d/drweb` file (create it if required). You can use the supplied `drweb.cfg` file as an example.
- In the `munin-node` configuration file (`munin-node.conf`), specify a regular expression to include IP addresses of hosts that are allowed to connect munin servers (masters) to `munin-node` on the current host for receiving the values of the monitored parameters, for example:

  ```
  allow ^10\.20\.30\.40$
  ```

  In this case, only a host with the IP address of `10.20.30.40` is allowed to receive the parameters of that host.
- Restart `munin-node`, for example, by running the command:

  ```
  # service munin-node restart
  ```

2. Munin server (master) configuration

   Add the address and identifier of the monitored host to the Munin configuration file (`munin.conf`), which is located, by default, in the `/etc` directory (`/etc/munin/munin.conf` on Debian and Ubuntu):

   ```
   [<ID>;<hostname>.<domain>]
   address <host IP address>
   use_node_name yes
   ```

   where <*ID*> is the displayed host identifier, <*hostname*> is the name of the host, <*domain*> is the name of the domain, <*host IP address*> is the IP address of the host.

For official documentation on configuring the Munin monitoring system, follow the link.

Integration with the Zabbix monitoring system

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix` for FreeBSD contains the following template files required for connecting Dr.Web SNMPD to the Zabbix monitoring system:

File	Description
`zbx_drweb.xml`	Template for description of the monitored host that features installed Dr.Web Industrial for Unix File Servers.
`snmptt.drweb.zabbix.conf`	Settings of the `snmptt` utility, which receives *SNMP trap* notifications.

The template for description of the monitored host features:

- A set of descriptions of counters ("*items*" in the terminology of Zabbix). By default, the template is set to be used with SNMP v2.
- A set of predefined graphs: a number of scanned files and distribution of detected threats by their types.

Connecting a host to Zabbix

In the present instruction, it is assumed that the Zabbix monitoring system is already properly deployed on the monitoring server and the monitored host features an installed and properly functioning Dr.Web SNMPD (optionally, in proxy mode together with `snmpd`). Furthermore, if you want to receive *SNMP trap* notifications from the monitored host (including notification of threats detected by Dr.Web Industrial for Unix File Servers on the protected server), install the `net-snmp` package on the monitoring server (the `snmptt` and `snmptrapd` standard tools are used).

1. In the Zabbix web interface, on the **Configuration** → **Templates** tab, import the template of the monitored host from the file `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix/zbx_drweb.xml` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix/zbx_drweb.xml` for FreeBSD.
2. Add the monitored host to the list of hosts (at **Hosts** → **Create host**). Specify parameters of the host and valid settings of the SNMP interface (they must match the settings of `drweb-snmpd` and snmpd on the host):
   - **Host** tab:

     Host name: *drweb-host*

     Visible name: *DRWEB_HOST*

     Groups: select **Linux servers**

Snmp interfaces: click **Add** and specify the IP address and the port used by Dr.Web SNMPD (by default, it is assumed that Dr.Web SNMPD operates on the local host, so address *127.0.0.1* and standard port *161* are indicated here).

- **Templates** tab:

  Click **Add**, select *DRWEB*, click **Select**.

- **Macros** tab:

  Macro: *{$SNMP_COMMUNITY}*

  Value: select *read community* for SNMP V2c (*public* by default).

  Click **Save**.



The *{$SNMP_COMMUNITY}* macro can be specified directly in the host template.

By default, the imported *DRWEB* template is configured for SNMP v2. If you need to use another version of SNMP, edit the template on the corresponding page.

3. After the template is bound to the monitored host, if SNMP settings are valid, the Zabbix monitoring system will start to collect data for counters (*items*) of the template. The collected data will be displayed on the following tabs of the web interface: **Monitoring** → **Latest Data** and **Monitoring** → **Graphs**.
4. A special *item drweb-traps* is used for collecting *SNMP trap* notifications from Dr.Web SNMPD. The log of received *SNMP trap* notifications is available on the page **Monitoring** → **Latest Data** → **drweb-traps** → **history**. To collect notifications, Zabbix uses the standard `snmptt` and `snmptrapd` utilities from the `net-snmp` package. For details on how to configure these utilities for receiving *SNMP trap* notifications from Dr.Web SNMPD, see below.
5. If necessary, you can configure a trigger for the added monitored host. The trigger will change its state upon receiving *SNMP trap* notifications from Dr.Web SNMPD. Changing the state of this trigger can be used as an event source for generating corresponding notifications. A trigger for the monitored host is added in a common way. The example below shows an expression specified in the **trigger expression** field for the aforesaid trigger.

   - For Zabbix *2.x*:

     ```
     ({TRIGGER.VALUE}=0 & {DRWEB:snmptrap[.*\.1\.3\.6\.1\.4\.1\.29690\..*].nodata(60)}
     =1 )|({TRIGGER.VALUE}=1 &
     {DRWEB:snmptrap[.*\.1\.3\.6\.1\.4\.1\.29690\..*].nodata(60)}=0)
     ```

   - For Zabbix *3.x*:

     ```
     ({TRIGGER.VALUE}=0 and {drweb-host:snmptrap[".29690."].nodata(60)}=1 ) or
     ({TRIGGER.VALUE}=1 and {drweb-host:snmptrap[".29690."].nodata(60)}=0 )
     ```

   This trigger is activated (its value is set to 1) if the log of *SNMP trap* notifications from Dr.Web SNMPD was updated within the last minute. If the log was not updated within the last minute, the trigger is deactivated (its value is set to 0).

It is recommended that a notification type different from *Not classified*, for example, *Warning*, is indicated in the **Severity** field for this trigger.

Configuring receipt of SNMP trap notifications for Zabbix

1. On the monitored host, in Dr.Web SNMPD settings (the `TrapReceiver` parameter), specify an address to be listened by `snmptrapd` on the Zabbix host, for example:

   ```
   SNMPD.TrapReceiver = 10.20.30.40:162
   ```

2. In the `snmptrapd` configuration file (`snmptrapd.conf`), specify the same address and an application for processing received *SNMP trap* notifications (in this example, `snmptthandler`, which is a component of `snmptt`):

   ```
   snmpTrapdAddr 10.20.30.40:162
   traphandle default /usr/sbin/snmptthandler
   ```

   Add the following line to the file, so that `snmptt` does not discard *SNMP trap* notifications sent by Dr.Web SNMPD, as unknown:

   ```
   outputOption n
   ```

3. The `snmptthandler` component stores received *SNMP trap* notifications on the file on the disk in accordance with the specified format, which must correspond to the regular expression set in the host template for Zabbix (the *drweb-traps item*). The format of the stored message on the *SNMP trap* notification is specified in the file `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix/snmptt.drweb.zabbix.conf` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix/snmptt.drweb.zabbix.conf` for FreeBSD, which must be copied to the `/etc/snmp` directory.

4. Furthermore, the path to the format files must be specified in the `snmptt.ini` configuration file:

   ```
   [TrapFiles]
   # A list of snmptt.conf files (this is NOT the snmptrapd.conf file).
   # The COMPLETE path and filename. Ex: '/etc/snmp/snmptt.conf'
   snmptt_conf_files = <<END
   /etc/snmp/snmptt.conf
   /etc/snmp/snmptt.drweb.zabbix.conf
   END
   ```

   After that, restart `snmptt` if it was started in daemon mode.

5. Specify the following settings (or check if they are already specified) in the configuration file of the Zabbix server (`zabbix-server.conf`):

   ```
   SNMPTrapperFile=/var/log/snmptt/snmptt.log
   StartSNMPTrapper=1
   ```

   where `/var/log/snmptt/snmptt.log` is a log file used by `snmptt` to register information about received SNMP trap notifications.

For official documentation on Zabbix, follow the link.

Integration with the Nagios monitoring system

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/nagios` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/nagios` for FreeBSD contains the following Nagios example configuration files required for connecting Dr.Web SNMPD to the Nagios monitoring system:

File	Description
`nagiosgraph/rrdopts.conf-sample`	Example of the RRD configuration file
`objects/drweb.cfg`	Configuration file describing *drweb* objects
`objects/nagiosgraph.cfg`	Configuration file of the Nagiosgraph component for graph plotting, used by Nagios
`plugins/check_drweb`	Script for collecting data from a Dr.Web Industrial for Unix File Servers host
`plugins/eventhandlers/submit_check_result`	Script for handling *SNMP trap* notifications
`snmp/snmptt.drweb.nagios.conf`	Settings of the `snmptt` utility, which receives *SNMP trap* notifications

Connecting a host to Nagios

In the present instruction, it is assumed that the Nagios monitoring system is already properly deployed on the monitoring server (including configuring the web server and the Nagiosgraph graphical tool) and the monitored host features installed and properly functioning Dr.Web SNMPD (optionally, in proxy mode together with snmpd). Furthermore, if you want to receive an *SNMP trap* notification from the monitored host (including notifications of threats detected by Dr.Web Industrial for Unix File Servers on the protected server), install the `net-snmp` package on the monitoring server (the `snmptt` and `snmptrapd` standard utilities are used).

In the current manual, the following path conventions are used (actual paths depend on the operating system and Nagios installation):

- <*NAGIOS_PLUGINS_DIR*>—directory with Nagios plug-ins, for example, `/usr/lib64/nagios/plugins`.
- <*NAGIOS_ETC_DIR*>—directory with Nagios settings, for example, `/etc/nagios`.
- <*NAGIOS_OBJECTS_DIR*>—directory with Nagios objects, for example, `/etc/nagios/objects`.
- <*NAGIOSGRAPH_DIR*>—Nagiosgraph directory, for example, `/usr/local/nagiosgraph`.
- <*NAGIOS_PERFDATA_LOG*>—file in which Nagios stores results obtained from running service scan commands (must be the same as the `perflog` file from

<*NAGIOSGRAPH_DIR*>/etc/nagiosgraph.conf). Records from this file are read by the <*NAGIOSGRAPH_DIR*>/bin/insert.pl script and are written to the corresponding RRA archives of RRDtool.

Configuring Nagios

1. Copy the check_drweb file to the <*NAGIOS_PLUGINS_DIR*> directory and the drweb.cfg file to the <*NAGIOS_OBJECTS_DIR*> directory.
2. Add Dr.Web Industrial for Unix File Servers hosts to be monitored to the *drweb* group (Dr.Web SNMPD must be running on them). By default, only the *localhost* local host is added to this group.
3. If required, edit the check_drweb command to communicate with Dr.Web SNMPD on *drweb* hosts via the snmplwalk utility:

   ```
   snmpwalk -c public -v 2c $HOSTADDRESS$:161
   ```

   Specify the correct SNMP version and such parameters as *community string*, or authentication parameters as well as a port. The $HOSTADDRESS$ variable must be included in the command (as this variable is later automatically substituted by Nagios with the correct host address when the command is invoked). It is not required to indicate OID in the command. It is also recommended that you specify the command together with the full path to the executable file (usually /usr/local/bin/snmpwalk).
4. Connect *DrWeb* objects in the <*NAGIOS_ETC_DIR*>/nagios.cfg configuration file by appending it with the following line:

   ```
   cfg_file=<NAGIOS_OBJECTS_DIR>/drweb.cfg
   ```
5. Add RRDtool settings for *DrWeb* graphs from the rrdopts.conf-sample file to the <*NAGIOSGRAPH_DIR*>/etc/rrdopts.conf file.
6. Configure the Nagiosgraph component if it is still not configured:
   - Copy the nagiosgraph.cfg file to the <*NAGIOS_OBJECTS_DIR*> directory and edit the path to the insert.pl script in the process-service-perfdata-for-nagiosgraph command, for example, as follows:

     ```
     $ awk '$1 == "command_line" { $2 = "<NAGIOSGRAPH_DIR>/bin/insert.pl" }{ print }'
     ./objects/nagiosgraph.cfg > <NAGIOS_OBJECTS_DIR>/nagiosgraph.cfg
     ```
   - Connect this file in the <*NAGIOS_ETC_DIR*>/nagios.cfg configuration file by appending the following line to it:

     ```
     cfg_file=<NAGIOS_OBJECTS_DIR>/nagiosgraph.cfg
     ```

7. Verify Nagios parameter values in the *<NAGIOS_ETC_DIR>*/nagios.cfg configuration file:

```
check_external_commands=1
execute_host_checks=1
accept_passive_host_checks=1
enable_notifications=1
enable_event_handlers=1

process_performance_data=1
service_perfdata_file=/usr/nagiosgraph/var/rrd/perfdata.log
service_perfdata_file_template=$LASTSERVICECHECK$||$HOSTNAME$||$SERVICEDESC$||$SERVIC
EOUTPUT$||$SERVICEPERFDATA$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=30
service_perfdata_file_processing_command=process-service-perfdata-for-nagiosgraph

check_service_freshness=1
enable_flap_detection=1
enable_embedded_perl=1
enable_environment_macros=1
```

Configuring receipt of SNMP trap notifications for Nagios

1. On the monitored host, in Dr.Web SNMPD settings (the TrapReceiver parameter), specify an address to be listened by snmptrapd on the Nagios host, for example:

```
SNMPD.TrapReceiver = 10.20.30.40:162
```

2. Ensure the existence of the *<NAGIOS_PLUGINS_DIR>*/eventhandlers/submit_check_result script to be run when *SNMP trap* notifications are received. If such script does not exist, copy the submit_check_result file to this location from the directory /opt/drweb.com/share/drweb-snmpd/connectors/nagios/plugins/eventhandlers/ for GNU/Linux or /usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/nagios/plugins/eventhandlers/ for FreeBSD. Change the path in this file; the path is specified in the CommandFile parameter. It must have the same value as the command_file parameter in the *<NAGIOS_ETC_DIR>*/nagios.cfg file.
3. Copy the snmptt.drweb.nagios.conf file to the /etc/snmp/snmp/ directory. Change the path to the submit_check_result script in this file, for example, by running the command:

```
$ awk '$1 == "EXEC" { $2 = <NAGIOS_PLUGINS_DIR>/eventhandlers/submit_check_result }
{ print}' ./snmp/snmptt.drweb.nagios.conf > /etc/snmp/snmp/snmptt.drweb.nagios.conf
```

4. Add the "/etc/snmp/snmptt.drweb.nagios.conf" line to the /etc/snmp/snmptt.ini file. After that, restart snmptt if it was started in daemon mode.

After all required Nagios configuration files are added and edited, start Nagios in debug mode by running the command:

```
# nagios -v <NAGIOS_ETC_DIR>/nagios.cfg
```

Here, Nagios will check for configuration errors. If no errors have been found, restart Nagios in the usual manner, for example, by running the command:

```
# service nagios restart
```

For official documentation on Nagios, follow the link.

9.12.5. Dr.Web SNMP MIB

The list of operating parameters of Dr.Web Industrial for Unix File Servers that can be fetched by external monitoring systems via the SNMP protocol is provided in the table below.

Parameter name	**OID parameter**	**Type and description of the parameter**
Common prefix for all names: `.iso.org.dod.internet.private.enterprises.drweb.drwebSnmpd` Common OID prefix for all names: `.1.3.6.1.4.1.29690.2`		
alert	**Asynchronous notifications about events (SNMP traps)**	
threatAlert	`.1.1`	Notification about a detected threat
`threatAlertFile`	`.1.1.1`	Name of the infected file (string)
`threatAlertType`	`.1.1.2`	Threat type (integer*)
`threatAlertName`	`.1.1.3`	Name of the threat (string)
`threatAlertOrigin`	`.1.1.4`	Identifier of the component that detected the threat (integer***)
`threatAlertRemoteIp`	`.1.1.5`	IP address of the remote host that was used to access the file (string)
`threatAlertRemoteUser`	`.1.1.6`	Name of the remote user who accessed the file (string)
`threatAlertRemoteDomain`	`.1.1.7`	IP address of the remote host (FQDN) that was used to access the file (string)
threatActionErrorAlert	`.1.2`	Notification about an error which occurred when trying to neutralize the threat

Parameter name	OID parameter	Type and description of the parameter
threatActionErrorAlertFile	.1.2.1	Name of the infected file (string)
threatActionErrorAlertType	.1.2.2	Threat type (integer*)
threatActionErrorAlertName	.1.2.3	Name of the threat (string)
threatActionErrorAlertOrigin	.1.2.4	Identifier of the component that detected the threat (integer***)
threatActionErrorAlertError	.1.2.5	Description of an error (string)
threatActionErrorAlertErrorCode	.1.2.6	Last exit code (an integer corresponding to codes from the error catalog)
threatActionErrorAlertAction	.1.2.7	Failed action (integer: 1—cure; 2—quarantine; 3—delete; 4—report; 5—ignore)
componentFailureAlert	.1.3	Notification about a component failure
componentFailureAlertName	.1.3.1	Component identifier (integer***)
componentFailureAlertExitCodeDescription	.1.3.2	Component exit code description (string)
componentFailureAlertExitCode	.1.3.3	Last exit code (an integer corresponding to codes from the error catalog)
infectedUrlAlert	.1.4	Notification about blocking a malicious URL (for HTTP/HTTPS connections)
infectedUrlAlertUrl	.1.4.1	The URL that was blocked (string)
infectedUrlAlertDirection	.1.4.2	HTTP message direction (integer: 1—request, 2—response)
infectedUrlAlertType	.1.4.3	Threat type (integer*)
infectedUrlAlertName	.1.4.4	Name of the threat (string)
infectedUrlAlertOrigin	.1.4.5	Identifier of the component that detected the threat (integer***)

Parameter name	OID parameter	Type and description of the parameter
infectedUrlAlertSrcIp	.1.4.6	IP address of a connection source (string)
infectedUrlAlertSrcPort	.1.4.7	Port of the connection source (integer)
infectedUrlAlertDstIp	.1.4.8	IP address of a connection destination point (string)
infectedUrlAlertDstPort	.1.4.9	Port of the connection destination point (integer)
infectedUrlAlertSniHost	.1.4.10	SNI of the connection destination point (for SSL connections) (string)
infectedUrlAlertExePath	.1.4.11	Executable path of the program that established the connection (string)
infectedUrlAlertUserName	.1.4.12	Name of the user as whom the program establishing the connection is running (string)
categoryUrlAlert	.1.6	Notification about blocking a URL belonging to the unwanted category
categoryUrlAlertUrl	.1.6.1	The URL that was blocked (string)
categoryUrlAlertCategory	.1.6.2	Web resource category to which the URL belongs (integer**)
categoryUrlAlertOrigin	.1.6.3	Identifier of the component that detected the threat (integer***)
categoryUrlAlertSrcIp	.1.6.4	IP address of a connection source (string)
categoryUrlAlertSrcPort	.1.6.5	Port of the connection source (integer)
categoryUrlAlertDstIp	.1.6.6	IP address of a connection destination point (string)
categoryUrlAlertDstPort	.1.6.7	Port of the connection destination point (integer)

Parameter name	OID parameter	Type and description of the parameter
categoryUrlAlertSniHost	.1.6.8	SNI of the connection destination point (for SSL connections) (string)
categoryUrlAlertExePath	.1.6.9	Executable path of the program that established the connection (string)
categoryUrlAlertUserName	.1.6.10	Name of the user as whom the program establishing the connection is running (string)
expiringLicenseAlert	.1.10	Days before license expiration (integer)
outdatedBasesAlert	.1.11	Timestamp of the last successful attempt of updating databases (*Unix time*)
stat	**Statistics on the operation of Dr.Web Industrial for Unix File Servers**	
threatCounters	.2.1	Counters of detected threats
knownVirus	.2.1.1	Number of detected known threats (counter; integer)
suspicious	.2.1.2	Number of detected suspicious objects (counter; integer)
adware	.2.1.3	Number of detected adware (counter; integer)
dialers	.2.1.4	Number of detected dialers (counter; integer)
joke	.2.1.5	Number of detected joke programs (counter; integer)
riskware	.2.1.6	Number of detected riskware (counter; integer)
hacktool	.2.1.7	Number of detected hacktools (counter; integer)
scanErrors	.2.2	Counters of the errors that occurred while files were scanned

Parameter name	OID parameter	Type and description of the parameter
sePathNotAbsolute	.2.2.1	Number of occurrences of the “Path is not absolute” error (counter; integer)
seFileNotFound	.2.2.2	Number of occurrences of the “File not found” error (counter; integer)
seFileNotRegular	.2.2.3	Number of occurrences of the “File is not a regular file” error (counter; integer)
seFileNotBlockDevice	.2.2.4	Number of occurrences of the “File is not a block device” error (counter; integer)
seNameTooLong	.2.2.5	Number of occurrences of the “Path or file name is too long” error (counter; integer)
seNoAccess	.2.2.6	Number of occurrences of the “Permission denied” error (counter; integer)
seReadError	.2.2.7	Number of occurrences of “Read error” (counter; integer)
seWriteError	.2.2.8	Number of occurrences of “Write error” (counter; integer)
seFileTooLarge	.2.2.9	Number of occurrences of the “File size too big” error (counter; integer)
seFileBusy	.2.2.10	Number of occurrences of the “File is busy” error (counter; integer)
seUnpackingError	.2.2.20	Number of occurrences of “Unpacking error” (counter; integer)
sePasswordProtecetd	.2.2.21	Number of occurrences of the “Password protected” error (counter; integer)
seArchCrcError	.2.2.22	Number of occurrences of “Archive Cyclic Redundancy Check error” (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
seArchInvalidHeader	.2.2.23	Number of occurrences of the “Invalid archive header” error (counter; integer)
seArchNoMemory	.2.2.24	Number of occurrences of the “Not enough memory to process archive” error (counter; integer)
seArchIncomplete	.2.2.25	Number of occurrences of the “Incomplete archive” error (counter; integer)
seCanNotBeCured	.2.2.26	Number of occurrences of the “Object cannot be cured” error (counter; integer)
sePackerLevelLimit	.2.2.30	Number of occurrences of the “Packer nesting level limit reached” error (counter; integer)
seArchiveLevelLimit	.2.2.31	Number of occurrences of the “Archive nesting level limit reached” error (counter; integer)
seMailLevelLimit	.2.2.32	Number of occurrences of the “Mail nesting level limit reached” error (counter; integer)
seContainerLevelLimit	.2.2.33	Number of occurrences of the “Container nesting level limit reached” error (counter; integer)
seCompressionLimit	.2.2.34	Number of occurrences of the “Compression rate limit reached” error (counter; integer)
seReportSizeLimit	.2.2.35	Number of occurrences of the “Report size limit reached” error (counter; integer)
seScanTimeout	.2.2.40	Number of occurrences of the “Scan timeout expired” error (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
seEngineCrash	.2.2.41	Number of occurrences of the “Engine crash detected” error (counter; integer)
seEngineHangup	.2.2.42	Number of occurrences of the “Engine hang-up detected” error (counter; integer)
seEngineError	.2.2.44	Number of occurrences of “Engine internal error” (counter; integer)
seNoLicense	.2.2.45	Number of occurrences of the “No valid license” error (counter; integer)
seCuringLimitReached	.2.2.47	Number of occurrences of the “Curing limit reached” error (counter; integer)
seNonSupportedDisk	.2.2.50	Number of occurrences of the “Unsupported disk” error (counter; integer)
seUnexpectedError	.2.2.100	Number of occurrences of the “Unexpected error” (counter; integer)
scanLoadAverage	.2.3	Metrics of the file scanning load
filesScannedTable	.2.3.1	Average numbers of files scanned at the request of other components
filesScannedEntry	.2.3.1.1	Component (table row; record)
filesScannedIndex	.2.3.1.1.1	Index of the component (identifier; integer***)
filesScannedOrigin	.2.3.1.1.2	Name of the component
filesScanned1min	.2.3.1.1.3	Average (for a minute) number of files scanned per second (string)
filesScanned5min	.2.3.1.1.4	Average (for 5 minutes) number of files scanned per second (string)

Parameter name	OID parameter	Type and description of the parameter
filesScanned15min	.2.3.1.1.5	Average (for 15 minutes) number of files scanned per second (string)
bytesScannedTable	.2.3.2	Average speed (in bytes) of scanning performed at the request of other components
bytesScannedEntry	.2.3.2.1	Component (table row; record)
bytesScannedIndex	.2.3.2.1.1	Index of the component (identifier; integer***)
bytesScannedOrigin	.2.3.2.1.2	Name of the component
bytesScanned1min	.2.3.2.1.3	Average (for a minute) number of bytes scanned per second (string)
bytesScanned5min	.2.3.2.1.4	Average (for 5 minutes) number of bytes scanned per second (string)
bytesScanned15min	.2.3.2.1.5	Average (for 15 minutes) number of bytes scanned per second (string)
cacheHitFilesTable	.2.3.3	Cache usage for scanned files at the request of the components
cacheHitFilesEntry	.2.3.3.1	Component (table row; record)
cacheHitFilesIndex	.2.3.3.1.1	Index of the component (identifier; integer***)
cacheHitFilesOrigin	.2.3.3.1.2	Name of the component
cacheHitFiles1min	.2.3.3.1.3	The average (for a minute) number of reports retrieved from the cache per second (string)
cacheHitFiles5min	.2.3.3.1.4	Average (for 5 minutes) number of reports retrieved from cache per second (string)
cacheHitFiles15min	.2.3.3.1.5	Average (for 15 minutes) number of reports retrieved from cache per second (string)

Parameter name	OID parameter	Type and description of the parameter
errorsTable	.2.3.4	Number of errors during scanning performed at the request of the components
errorsEntry	.2.3.4.1	Component (table row; record)
errorsIndex	.2.3.4.1.1	Index of the component (identifier; integer***)
errorsOrigin	.2.3.4.1.2	Name of the component
errors1min	.2.3.4.1.3	Average (for a minute) number of scanning errors per second (string)
errors5min	.2.3.4.1.4	Average (for 5 minutes) number of scanning errors per second (string)
errors15min	.2.3.4.1.5	Average (for 15 minutes) number of scanning errors per second (string)
info	**Information about the current state of Dr.Web Industrial for Unix File Servers**	
components	.3.1	Status of Dr.Web Industrial for Unix File Servers Components
configd	.3.1.1	drweb-configd component data
configdState	.3.1.1.1	Component status (integer****)
configdExitCode	.3.1.1.2	Last exit code (an integer corresponding to codes from the error catalog)
configdExitTime	.3.1.1.3	Termination time (*Unix time*)
configdInstalledApps	.3.1.1.101	List of installed components
configdAppEntry	.3.1.1.101.1	Information about the installed component (table row; record)
configdAppIndex	.3.1.1.101.1.1	Index (number) of the installed component (integer)

Parameter name	OID parameter	Type and description of the parameter
configdAppName	.3.1.1.101.1.2	Name of the installed component (string)
configdAppExePath	.3.1.1.101.1.3	Component executable path (string)
configdAppInstallTime	.3.1.1.101.1.4	Component installation time (*Unix time*)
configdAppIniSection	.3.1.1.101.1.5	Name of the section with the component parameters in the configuration file (string)
scanEngine	.3.1.2	`drweb-se` component data
scanEngineState	.3.1.2.1	Component status (integer****)
scanEngineExitCode	.3.1.2.2	Last exit code (an integer corresponding to codes from the error catalog)
scanEngineExitTime	.3.1.2.3	Termination time (*Unix time*)
scanEngineStatus	.3.1.2.101	Dr.Web Virus-Finding Engine status (integer)
scanEngineVersion	.3.1.2.102	Dr.Web Virus-Finding Engine version (string)
scanEngineVirusRecords	.3.1.2.103	Number of threat records (integer)
scanEngineMaxForks	.3.1.2.104	Maximum number of child processes for scanning (integer)
scanEngineQueues	.3.1.2.105	Scan task queues
scanEngineQueuesLow	.3.1.2.105.1	The queue of low-priority tasks
scanEngineQueueLowOut	.3.1.2.105.1.1	Number of low-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueLowSize	.3.1.2.105.1.2	Number of low-priority tasks in the queue waiting to be processed (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
scanEngineQueuesMedium	.3.1.2.105.2	The queue of normal-priority tasks
scanEngineQueueMediumOut	.3.1.2.105.2.1	Number of normal-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueMediumSize	.3.1.2.105.2.2	Number of normal-priority tasks in the queue waiting to be processed (counter; integer)
scanEngineQueuesHigh	.3.1.2.105.3	Queue of high-priority tasks
scanEngineQueueHighOut	.3.1.2.105.3.1	Number of high-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueHighSize	.3.1.2.105.3.2	Number of high-priority tasks in the queue waiting to be processed (counter; integer)
scanEngineVirusBasesTable	.3.1.2.106	The list of virus databases
scanEngineVirusBasesEntry	.3.1.2.106.1	Information about the virus database (table row; record)
scanEngineVirusBaseIndex	.3.1.2.106.1.1	Index of the virus database (integer)
scanEngineVirusBasePath	.3.1.2.106.1.2	Path to the virus database file (string)
scanEngineVirusBaseRecords	.3.1.2.106.1.3	Number of records in the virus database (integer)
scanEngineVirusBaseVersion	.3.1.2.106.1.4	Version of the virus database (integer)
scanEngineVirusBaseTimestamp	.3.1.2.106.1.5	Timestamp of the virus database (*Unix time*)
scanEngineVirusBaseMD5	.3.1.2.106.1.6	MD5 checksum (string)
scanEngineVirusBaseLoadResult	.3.1.2.106.1.7	Result of downloading the virus database (string)
scanEngineQueuesTab	.3.1.2.107	The list of scan task queues

Parameter name	OID parameter	Type and description of the parameter
scanEngineQueueEntry	.3.1.2.107.1	Information about the queue (table row; record)
scanEngineQueueIndex	.3.1.2.107.1.1	Index (number) of the queue (integer)
scanEngineQueueName	.3.1.2.107.1.2	Name of the queue (string)
scanEngineQueueOut	.3.1.2.107.1.3	Number of high-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueSize	.3.1.2.107.1.4	Number of tasks in the queue waiting to be processed (counter; integer)
fileCheck	.3.1.3	drweb-filecheck component data
fileCheckState	.3.1.3.1	Component status (integer****)
fileCheckExitCode	.3.1.3.2	Last exit code (an integer corresponding to codes from the error catalog)
fileCheckExitTime	.3.1.3.3	Termination time (*Unix time*)
fileCheckScannedFiles	.3.1.3.101	Number of scanned files (counter; integer)
fileCheckScannedBytes	.3.1.3.102	Number of scanned bytes (counter; integer)
fileCheckCacheHitFiles	.3.1.3.103	Number of scan reports retrieved from the cache for scanned files (counter; integer)
fileCheckScanErrors	.3.1.3.104	Number of scanning engine errors (counter; integer)
fileCheckScanStat	.3.1.3.105	List of clients
fileCheckClientEntry	.3.1.3.105.1	Information about a client (table row; record)
fileCheckClientIndex	.3.1.3.105.1.1	Index (number) of the client (integer)

Parameter name	OID parameter	Type and description of the parameter
fileCheckClientName	.3.1.3.105.1.2	Name of the client component (string)
fileCheckClientScannedFiles	.3.1.3.105.1.3	Number of files scanned for this client (counter; integer)
fileCheckClientScannedBytes	.3.1.3.105.1.4	Number of bytes scanned for this client (counter; integer)
fileCheckClientCacheHitFiles	.3.1.3.105.1.5	Number of scan reports retrieved for this client from the cache for scanned files (counter; integer)
fileCheckClientScanErrors	.3.1.3.105.1.6	Number of scanning engine errors for this client (counter; integer)
update	.3.1.4	`drweb-update` component data
updateState	.3.1.4.1	Component status (integer****)
updateExitCode	.3.1.4.2	Last exit code (an integer corresponding to codes from the error catalog)
updateExitTime	.3.1.4.3	Termination time (*Unix time*)
updateBytesSent	.3.1.4.101	Number of sent bytes (counter; integer)
updateBytesReceived	.3.1.4.102	Number of received bytes (counter; integer)
esagent	.3.1.5	`drweb-esagent` component data
esagentState	.3.1.5.1	Component status (integer****)
esagentExitCode	.3.1.5.2	Last exit code (an integer corresponding to codes from the error catalog)
esagentExitTime	.3.1.5.3	Termination time (*Unix time*)
esagentWorkStatus	.3.1.5.101	Component operation mode (integer: 1—standalone mode, 2—connecting, 3—awaiting

Parameter name	OID parameter	Type and description of the parameter
		connection, 4—connection has been approved)
esagentIsConnected	.3.1.5.102	Connected to the server (integer: 0—no, 1—yes)
esagentServer	.3.1.5.103	Address of the centralized protection server in use (string)
netcheck	.3.1.6	drweb-netcheck component data
netcheckState	.3.1.6.1	Component status (integer****)
netcheckExitCode	.3.1.6.2	Last exit code (an integer corresponding to codes from the error catalog)
netcheckExitTime	.3.1.6.3	Termination time (*Unix time*)
netcheckLocalSeForks	.3.1.6.101	The number of scan engine processes available locally (integer)
netcheckRemoteSeForks	.3.1.6.102	Number of scan engine processes available remotely (integer)
netcheckLocalFilesScanned	.3.1.6.103	Number of files that have been scanned locally (counter; integer)
netcheckNetworkFilesScanned	.3.1.6.104	Number of files that have been scanned remotely (counter; integer)
netcheckLocalBytesScanned	.3.1.6.105	Number of bytes that have been scanned locally (counter; integer)
netcheckNetworkBytesScanned	.3.1.6.106	Number of bytes that have been scanned remotely (counter; integer)
netcheckLocalBytesIn	.3.1.6.107	Number of bytes received from local clients (counter; integer)
netcheckLocalBytesOut	.3.1.6.108	Number of bytes sent back to local clients (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
netcheckNetworkBytesIn	.3.1.6.109	Number of bytes received from remote hosts (counter; integer)
netcheckNetworkBytesOut	.3.1.6.110	Number of bytes sent to remote hosts (counter; integer)
netcheckLocalScanErrors	.3.1.6.111	Number of error occurrences in local scan engine processes (counter; integer)
netcheckNetworkScanErrors	.3.1.6.112	Number of error occurrences in remote scan engine processes (counter; integer)
httpd	.3.1.7	`drweb-httpd` component data
httpdState	.3.1.7.1	Component status (integer****)
httpdExitCode	.3.1.7.2	Last exit code (an integer corresponding to codes from the error catalog)
httpdExitTime	.3.1.7.3	Termination time (*Unix time*)
snmpd	.3.1.8	`drweb-snmpd` component data
snmpdState	.3.1.8.1	Component status (integer****)
snmpdExitCode	.3.1.8.2	Last exit code (an integer corresponding to codes from the error catalog)
snmpdExitTime	.3.1.8.3	Termination time (*Unix time*)
statd	.3.1.9	`drweb-statd` component data
statdState	.3.1.9.1	Component status (integer****)
statdExitCode	.3.1.9.2	Last exit code (an integer corresponding to codes from the error catalog)
statdExitTime	.3.1.9.3	Termination time (*Unix time*)
statdConnectionsIn	.3.1.9.101	Number of accepted incoming connections (counter; integer)
statdConnectionsCount	.3.1.9.102	Number of currently established connections (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
smbspider	.3.1.40	drweb-smbspider-daemon component data
smbspiderState	.3.1.40.1	Component status (integer****)
smbspiderExitCode	.3.1.40.2	Last exit code (an integer corresponding to codes from the error catalog)
smbspiderExitTime	.3.1.40.3	Termination time (*Unix time*)
smbspiderConnectionsIn	.3.1.40.101	Total number of established connections (counter; integer)
smbspiderConnectionsCount	.3.1.40.102	Number of currently established connections (counter; integer)
smbspiderShareTable	.3.1.40.103	Statistics on the protected Samba resources
smbspiderShareEntry	.3.1.40.103.1	Information about the protected Samba resource (table row; record)
smbspiderShareIndex	.3.1.40.103.1.1	Index (number) of the protected Samba resource (integer)
smbspiderSharePath	.3.1.40.103.1.2	Path to the protected Samba resource (string)
smbspiderShareConnectionsIn	.3.1.40.103.1.3	Total number of established connections (counter; integer)
smbspiderShareConnectionsCount	.3.1.40.103.1.4	Number of currently established connections (counter; integer)
meshd	.3.1.52	drweb-meshd component data
meshdState	.3.1.52.1	Component status (integer****)
meshdExitCode	.3.1.52.2	Last exit code (an integer corresponding to codes from the error catalog)
meshdExitTime	.3.1.52.3	Termination time (*Unix time*)
linuxspider	.3.1.201	drweb-spider component data (for GNU/Linux)

Parameter name	OID parameter	Type and description of the parameter
linuxspiderState	.3.1.201.1	Component status (integer****)
linuxspiderExitCode	.3.1.201.2	Last exit code (an integer corresponding to codes from the error catalog)
linuxspiderExitTime	.3.1.201.3	Termination time (*Unix time*)
linuxspiderWorkStatus	.3.1.201.101	Component operation mode (integer: 0—not specified, 1—loading, 2—running via fanotify)
ctl	.3.1.300	`drweb-ctl` component data
ctlState	.3.1.300.1	Component status (integer****)
ctlExitCode	.3.1.300.2	Last exit code (an integer corresponding to codes from the error catalog)
ctlExitTime	.3.1.300.3	Termination time (*Unix time*)
license	.3.2	License status
licenseEsMode	.3.2.1	The license has been granted by the centralized protection server (integer: 0—no, 1—yes)
licenseNumber	.3.2.2	License number (integer)
licenseOwner	.3.2.3	License owner (string)
licenseActivated	.3.2.4	License activation date (*Unix time*)
licenseExpires	.3.2.5	License expiration date (*Unix time*)

*) Threat types:

Code	Threat Type
1	Known threat (*known virus*)
2	Suspicious object (*suspicious*)
3	Adware (*adware*)
4	Dialer (*dialer*)

Code	Threat Type
5	Joke (*joke program*)
6	Riskware (*riskware*)
7	Hacktool (*hacktool*)

**) URL categories:

Code	Threat Type
1	Infection source (*infectionSource*)
2	Not recommended (*notRecommended*)
3	Adult content (*adultContent*)
4	Violence (*violence*)
5	Weapons (*weapons*)
6	Gambling (*gambling*)
7	Drugs (*drugs*)
8	Obscene language (*obsceneLanguage*)
9	Chats (*chats*)
10	Terrorism (*terrorism*)
11	Free email (*freeEmail*)
12	Social networks (*socialNetworks*)
13	URL added due to a notice from copyright owner (*ownerNotice*)
14	Online games (*onlineGames*)
15	Anonymizers (*anonymizers*)
16	Cryptocurrency mining pools (*cryptocurrencyMiningPools*)
17	Jobs (*Jobs*)
20	Black list (*blackList*)

***) Codes of Dr.Web components:

Code	Component
1	Dr.Web ConfigD (`drweb-configd`)
2	Dr.Web Scanning Engine (`drweb-se`)
3	Dr.Web File Checker (`drweb-filecheck`)
4	Dr.Web Updater (`drweb-update`)
5	Dr.Web ES Agent (`drweb-esagent`)
6	Dr.Web Network Checker (`drweb-netcheck`)
7	Dr.Web HTTPD (`drweb-httpd`)
8	Dr.Web SNMPD (`drweb-snmpd`)
40	SpIDer Guard for SMB (`drweb-smbspider-daemon`)
52	Dr.Web MeshD (`drweb-meshd`)
201	SpIDer Guard (`drweb-spider`)
300	Dr.Web Ctl (`drweb-ctl`)
400	Scanning tasked by a centralized protection server (not a component of Dr.Web Industrial for Unix File Servers)

****) Possible states of the components:

Code	Status
0	Not installed
1	Installed but not started
2	Is starting
3	Is running
4	Is exiting

To get the values of the variables directly, use the `snmpwalk` utility:

```
$ snmpwalk -Os -c <community> -v <SNMP version> <host address> <OID>
```

For example, to get statistics about the threats detected on the local host, if Dr.Web SNMPD has default settings, run the command:

```
$ snmpwalk -Os -c public -v 2c 127.0.0.1 .1.3.6.1.4.1.29690.2.2.1
```

9.13. Dr.Web MeshD

The Dr.Web MeshD component is an agent that integrates a host on which Dr.Web Industrial for Unix File Servers is installed with a "local cloud", which combines hosts with installed Dr.Web for Unix products. This cloud allows some cloud hosts to access the scanning engine of other cloud hosts, in other words, to receive file scanning services.

To combine hosts with Dr.Web for Unix products installed, the Dr.Web MeshD component, which integrates a host with the cloud, must be installed on every host. Host privileges within the cloud and cloud features used by a host are flexibly adjusted with Dr.Web MeshD settings.

Data is shared with other cloud hosts over a protected SSH channel.

9.13.1. Operating Principles

In this section

- Connection types
- Operation modes
- Services
  - Remote file scanning (Engine)
  - Sending files for scanning (File)
  - URL checking (Url)

Dr.Web MeshD is a mediator that ensures interaction of a host with Dr.Web Industrial for Unix File Servers installed and other cloud hosts.

Connection types

Dr.Web MeshD uses the following connection types:

- *Client (service)*—used by Dr.Web MeshD to connect to other cloud hosts that are clients of services provided by the given host.

  Dr.Web Industrial for Unix File Servers components operating on the host and using services provided by the cloud connect to Dr.Web MeshD, which operates on the same host, through a local Unix socket. At that, a client connection is not used.

- *Partner (peer to peer)*—used by Dr.Web MeshD for interaction with peer (within a service) partner cloud hosts. Usually such horizontal connections are used for scaling and distributing the load when interacting with the cloud, as well as for synchronization of cloud hosts.
- *Uplink*—used by Dr.Web MeshD for connecting this host as a client to cloud hosts that provide services (for example, sending files for scanning and so on).

The use of all three types of connections is configured for different cloud services independently from each other. At that, the same host can be configured as a server for processing client requests within one service (for example, checking URLs) and as a client within another service (for example, remote file scanning).

Within cloud, hosts perform authorized interaction via SSH, that is, all sides of interhost communication are always mutually authenticated. For the authentication, *host keys* are used in compliance with RFC 4251. A client connection from a local component is always considered as trusted.

Operation modes

Dr.Web MeshD can either operate in daemon mode or run at the request of other Dr.Web Industrial for Unix File Servers components installed on the local host. If Dr.Web MeshD is configured to serve client connections (the `ListenAddress` parameter is not empty) and at least one of the services is activated, Dr.Web MeshD starts as a daemon and awaits client connections.

If Dr.Web MeshD is not set to process client connections (the `ListenAddress` parameter is empty) and there are no requests to this component during a time interval specified by the `IdleTimeLimit` parameter, the component shuts down automatically.

Services

Remote file scanning (Engine)

This service allows to use Dr.Web Scanning Engine for scanning remote files: hosts acting as clients send files for scanning to a server host, and server hosts provide a service for scanning files sent by the client hosts. Typical client settings are as follows:

```
…
[MeshD]
EngineChannel = On
EngineUplink = <server address>
ListenAddress =
…
```

The following settings are specified on the host acting as a local scanning server:

```
EngineChannel = On
EngineUplink =
ListenAddress = <address>:<port>
```

Here, <*server address*> in the uplink connection of the client must refer to the <*address*> and <*port*> that are used by the server host for managing client connections.

Sending files for scanning (File)

This feature is not used (remote scanning is provided by the *Engine* service).

URL checking (Url)

This service allows to check whether a URL belongs to potentially dangerous and unwanted categories: client hosts send a URL to be checked to a server host. Typical client settings are as follows:

```
…
[MeshD]
UrlChannel = On
UrlUplink = <server address>
ListenAddress =
…
```

The following settings are specified on the host acting as a URL checking server:

```
UrlChannel = On
UrlUplink =
ListenAddress = <address>:<port>
```

Here, *<server address>* in the uplink connection of the client must refer to the *<address>* and *<port>* that are used by the server host for managing client connections.

9.13.2. Command-Line Arguments

To start the Dr.Web MeshD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-meshd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-meshd [<parameters>]
```

Dr.Web MeshD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v`

	Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-meshd --help
```

This command outputs short help information about the Dr.Web MeshD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-meshd` command.

9.13.3. Configuration Parameters

The component uses configuration parameters specified in the `[MeshD]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-meshd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-meshd`

Parameter	Description
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`DebugSsh` *{boolean}*	Log or do not log SSH messages (messages and data are transmitted via SSH) received and sent by the Dr.Web MeshD component operating on the host, if the logging level is `LogLevel = Debug`. Default value: `No`
`ListenAddress` *<IP address>:<port>*	Network socket (address and port) of the client connection where the component is waiting to receive connections from cloud hosts. These hosts are clients of services provided by this cloud host. In order for the component to listen on the IPv6 interface and detect cloud client hosts via IPv6, this parameter must be defined. If the value is not specified, the component does not receive requests from clients. Default value: `0.0.0.0:7090` if the `drweb-scanning-server` package is installed, not specified otherwise
`DnsResolverConfPath` *{path to file}*	Path to a DNS resolver configuration file. Default value: `/etc/resolv.conf`
`DiscoveryResponderPort` *{port number}*	Port on which a higher-level MeshD host responds to client requests via UDP. The *discovery* mechanism is enabled only if the `ListenAddress` parameter is defined. Default value: `18008`
`FileChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to participate in file exchange services. If the parameter is set to `On`, the Dr.Web MeshD component is automatically started by the Dr.Web ConfigD configuration daemon. Default value: `On`
`FileUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD receiving files from this host for scanning. Allowed values: • *value is not specified*—server is not specified for this service, and Dr.Web MeshD will not establish connections.

Parameter	Description
	• *<IP address>:<port>*—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:<service name>[:<domain>]`—address and port of the server are determined by searching for the SRV record of the DNS domain *<domain>*. If *<domain>* is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`FileDebugIpc` *{boolean}*	Log or do not log the debug information for the file exchange service if the logging level is `LogLevel = Debug`. Default value: `No`
`EngineChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to provide scanning engine services. If the parameter is set to `On`, the Dr.Web MeshD component is automatically started by the Dr.Web ConfigD configuration daemon. Default value: `On`
`EngineUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD providing scanning engine services for this host. Allowed values: • *value is not specified*—server is not specified for this service, and Dr.Web MeshD will not establish connections. • *<IP address>:<port>*—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:<service name>[:<domain>]`—address and port of the server are determined by searching for the SRV record of the DNS domain *<domain>*. If *<domain>* is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`EngineDebugIpc` *{boolean}*	Log or do not log the debug information for the file exchange service if the logging level is `LogLevel = Debug`. Default value: `No`

Parameter	Description
`UrlChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to provide URL checking services. Default value: `On`
`UrlUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD providing URL checking services for this host. Allowed values: • *value is not specified*—URL checking server is not specified. • <*IP address*>:<*port*>—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:`<*service name*>`[:`<*domain*>`]`—address and port of the server are determined by searching for the SRV record of the DNS domain <*domain*>. If <*domain*> is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`UrlDebugIpc` *{boolean}*	Log or do not log the debug information for the URL checking service if the logging level is `LogLevel = Debug`. Default value: `No`



The current version of Dr.Web Industrial for Unix File Servers does not use the *File* service for sending files for scanning. Use the *Engine* service of the scanning engine instead.

9.14. Dr.Web StatD

The Dr.Web StatD component is designed for accumulating statistics on events that occur during the operation of the Dr.Web Industrial for Unix File Servers components. The events are registered in permanent storage and can be obtained on request.

9.14.1. Operating Principles

The Dr.Web StatD component ensures accumulation and permanent storage of events obtained during the operation of Dr.Web Industrial for Unix File Servers components. The events related to unexpected component shutdown are logged.

To view and manage events, use the `events` command of the Dr.Web Ctl utility.

9.14.2. Command-Line Arguments

To start the Dr.Web StatD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-statd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-statd [<parameters>]
```

Dr.Web StatD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-statd --help
```

This command outputs short help information about the Dr.Web StatD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Industrial for Unix File Servers from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-statd` command.

9.14.3. Configuration Parameters

The component uses configuration parameters specified in the `[StatD]` section of the unified configuration file of Dr.Web Industrial for Unix File Servers.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-statd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-statd`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`

Parameter	Description
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`MaxEventStoreSize` *{size}*	Maximum allowed size of the event database. Defined in `mb`, for example: `MaxEventStoreSize = 100mb`. Minimal value: `50mb`. Default value: `1GB`

10. Appendices

10.1. Appendix A. Types of Computer Threats

In this section

- Computer viruses
- Computer worms
- Trojans
- Hacktools
- Adware
- Jokes
- Dialers
- Riskware
- Suspicious objects

Herein, the term *"threat"* is defined as any kind of software potentially or directly capable of inflicting damage to a computer or network and compromising user information or rights (that is, malicious and other unwanted software). Broadly speaking, the term "threat" can be used to indicate any type of potential danger to a computer or network (that is, a vulnerability, which can be used in cyberattacks).

All of the program types stated below are capable of endangering user data or confidentiality. Programs that do not conceal their presence in the system (e.g. spam distribution software or traffic analyzers) are usually not considered computer threats, although they can also harm the user under certain circumstances.

Computer viruses

Computer threats of this type are capable of embedding their code in other programs. Such embedding is called *infection*. In most cases, an infected file becomes a virus carrier and the embedded code does not necessarily match the original one. A significant number of viruses is designed to damage or destroy data.

In Doctor Web classification, viruses are separated by the type of objects they infect:

- *File viruses* infect files of the operating system (usually executable files and dynamic libraries) and are activated when an infected file is started.
- *Macro-viruses* infect documents used by Microsoft® Office (and other applications supporting macros, for example, written in Visual Basic). *Macros* are embedded programs written in a fully functional programming language and can be run in specific conditions (for instance, in Microsoft® Word, macros can be automatically started upon opening, closing or saving a document).

- *Script viruses* are created using script languages and usually infect other script files (e.g. service files of an operating system). They can also infect files of other types that allow execution of scripts and can spread, for example, via vulnerable web applications.
- *Boot viruses* infect boot records of disks and partitions as well as master boot records of hard drives. They do not require much memory and remain ready to continue performing their tasks until the system is unloaded, restarted or shut down.

Most viruses employ certain mechanisms of protection against detection. They are constantly being improved, and ways to cope with them are constantly being developed. All viruses can also be classified according to their type of protection against detection:

- *Encrypted viruses* encrypt their code upon every infection to hinder their detection in a file, a boot sector or memory. All instances of such viruses contain only a short common code fragment (the decryption procedure) that can be used as a signature.
- *Polymorphic viruses* not only encrypt their code, but they also generate a special decryption procedure that is different in every instance of the virus. As a result, such viruses do not have byte signatures.
- *Stealth viruses* (invisible viruses) perform certain actions to disguise their activity and to conceal their presence in infected objects. Such viruses gather the characteristics of an object before infecting it and then pass old data when the operating system or a program scans for modified files.

Viruses can also be classified according to a programming language in which they are written (for example, a low-level language such as an assembly language or a high-level language such as Go) or according to infected operating systems.

Computer worms

Like viruses, programs of the *"computer worm"* type can copy themselves, but they do not infect other objects. A worm gets into a computer from a network (typically as an email attachment or from the internet) and sends its functioning copies to other computers. Worms either rely on user actions or spread automatically.

Worms do not necessarily consist of only one file (the worm body). Many of them have a so called infectious part (the shellcode), which loads into the computer operating memory and then downloads the worm body as an executable file over the network. Until the worm body is downloaded to the system, the worm can be avoided by rebooting the computer (at which the operating memory is reset). However, if the worm body infiltrates the system, then only an anti-virus program can cope with it.

Worms are capable of rendering entire networks inoperable even if these worms do not carry any payload (i.e. do not cause any direct damage to the system).

In Doctor Web classification, worms are separated by their distribution method (environment):

- *Network worms* spread via various network and file sharing protocols.
- *Mail worms* spread via email protocols (POP3, SMTP and so on).

- *Chat worms* spread via popular instant messaging services (ICQ, IM, IRC and so on).

Trojans

Malware of this type cannot reproduce itself. A trojan pretends to be a popular program and performs its functions (or imitates its operation). Meanwhile, it performs some malicious actions (damages or deletes data, sends confidential information and so on) or makes it possible for a hacker to access the computer without permission, for example, in order to harm a third party.

Trojan masking and malicious features are similar to those of a virus. A trojan can even be a component of a virus. However, most trojans spread as separate executable files (through file exchange servers, removable data carriers or email attachments) that are started by users or certain system processes.

It is very hard to classify trojans due to the fact that they are often spread by viruses and worms and also because many malicious actions that can be performed by other types of threats are attributed to trojans only. Here are some trojan types that are classified by Doctor Web as separate classes:

- *Backdoors* are trojans that allow to gain privileged access to a system, bypassing existing access and security measures. Backdoors do not infect files.
- *Rootkits* are used to intercept system functions of an operating system in order to conceal themselves. Furthermore, a rootkit can hide processes of other programs, directories and files. It can spread either as an independent program or as an auxiliary component of another malicious program. There are two kinds of rootkits according to their operation mode: *User Mode Rootkits—UMR* (intercept functions of user mode libraries) and *Kernel Mode Rootkits—KMR* (intercept functions at the system kernel level, which makes them harder to detect and neutralize).
- *Keyloggers* are used to log data that users enter by means of a keyboard. The aim of this is to steal personal information (i.e. network passwords, logins, primary account numbers and so on).
- *Clickers* redefine hyperlinks when they are clicked and thus redirect users to certain websites (sometimes malicious). This is usually done to increase ad traffic of websites or perform distributed denial-of-service (DDoS) attacks.
- *Proxy trojans* provide anonymous internet access to a malicious actor through a victim's computer.

In addition, trojans can also change the start page in a web browser or delete certain files. However, these actions can also be performed by other types of threats (viruses and worms).

Hacktools

Hacktools are programs designed to assist an intruder with hacking. The most common among them are port scanners that detect vulnerabilities in firewalls and other components of a

computer protection system. Besides hackers, such tools are used by administrators to test security of their networks. Occasionally, programs that use social engineering techniques are classified as hacktools.

Adware

Usually, this term refers to program code embedded in freeware programs that forces displaying of advertisements to a user. However, sometimes such code can be distributed via other malicious programs and display advertisements, for example, in web browsers. Many adware programs operate with data collected by spyware.

Jokes

Like adware, this type of malware cannot be used to inflict any direct damage to the system. Joke programs usually just generate messages about errors that never occurred and threaten to perform actions that will lead to data loss. Their purpose is to frighten or annoy users.

Dialers

These are special programs that are designed to scan a range of telephone numbers and find those where a modem answers. These numbers are then used to mark up the price of telephoning facilities or to connect the user to expensive telephone services.

Riskware

These programs were not created for malicious purposes, but due to their characteristics they can pose a threat to the system security. Riskware can accidentally damage or delete data or be used by malicious actors or other programs to harm the system. Riskware includes various remote chat and administrative tools, FTP servers and so on.

Suspicious objects

Suspicious objects include any potential threats detected by a heuristic analyzer. Such objects can potentially relate to any type of computer threats (even unknown to information security specialists) or appear to be safe in case of false positives. It is recommended that files containing suspicious objects are quarantined and sent to Doctor Web anti-virus laboratory experts for analysis.

10.2. Appendix B. Neutralizing Computer Threats

In this section

- Detection methods
  - Signature analysis
  - Origins Tracing™
  - Execution emulation
  - Heuristic analysis
- Threat-related actions

All Doctor Web anti-virus solutions use a set of methods to detect threats, which allows to scan suspicious objects thoroughly.

Detection methods

Signature analysis

This method of detection is the first to run. It is applied by scanning object contents for known threat signatures. A signature is a continuous finite sequence of bytes necessary and sufficient to identify a threat unequivocally. At that, the search for signatures of the objects being scanned is performed using checksums, which allows to reduce virus databases significantly in size having preserved, at the same time, unequivocal matching and correct detection of threats and curing infected objects. Dr.Web virus databases are composed such that the same record can cover entire classes or families of threats.

Origins Tracing™

This unique Dr.Web technology allows to detect new and modified threats using already known infecting and damaging techniques covered by virus databases. The technology is used after completion of the signature analysis and protects users using Dr.Web anti-virus solutions against such threats as the notorious Trojan.Encoder.18 ransomware (also known as gpcode). Furthermore, using the Origins Tracing™ technology allows to considerably reduce the number of false positives of the heuristic analyzer. Objects detected using Origins Tracing™ have the `.Origin` postfix added to their names.

Execution Emulation

The technology of program code emulation is used for detection of polymorphic and encrypted viruses when a search by checksums cannot be performed directly, or is considerably complicated (for example, it is impossible to generate reliable signatures). The method consists in emulating the execution of an analyzed code with an *emulator*—a programming model of a

processor and runtime environment. An emulator operates with a protected memory area (an *emulation buffer*). At that, instructions are not passed to a CPU for actual execution. If the code processed by the emulator is infected, the emulation results in restoring the original malicious code available for signature analysis.

Heuristic analysis

The operation of the heuristic analyzer is based on a set of *heuristics*—assumptions about fingerprints of both malware and safe code, which statistical significance is proved experimentally. Each fingerprint has a certain *weight* (that is, a number which determines a level of its severity and reliability). The weight can be positive if the fingerprint is indicative of malicious behavior or negative if the fingerprint is non-typical for computer threats. Depending on the total weight of contents of an object, the heuristic analyzer calculates a probability of this object containing unknown malware. If a threshold is exceeded, the heuristic analyzer returns a verdict that the analyzed object is malicious.

The heuristic analyzer also uses the FLY-CODE™ technology, which is a versatile algorithm to unpack files. This technology allows to make heuristic assumptions about the presence of malware in objects packed not only by those packers that Dr.Web developers are aware of, but also by new, previously unknown programs. While scanning packed objects, their structural entropy is being analyzed, thereby allowing to detect threats on the basis of the allocation of their code segments. This technology allows to detect multiple different threats packed by the same polymorphous packer on the basis of a single record in a virus database.

As any system of hypothesis testing under uncertainty, the heuristic analyzer can make type I or type II errors (skipping unknown threats or raising false positives correspondingly). Thus, objects detected by the heuristic analyzer are treated as “suspicious”.

While performing any of the scans, Dr.Web anti-virus solutions use the most recent information about all known malware. Threat signatures, footprints and behavioral patterns are updated and added to virus databases as soon as experts of the Doctor Web anti-virus laboratory discover new threats, occasionally several times per hour. Even if the newest malicious program passes Dr.Web real-time protection and penetrates the system, this program will be detected in the list of processes and neutralized after updating virus databases.

Threat-related actions

Dr.Web products implement a number of actions that can be applied to detected objects to neutralize computer threats. The user can keep default actions applied to specific types of threats automatically, adjust these actions or choose the required action manually each time upon detection. Available actions are provided below. The brackets contain a parameter denoting a corresponding action and used in the unified configuration file and commands of the Dr.Web Ctl tool.

- **Report** (`REPORT`)—report the threat without applying any action to the infected object.
- **Ignore** (`PASS`)—skip the detected threat without applying any action.

- **Block** (`BLOCK`)—block all attempts to access the infected file (the action can be unavailable for some components).
- **Cure** (`CURE`)—attempt to cure the infected object by removing only malicious content from its body.



This action can be applied only to certain types of threats.

- **Quarantine** (`QUARANTINE`)—put the infected object (if possible) in a specialized quarantine directory to isolate it.
- **Delete** (`DELETE`)—permanently delete the infected object.



If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined and not deleted.

10.3. Appendix C. Technical Support

If you have a problem installing or using Doctor Web products, please try the following before contacting technical support:

1. Download and review the latest manuals and guides.
2. See the Frequently Asked Questions section.
3. Browse the official Doctor Web forum.

If you haven't found a solution to your problem, you can fill out a web form in the appropriate section of the official website to request direct assistance from Doctor Web technical support specialists.

For information on regional and international offices of Doctor Web, please visit the official website.

To facilitate processing of your issue, we recommend that you generate a data set for the installed product, its configuration, and system environment before contacting our technical support. To do that, you can use a special utility included in Dr.Web Industrial for Unix File Servers.

To collect data for our technical support, run the command:

- for GNU/Linux:

```
# /opt/drweb.com/bin/support-report.sh <path to file>
```

- for FreeBSD:

```
# /usr/local/libexec/drweb.com/bin/support-report.sh <path to file>
```

where <*path to file*>—path to an archive in the `.tgz` format to which the debugging information about the product and the system environment will be written, for example, `/tmp/report.tgz`. Already existing files will not be rewritten. If the path is not specified, the archive will be stored in the directory of the superuser who started the utility (for example, `/root`) and will be named as follows:

```
drweb.report.<timestamp>.tgz
```

where <*timestamp*> is a full timestamp of creating the report, down to milliseconds, for example: 20190618151718.23625.



The utility for collecting data for our technical support must be run with superuser (*root*) privileges. To get superuser privileges, use the `su` command to change the user or the `sudo` command to run the command as another user.

During operation, the utility collects and archives the following information:

- information about your OS (name, architecture, output of the `uname -a` command);
- list of packages installed on your system, including Doctor Web packages;
- log contents:
  - Dr.Web Industrial for Unix File Servers logs (if configured for separate components);
  - log of the `syslog` system daemon (`/var/log/syslog`, `/var/log/messages`);
  - log of a system package manager (`apt`, `yum`, etc.);
  - `dmesg` log;
- output of the commands `df`, `ip a` (`ifconfig -a`), `ldconfig -p`, `iptables-save`, `nft export xml`.
- information about settings and configuration of Dr.Web Industrial for Unix File Servers:
  - list of downloaded virus databases (`drweb-ctl baseinfo -l`);
  - list of files from Dr.Web Industrial for Unix File Servers directories and MD5 hash values of these files;
  - Dr.Web Virus-Finding Engine version and MD5 hash value;
  - information about the user and permissions retrieved from the key file, if Dr.Web Industrial for Unix File Servers is not running in centralized protection mode.

10.4. Appendix D. Dr.Web Industrial for Unix File Servers Configuration File

Configuration parameters of all Dr.Web Industrial for Unix File Servers components are managed by the Dr.Web ConfigD configuration management daemon. Changed parameters are stored in the `drweb.ini` file whose default directory is `/etc/opt/drweb.com/` for GNU/Linux or `/usr/local/etc/drweb.com/` for FreeBSD.



The text configuration file stores only those parameters whose values differ from defaults. If a parameter is absent in the configuration file, its default value is used.

View the list of all available parameters, including those that are absent in the configuration file and have default values, by running the command:

```
$ drweb-ctl cfshow
```

You can change any parameter value in two ways:

- By running the command:

```
# drweb-ctl cfset <section>.<parameter> <new value>
```



This command requires to run the Dr.Web Ctl management tool with superuser (the *root* user) privileges. To obtain superuser privileges, use the `su` or `sudo` command.

For further information about the syntax of the `cfshow` and `cfset` commands of the Dr.Web Ctl tool (the `drweb-ctl` module), refer to the Dr.Web Ctl section.

- By specifying this parameter in the configuration file (by editing the file in any text editor) and reloading the configuration of Dr.Web Industrial for Unix File Servers to apply changes by running the command:

```
# drweb-ctl reload
```

10.4.1. File Structure

The configuration file complies with the rules indicated below.

- The file content is separated into named sections. Allowed section names are strictly set and cannot be arbitrary. A section name is specified in square brackets and is identical to the name of the Dr.Web Industrial for Unix File Servers component that uses parameters of this section (except for the `[Root]` section, which stores parameters of the Dr.Web ConfigD configuration daemon).
- The `;` or `#` characters in the beginning of the lines of the configuration file indicate a comment. Such lines are skipped by Dr.Web Industrial for Unix File Servers components while reading configuration parameters from the configuration file.

- Each line in the file can contain only one parameter:

  ```
  <parameter> = <value>
  ```

- All parameter names are strictly set and cannot be arbitrary.
- All section and parameter names are case-insensitive. Parameter values, except for names of directories and files in paths (for Unix-like OSes), are also case-insensitive.
- Parameter values in the configuration file must be enclosed in quotation marks if these values contain spaces.
- Some parameters can accept multiple values. If so, the values are either comma-separated or specified several times in different lines of the configuration file. In the former case, spaces around a comma are ignored. If a space character is a part of a parameter value, the entire value must be enclosed in quotation marks.

  You can specify multiple values as:

  - a comma-separated list:

    ```
    Parameter = "Value1", "Value2","Value 3"
    ```

  - a sequence of lines in the configuration file:

    ```
    Parameter = Value2
    Parameter = Value1
    Parameter = "Value 3"
    ```

  The order of values is unimportant.

  Paths to files are always enclosed in quotation marks when separated by commas, for example:

  ```
  ExcludedPaths = "/etc/file1", "/etc/file2"
  ```

For the description of the sections of the configuration file, see the description of the Dr.Web Industrial for Unix File Servers components using it.

10.4.2. Parameter Types

Configuration parameters can be of the following types:

1. *Address*—network connection address specified as *<IPv4 address>:<port>* or *<IPv6 address>:<port>*. Use square brackets to set an IPv6 address value. In some cases the port can be omitted (in each case this is indicated in the description of the parameter).
2. *Boolean*—flag parameter accepting `On`, `Yes`, `True` (the parameter is enabled) and `Off`, `No`, `False` (the parameter is disabled).
3. *Integer*—non-negative integer.
4. *Fractional number*—non-negative number with a fractional part can be indicated as a parameter value.

5. *Time interval*—time interval consisting of a non-negative integer and a suffix character indicating the specified unit of measurement. The following suffixes representing units of measurement can be used:
   - `w`—weeks (`1w = 7d`);
   - `d`—days (`1d = 24h`);
   - `h`—hours (`1h = 60m`);
   - `m`—minutes (`1m = 60s`);
   - `s` or no suffix—seconds.

   If the time interval is specified in seconds, you can specify milliseconds after a point (no more than three digits, for example, `0.5s`—500 milliseconds). It is possible to specify multiple time intervals in different time units as a single time interval; in this case, it is counted as a sum of intervals (in fact, the configuration parameters always store a number of milliseconds covered by this time interval).

   In general terms, any time interval can be represented by the following expression: N_1wN_2dN_3hN_4mN_5[.N_6]s, where N_1, ..., N_6 is a number of the corresponding time unites included in this interval. For example, a year (365 days) can be represented as follows (all records are equivalent): `365d`, `52w1d`, `52w24h`, `51w7d24h`, `51w7d23h60m`, `8760h`, `525600m`, `31536000s`.

   The examples below show you how intervals of 30 minutes, 2 seconds, 500 milliseconds can be specified:

   - in the configuration file:

   ```
   UpdateInterval = 30m2.5s
   ```

   - by running the `drweb-ctl cfset` command:

   ```
   # drweb-ctl cfset Update.UpdateInterval 1802.5s
   ```

   - via a command-line parameter (for example, for the scanning engine):
     - for GNU/Linux:

     ```
     # /opt/drweb.com/bin/drweb-se <socket> --WatchdogInterval 1802.5
     ```

     - for FreeBSD:

     ```
     # /usr/local/libexec/drweb.com/bin/drweb-se <socket> --WatchdogInterval 1802.5
     ```

6. *Size*—the parameter value represents a size of an object (a file, a buffer, a cache, and so on), specified as a non-negative integer and a suffix representing a measurement unit. The following suffixes that specify size units can be used:
   - `GB`—gigabytes (`1GB = 1024MB`);
   - `MB`—megabytes (`1MB = 1024KB`);
   - `KB`—kilobytes (`1KB = 1024B`);
   - `B`—bytes.

If a suffix is omitted, the size is considered to be in bytes. A single size record can be specified by multiple sizes in different units; in this case, the resulting size is counted as their sum (in fact, the configuration parameters always store the size in bytes).

7. *Path to directory (file)*—the parameter value is a string representing an acceptable path to a directory (a file).



The file path must be ended with a file name.



On Unix-like operating systems, names of directories and files are case-sensitive. If it is not explicitly mentioned in a parameter description, paths cannot be represented by masks with special characters (`?`, `*`).

8. *Logging level*—level at which the Dr.Web Industrial for Unix File Servers component events are logged. The following values are allowed:
   - `Debug`—the most detailed logging level. All messages and debug information are logged.
   - `Info`—all messages are logged.
   - `Notice`—all error messages, warnings and notifications are logged.
   - `Warning`—all error messages and warnings are logged.
   - `Error`—only error messages are logged.
9. *Log type*—the parameter value specifies the Dr.Web Industrial for Unix File Servers component logging method. The following values are allowed:
   - `Stderr[:ShowTimestamp]`—messages are output to *stderr* (standard error stream). This value can be used *only* in the settings of the configuration daemon. At that, if it works in the background ("*daemonized*"), i.e. it is started with the `-d` parameter, this value *cannot* be used because components operating in the background cannot access input/output streams of the terminal). The additional `ShowTimestamp` parameter instructs to add a timestamp to every message.
   - `Auto`—messages for logging are passed to the Dr.Web ConfigD configuration daemon, which stores them in a single location specified in its own settings (the `Log` parameter in the `[Root]` section). This value is specified for all components *except for the configuration daemon* and is used as a default value.
   - `Syslog[:`<*facility*>`]`—the component will pass messages to the `syslog` system logging service.
   - The additional <*facility*> option is used to specify a type of a log to store messages logged by `syslog`. Allowed values:
     - `Daemon`—messages from daemons,
     - `User`—messages from user processes,
     - `Mail`—messages from mail programs,
     - `Local0`—messages from local processes 0,

    - ...
    - `Local7`—messages from local processes 7.
  - *<path to file>*—messages will be stored by the component directly in the specified log.

  Examples of how to specify a parameter value:

  - in the configuration file:

```
Log = Stderr:ShowTimestamp
```

  - by running the `drweb-ctl cfset` command:

```
# drweb-ctl cfset Root.Log /tmp/log
```

  - via a command-line parameter (for example, for the scanning engine):
    - for OSes of the GNU/Linux family:

```
# /opt/drweb.com/bin/drweb-se <socket> --Log Syslog:DAEMON
```

    - for FreeBSD:

```
# /usr/local/libexec/drweb.com/bin/drweb-se <socket> --Log Syslog:DAEMON
```

10. *Action*—action to be applied by the Dr.Web Industrial for Unix File Servers component upon detection of certain threats or upon another event. Allowed values:
    - **Report** (`REPORT`)—only notify of threat detection without applying other actions;
    - **Block** (`BLOCK`)—block all attempts to access the file without modifying it (the action might not be available for all components);
    - **Cure** (`CURE`)—attempt to cure the threat (that is, remove only malicious content from the file body);
    - **Quarantine** (`QUARANTINE`)—put the infected file in quarantine;
    - **Delete** (`DELETE`)—delete the infected file.

    Some of the actions can be applied only upon certain events (for example, a “Scanning error” event cannot trigger the `CURE` action). Allowed actions are always listed in the description of each parameter of the *action* type.

Other types of parameters and their allowed values are explicitly specified in the description of these parameters.

10.5. Appendix E. Generating SSL Certificates

For the Dr.Web Industrial for Unix File Servers components that use a secure SSL/TLS data channel and application protocols, such as HTTPS, LDAPS, SMTPS and so on, to exchange data, it is necessary to provide private SSL keys and the corresponding certificates. Keys and certificates for some components are generated automatically; as for the others, they should be provided by a Dr.Web Industrial for Unix File Servers user. All the components use certificates in the PEM format.

To generate private keys and certificates used for connections via SSL/TLS, including verification certificates of Certification Authority (CA) and signed certificates, you can use the `openssl` command-line utility (included in the OpenSSL cryptographic package).

Consider a sequence of actions required for generating a private key and the corresponding SSL certificate together with an SSL certificate signed with a CA verification certificate.

To generate a private SSL key and a certificate

1. To generate a private key (the RSA algorithm, the key length is 2048 bits), run the command:

   ```
   $ openssl genrsa -out keyfile.key 2048
   ```

   If you want to password protect the key, use the `-des3` option. The generated key is in the `keyfile.key` file located in the current directory.

   To view the generated key, run the command:

   ```
   $ openssl rsa -noout -text -in keyfile.key
   ```

2. To generate a certificate for a specified time period based on the existing private key (in this case, for 365 days), run the command:

   ```
   $ openssl req -new -x509 -days 365 -key keyfile.key -out certificate.crt
   ```

   This command will request data (a name, an organization and so on) that identify the certified object. The generated certificate will be located in the `certificate.crt` file.

   To scan the contents of the generated certificate, run the command:

   ```
   $ openssl x509 -noout -text -in certificate.crt
   ```

To register a certificate as a trusted CA certificate

1. Move or copy the certificate file to the system trusted certificate directory (`/etc/ssl/certs` on Debian or Ubuntu).
2. In the trusted certificate directory, create a symbolic link to the certificate, where the name of the link is the hash value of the certificate.
3. Reindex the contents of the system directory containing certificates.

The example commands provided below perform all these three actions. It is assumed that the current directory is the trusted certificate directory `/etc/ssl/certs` and the certificate that is registered as a trusted one is located in the `/home/user/ca.crt` file:

```
# cp /home/user/ca.crt .
# ln -s ca.crt `openssl x509 -hash -noout -in ca.crt`.0
# c_rehash /etc/ssl/certs
```

To create a signed certificate

1. Generate a request file for signing a certificate *(Certificate Signing Request—CSR)* based on an existing private key. If the key is absent, generate it.

   The request for signing is created by running the command:

   ```
   $ openssl req -new -key keyfile.key -out request.csr
   ```

   This command, as well as the command for certificate creation, requests data that identifies the certified object. Here, `keyfile.key` is the existing file of the private key. The received request will be saved to the `request.csr` file.

   To check the result of request creation, run the command:

   ```
   $ openssl req -noout -text -in request.csr
   ```

2. To create a signed certificated based on the request and the existing CA certificate, run the command:

   ```
   $ openssl x509 -req -days 365 -CA ca.crt -CAkey ca.key -set_serial 01 -in request.csr
   -out sigcert.crt
   ```

   To create a signed certificate, you must have the following three files: the file of the root certificate `ca.crt` and its private key `ca.key` (the `certificate.crt` certificate and the `keyfile.key` key may be used istead of `ca.crt` and `ca.key`, then the obtained certificate will be self-signed), as well as the request for signing `request.csr`. The created signed certificate will be saved to the file `sigcert.crt`.

   To check the result, run the command:

   ```
   $ openssl x509 -noout -text -in sigcert.crt
   ```

Repeat the procedure of creating a key and a certificate (or a signed certificate, if necessary) as many times as the number of unique certificates you need to create. For example, from a security point of view, every agent for distributed file scanning by Dr.Web Network Checker within a scanning cluster should have its own key/certificate pair.

To convert a signed certificate

Some browsers or mail clients may require to convert the signed certificate used for authorization to the `PKCS12` format.

Perform such conversion by running the command:

```
# openssl pkcs12 -export -in sigcert.crt -out sigcert.pfx -inkey keyfile.key
```

Here, `sigcert.crt` is the existing file of the signed certificate, `keyfile.key` is the file of the corresponding private key. The resulting converted certificate is saved to the `sigcert.pfx` file.

10.6. Appendix F. Known Errors

In this section

- Recommendations for error identification
- Error codes
- Errors without a code

If the occurred error is not present in this section, it is recommended that you contact our technical support. Be ready to provide the error code and describe steps to reproduce the issue.

Recommendations for error identification

- To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.
- To identify errors, we recommend that you store output of the log in a separate file and enable output of detailed debug information. For that, run the commands:

```
# drweb-ctl cfset Root.Log <log path>
# drweb-ctl cfset Root.DefaultLogLevel DEBUG
```

- To reset the logging settings to defaults, run the commands:

```
# drweb-ctl cfset Root.Log -r
# drweb-ctl cfset Root.DefaultLogLevel -r
```

Error codes

Error message: *Error on monitor channel*

Error code: `x1`

Internally used name: `EC_MONITOR_IPC_ERROR`

Description: One or several components cannot connect to the Dr.Web ConfigD configuration daemon.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Restart the configuration daemon:

   ```
   # service drweb-configd restart
   ```

2. Check whether the authentication mechanism for PAM is installed, configured and operates correctly. If necessary, install and configure it (for details refer to administration manuals for your OS

distribution).

3. If PAM is configured correctly and restarting the configuration daemon does not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

4. If it is not possible to start the configuration daemon, reinstall the `drweb-configd` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation is already in progress*

Error code: `x2`

Internally used name: `EC_ALREADY_IN_PROGRESS`

Description: The operation is already in progress.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Wait for the operation to finish. If necessary, repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation is in pending state*

Error code: `x3`

Internally used name: `EC_IN_PENDING_STATE`

Description: An operation requested by the user is in a pending state (possibly, a network connection is currently being established or one of the components is starting and initializing, which may take a

long time).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Wait for the operation to start. If necessary, repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Interrupted by user*

Error code: `x4`

Internally used name: `EC_INTERRUPTED_BY_USER`

Description: The action has been terminated by the user (possibly, it was taking a long time).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation canceled*

Error code: `x5`

Internally used name: `EC_CANCELED`

Description: The action has been canceled.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action.

If the error persists, contact our technical support and provide the error code.

Error message: *IPC connection terminated*

Error code: x6

Internally used name: EC_LINK_DISCONNECTED

Description: An IPC connection to one of the Dr.Web Industrial for Unix File Servers components has been terminated (possibly, the component was shut down due to being idle or by a user request).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

If the operation did not finish, repeat it later. Otherwise, the connection termination is not an error.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid IPC message size*

Error code: x7

Internally used name: EC_BAD_MESSAGE_SIZE

Description: A message of an invalid size has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Industrial for Unix File Servers:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid IPC message format*

Error code: x8

Internally used name: EC_BAD_MESSAGE_FORMAT

Description: A message of an invalid format has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Industrial for Unix File Servers:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Not ready*

Error code: `x9`

Internally used name: `EC_NOT_READY`

Description: The required action cannot be performed because the necessary component or device has not been initialized yet.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Component is not installed*

Error code: `x10`

Internally used name: `EC_NOT_INSTALLED`

Description: The component necessary for running the required function is not installed.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Install or reinstall the necessary component. If you do not know the component name, try to determine it from the log file.
2. If the installation or reinstallation of the necessary component does not help, reinstall Dr.Web Industrial for Unix File Servers.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected IPC message*

Error code: `x11`

Internally used name: `EC_UNEXPECTED_MESSAGE`

Description: An invalid message has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Industrial for Unix File Servers by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *IPC protocol violation*

Error code: `x12`

Internally used name: `EC_PROTOCOL_VIOLATION`

Description: A protocol violation has occurred during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Industrial for Unix File Servers by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Subsystem state is unknown*

Error code: `x13`

Internally used name: `EC_UNKNOWN_STATE`

Description: The required operation cannot be performed because a subsystem of Dr.Web Industrial for Unix File Servers is in an unknown state.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the operation.
2. If the error persists, restart Dr.Web Industrial for Unix File Servers by running the command:

   ```
   # service drweb-configd restart
   ```

   and repeat the operation afterwards.

If the error persists, contact our technical support and provide the error code.

Error message: *Path must be absolute*

Error code: `x20`

Internally used name: `EC_NOT_ABSOLUTE_PATH`

Description: A relative path to a file or a directory has been specified, whereas an absolute path is required.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Make the file or directory path absolute and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not enough memory*

Error code: `x21`

Internally used name: `EC_NO_MEMORY`

Description: Not enough memory to complete the requested operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Increase the size of the memory available for Dr.Web Industrial for Unix File Servers processes (for example, by changing the limits using the `ulimit` command), restart Dr.Web Industrial for Unix File Servers and repeat the operation.

In some cases the `systemd` system service can ignore the specified limit changes. In this case, edit (or create if it does not exist) the file `/etc/systemd/system/drweb-configd.service.d/limits.conf` and indicate the changed limit value in it, for example:

```
[Service]
LimitDATA=32767
```

Available `systemd` limits can be determined using the `man systemd.exec` command.

Restart Dr.Web Industrial for Unix File Servers by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *I/O error*

Error code: `x22`

Internally used name: `EC_IO_ERROR`

Description: An input/output error has occurred (for example, a disk device has not been initialized yet or a partition of the file system is not available anymore).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Ensure the availability of the required I/O device or partition of the file system and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *No such file or directory*

Error code: `x23`

Internally used name: `EC_NO_SUCH_ENTRY`

Description: An attempt to access a non-existent file or directory has been made.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the indicated path. If necessary, correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Permission denied*

Error code: `x24`

Internally used name: `EC_PERMISSION_DENIED`

Description: Insufficient privileges to access the specified file or directory.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Ensure that the path is correct and the component has the required privileges. If access to the object is denied, change its permissions or elevate component privileges and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a directory*

Error code: `x25`

Internally used name: `EC_NOT_A_DIRECTORY`

Description: The specified object of the file system is not a directory.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the object path. Specify the correct path and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Data file corrupted*

Error code: `x26`

Internally used name: `EC_DATA_CORRUPTED`

Description: The requested data is corrupted.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the operation.

2. If the error persists, restart Dr.Web Industrial for Unix File Servers:

```
# service drweb-configd restart
```

and repeat the operation afterwards.

If the error persists, contact our technical support and provide the error code.

Error message: *File already exists*

Error code: x27

Internally used name: EC_FILE_EXISTS

Description: A file with the same name already exists.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the file path. If necessary, correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Read-only file system*

Error code: x28

Internally used name: EC_READ_ONLY_FS

Description: The file system is read-only.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the object path. Change it so as to indicate a writable partition of the file system and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Network error*

Error code: x29

Internally used name: EC_NETWORK_ERROR

Description: A network error has occurred (possibly, a remote host stopped responding unexpectedly or required connection failed).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the network is available and network settings are correct. If necessary, change the network settings and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a drive*

Error code: `x30`

Internally used name: `EC_NOT_A_DRIVE`

Description: The input/output device being accessed is not a disk device.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified device name. Correct the path so as to indicate a disk device and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected EOF*

Error code: `x31`

Internally used name: `EC_UNEXPECTED_EOF`

Description: The end of the file has been reached enexpectedly while reading data.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified file name. If necessary, correct the path so as to indicate the correct file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *File was changed*

Error code: `x32`

Internally used name: `EC_FILE_WAS_CHANGED`

Description: The file being scanned has been modified.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat scanning.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a regular file*

Error code: `x33`

Internally used name: `EC_NOT_A_REGULAR_FILE`

Description: The file system object being accessed is not a regular file (it may a directory, a socket and so on).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified file name. If necessary, change the path so as to indicate a regular file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Name already in use*

Error code: `x34`

Internally used name: `EC_NAME_ALREADY_IN_USE`

Description: A file with the same name already exists.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified path. Correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Host is offline*

Error code: `x35`

Internally used name: `EC_HOST_OFFLINE`

Description: A remote host cannot be accessed over the network.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the required host is available. If necessary, correct the host address and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Resource limit reached*

Error code: `x36`

Internally used name: `EC_LIMIT_REACHED`

Description: A limit on resource usage has been reached.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the availability of the required resource. If necessary, raise the limit on the usage of this resource and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Mounting points are different*

Error code: `x37`

Internally used name: `EC_CROSS_DEVICE_LINK`

Description: Restoring the file implies moving it between two different mount points.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log`

command.

Troubleshooting

Choose another path for restoring the file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Unpacking error*

Error code: x38

Internally used name: EC_UNPACKING_ERROR

Description: Unable to unpack an archive (it may be password-protected or corrupted).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Make sure that the archive is not corrupted. If the archive is protected with a password, remove the protection having entered the correct password and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Virus database corrupted*

Error code: x40

Internally used name: EC_BASE_CORRUPTED

Description: Virus databases are corrupted.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Root.VirusBaseDir -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported virus database version*

Error code: `x41`

Internally used name: `EC_OLD_BASE_VERSION`

Description: Current virus databases are intended for an earlier program version.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Empty virus database*

Error code: `x42`

Internally used name: `EC_EMPTY_BASE`

Description: The virus database is empty.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Object cannot be cured*

Error code: `x43`

Internally used name: `EC_CAN_NOT_BE_CURED`

Description: The **Cure** action has been applied to an incurable object.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Select an action that can be applied to this object and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported virus database combination*

Error code: x44

Internally used name: EC_INVALID_BASE_SET

Description: Incompatible virus databases.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Scan limit reached*

Error code: x45

Internally used name: EC_SCAN_LIMIT_REACHED

Description: The specified limits have been exceeded during object scanning (for example, on the size of an unpacked file, on the nesting depth and so on).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Change the scanning limits (in the component settings) using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.

2. After changing the settings, repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Authentication failed*

Error code: `x47`

Internally used name: `EC_AUTH_FAILED`

Description: Invalid user credentials have been used for authentication.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Provide valid credentials of a user having required privileges and try to authenticate again.

If the error persists, contact our technical support and provide the error code.

Error message: *Authorization failed*

Error code: `x48`

Internally used name: `EC_NOT_AUTHORIZED`

Description: The current user has insufficient privileges for performing the requested operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Provide valid credentials of a user having required privileges and try to authorize again.

If the error persists, contact our technical support and provide the error code.

Error message: *Access token is invalid*

Error code: `x49`

Internally used name: `EC_INVALID_TOKEN`

Description: A component of Dr.Web Industrial for Unix File Servers has provided an invalid authorization token in an attempt to execute an operation requiring elevated privileges.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log`

command.

Troubleshooting

Authenticate by providing valid credentials of a user having required privileges and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid argument*

Error code: `x60`

Internally used name: `EC_INVALID_ARGUMENT`

Description: Unable to run the command since an invalid argument has been provided.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the command syntax and format.
2. Repeat the required action having indicated a valid argument.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid operation*

Error code: `x61`

Internally used name: `EC_INVALID_OPERATION`

Description: An attempt to run an invalid command has been made.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action using a valid command.

If the error persists, contact our technical support and provide the error code.

Error message: *Superuser privileges required*

Error code: `x62`

Internally used name: `EC_ROOT_ONLY`

Description: Superuser privileges are required to perform the required action.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Elevate you privileges to superuser ones and repeat the required action. To elevate the privileges, use the `su` or `sudo` command.

If the error persists, contact our technical support and provide the error code.

Error message: *Not allowed in centralized protection mode*

Error code: `x63`

Internally used name: `EC_STANDALONE_MODE_ONLY`

Description: The required action can be performed only when Dr.Web Industrial for Unix File Servers operates in standalone mode.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Switch Dr.Web Industrial for Unix File Servers to the standalone mode and repeat the operation.

To switch Dr.Web Industrial for Unix File Servers to the standalone mode:

- clear the **Enable centralized protection mode** check box on the **Centralized protection** of the management web interface;
- or run the command:

```
# drweb-ctl esdisconnect
```

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported OS*

Error code: `x64`

Internally used name: `EC_NON_SUPPORTED_OS`

Description: Dr.Web Industrial for Unix File Servers does not support the operating system installed on the host.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Install an operating system from the list provided in system requirements.

If the error persists, contact our technical support and provide the error code.

Error message: *Feature not implemented*

Error code: `x65`

Internally used name: `EC_UNKNOWN_OPTION`

Description: The required features of the component are not implemented in the current version.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Try to reset the settings of Dr.Web Industrial for Unix File Servers to defaults.

To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

- for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Unknown option*

Error code: `x66`

Internally used name: `EC_UNKNOWN_OPTION`

Description: The configuration file contains parameters that are unknown to or not supported by the current version of Dr.Web Industrial for Unix File Servers.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Open the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD in any text editor, remove the line containing the invalid parameter. Save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

2. If this does not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *Unknown section*

Error code: `x67`

Internally used name: `EC_UNKNOWN_SECTION`

Description: The configuration file contains sections unknown to or not supported by the current version of Dr.Web Industrial for Unix File Servers.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Open the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD in any text editor, remove the unknown (unsupported) section. Save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

2. If this does not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

- for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid option value*

Error code: `x68`

Internally used name: `EC_INVALID_OPTION_VALUE`

Description: One or more parameters in the configuration file have invalid values.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Set the parameter value using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.

   If you do not know which value is valid for the parameter, refer to a man page for the component which uses this parameter. You can also restore a default value of the parameter.
2. You can also directly edit the configuration file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD. To do this, open it in any text editor, find the line containing an invalid parameter value, set a valid value, then save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```
3. If the previous steps did not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

   ```
   # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
   ```

```
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid state*

Error code: `x69`

Internally used name: `EC_INVALID_STATE`

Description: Dr.Web Industrial for Unix File Servers or one of its components is in an invalid state and cannot complete the required operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the required action later.
2. If the error persists, restart Dr.Web Industrial for Unix File Servers by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Only one value allowed*

Error code: `x70`

Internally used name: `EC_NOT_LIST_OPTION`

Description: A list of values is set for a single-valued parameter in the configuration file.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Set the parameter value using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.

If you do not know which value is valid for the parameter, refer to a man page for the component which uses this parameter. You can also restore a default value of the parameter.

2. You can also directly edit the configuration file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD. To do this, open it in any text editor, find the line containing an invalid parameter value, set a valid value, then save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

```
# service drweb-configd restart
```

3. If the previous steps did not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

   - for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Record not found*

Error code: `x80`

Internally used name: `EC_RECORD_NOT_FOUND`

Description: The threat record is missing (it may have already been proccessed by another component).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *Record is in process now*

Error code: `x81`

Internally used name: `EC_RECORD_BUSY`

Description: The threat is already being processed by another component.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *File has already been quarantined*

Error code: `x82`

Internally used name: `EC_QUARANTINED_FILE`

Description: The file is already in quarantine. Probably, another component has already processed the threat.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *Update zone is not provided on disk*

Error code: `x84`

Internally used name: `EC_NO_ZONE_ON_DISK`

Description: An attempt to update the virus bases in offline mode was unsuccessful.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Make sure that the path to the device used for updating is correct.
2. Make sure that the user as whom you are trying to update has read permissions for the directory with updates.

If the error persists, contact our technical support and provide the error code.

Error message: *Cannot backup before update*

Error code: x89

Internally used name: EC_BACKUP_FAILED

Description: Prior to downloading updates from an update server, an attempt to make a backup copy of the files to be updated has failed.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the directory that stores backup copies of the files to be updated. Change the path, if necessary (the `BackupDir` parameter in the `[Update]` section of the configuration file).
   - To view and change the path, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Update.BackupDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Update.BackupDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Update.BackupDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```
3. If the error persists, check whether the user as whom the component is running has write permissions for a directory specified in the `BackupDir` parameter. The name of this user is specified in the `RunAsUser` parameter. If necessary, change the user specified in the `RunAsUser` parameter or grant the lacking permissions in the directory attributes.
4. If the error persists, reinstall the `drweb-update` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid DRL file*

Error code: x90

Internally used name: EC_BAD_DRL_FILE

Description: The integrity of one of the files storing a list of update servers is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `#`

less /var/log/messages command for FreeBSD). You can also run the # drweb-ctl log command.

Troubleshooting

1. Check the path to the file storing the list of servers and change the path if necessary (parameters containing *DrlDir in the [Update] section of the configuration file).
   - To view and change the path, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To view the current parameter value, run the command (replace <**DrlDir*> with a specific parameter name. If the parameter name is unknown, view all parameter values in the section, skipping the command part in square brackets):

   ```
   $ drweb-ctl cfshow Update[.<*DrlDir>]
   ```

   To set a new parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> <new path>
   ```

   To restore the default parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```
3. If the error persists, reinstall the drweb-update package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid LST file*

Error code: x91

Internally used name: EC_BAD_LST_FILE

Description: The integrity of the file storing a list of virus databases to be updated is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the # journalctl command for GNU/Linux or the # less /var/log/messages command for FreeBSD). You can also run the # drweb-ctl log command.

Troubleshooting

1. Update virus databases again after some time:
   - click **Update** on the **Main** page of the management web interface;

- or run the command:

```
$ drweb-ctl update
```

2. If the error persists, reinstall the `drweb-update` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid compressed file*

Error code: `x92`

Internally used name: `EC_BAD_LZMA_FILE`

Description: The integrity of the downloaded file containing updates is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update virus databases again after some time:

- click **Update** on the **Main** page of the management web interface;
- or run the command:

```
$ drweb-ctl update
```

If the error persists, contact our technical support and provide the error code.

Error message: *Proxy authentication error*

Error code: `x93`

Internally used name: `EC_PROXY_AUTH_ERROR`

Description: Unable to connect to update servers via the proxy server specified in the settings.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the parameters used to connect to the proxy server (set with the `Proxy` parameter in the `[Update]` section of the configuration file).
   - To view and set the connection parameters, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

To get the current parameter value, run the command:

```
$ drweb-ctl cfshow Update.Proxy
```

To set a new parameter value, run the command:

```
# drweb-ctl cfset Update.Proxy <new parameters>
```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Update.Proxy -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *No update servers available*

Error code: `x94`

Internally used name: `EC_NO_UPDATE_SERVERS`

Description: Unable to connect to any of the update servers.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check if the network is available. Change network settings if necessary.
2. If you can access the network only using a proxy server, configure connection to the proxy server (the `Proxy` parameter in the `[Update]` section of the configuration file).
   - To view and set the connection parameters, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Update.Proxy
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy <new parameters>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy -r
   ```
3. If the network connection parameters (including the parameters of the proxy server) are correct, but the error persists, make sure that you use a relevant list of update servers. The list of used update servers is set in `*DrlDir` parameters in the `[Update]` section of the configuration file.

If `*CustomDrlDir` parameters indicate an existing valid file storing a list of servers, the servers specified there will be used instead of the servers of the standard update zone (the value specified in the corresponding `*DrlDir` parameter is ignored).

- To view and set the connection parameters, go to the **Updater** page of the management web interface.
- You can also use the command-line management tool.

To view the current parameter value, run the command (replace <**DrlDir*> with a specific parameter name. If the parameter name is unknown, view all parameter values in the section, skipping the command part in square brackets):

```
$ drweb-ctl cfshow Update[.<*DrlDir>]
```

To set a new parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

```
# drweb-ctl cfset Update.<*DrlDir> <new path>
```

To restore the default parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

```
# drweb-ctl cfset Update.<*DrlDir> -r
```

4. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid key file format*

Error code: `x95`

Internally used name: `EC_BAD_KEY_FORMAT`

Description: The key file format is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check that you have a key file and a correct path to it. You can specify the path to the key file in the `KeyPath` parameter in the `[Root]` section of the configuration file.
   - To view and set the path to the key file, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.KeyPath
   ```

To set a new parameter value, run the command:

```
# drweb-ctl cfset Root.KeyPath <path to file>
```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Root.KeyPath -r
```

2. If you do not have a key file or the key file being used is corrupted, purchase and install it. For more details on the key file, purchasing and installing it, refer to the Licensing section.
3. To install the key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
4. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *License is already expired*

Error code: `x96`

Internally used name: `EC_EXPIRED_KEY`

Description: The license has expired.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Network operation timed out*

Error code: `x97`

Internally used name: `EC_NETWORK_TIMEDOUT`

Description: A network connection timed out (possibly, a remote host has stopped responding unexpectedly or the required connection has failed).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the network is available and network settings are correct. If necessary, change the network settings and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid checksum*

Error code: `x98`

Internally used name: `EC_BAD_CHECKSUM`

Description: The checksum of the downloaded file with updates is invalid.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update virus databases again after some time:

- click **Update** on the **Main** page of the management web interface;
- or run the command:

```
$ drweb-ctl update
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid trial license*

Error code: `x99`

Internally used name: `EC_BAD_TRIAL_KEY`

Description: The demo key file is invalid (for example, it was received for another computer).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Request a new demo period for this computer or purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Blocked license key*

Error code: `x100`

Internally used name: `EC_BLOCKED_LICENSE`

Description: The current license is blocked (the terms of use of Dr.Web Industrial for Unix File Servers may be violated).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid license*

Error code: `x101`

Internally used name: `EC_BAD_LICENSE`

Description: The license is intended for another product or does not cover Dr.Web Industrial for Unix File Servers components.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid configuration*

Error code: `x102`

Internally used name: `EC_BAD_CONFIG`

Description: A component of Dr.Web Industrial for Unix File Servers cannot operate because of invalid configuration settings.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If you do not know the name of the component that causes the error, try to determine it by viewing the log file.
2. If the error is caused by the SpIDer Guard component, the mode that is selected for the component operation is probably not supported by your operating system. Check the selected component mode and change it, if necessary, by setting the *Auto* value (the `Mode` parameter in the `[LinuxSpider]` section of the configuration file).
   - To view and change the mode, go to the **SpIDer Guard** page of the management web interface.
   - You can also use the command-line management tool.

   To set the value to *Auto*, run the command:

   ```
   # drweb-ctl cfset LinuxSpider.Mode Auto
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset LinuxSpider.Mode -r
   ```

   Operation of SpIDer Guard is guaranteed only if your OS is on the list of supported systems (see the System Requirements and Compatibility section).

3. If the error is caused by another component, reset its settings to defaults using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands;
   - edit the configuration file manually (delete all parameters from the component section).
4. If the previous steps did not help, reset Dr.Web Industrial for Unix File Servers settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid executable file*

Error code: `x104`

Internally used name: `EC_BAD_EXECUTABLE`

Description: Failed to start the component. The executable file is corrupted or the file path is invalid.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If you do not know the name of the component that causes the error, try to determine it by viewing the log file.
2. Check the executable path to the component in the Dr.Web Industrial for Unix File Servers configuration file (the `ExePath` parameter in the component section) by running the command (replace <*component section*> with the name of the corresponding section of the configuration file):

   ```
   $ drweb-ctl cfshow <component section>.ExePath
   ```

3. Reset the path to its default value by running the command (replace <*component section*> with the name of the corresponding section of the configuration file):

   ```
   # drweb-ctl cfset <component section>.ExePath -r
   ```

4. If the previous steps did not help, reinstall the package of the corresponding component.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Core engine is not available*

Error code: `x105`

Internally used name: `EC_NO_CORE_ENGINE`

Description: The file of Dr.Web Virus-Finding Engine is missing or unavailable.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the anti-virus engine file `drweb32.dll` and, if necessary, correct it (the `CoreEnginePath` parameter in the `[Root]` section of the configuration file).

- To view and change the path, go to the **General Settings** page of the management web interface.
- You can also use the command-line management tool.

To get the current parameter value, run the command:

```
$ drweb-ctl cfshow Root.CoreEnginePath
```

To set a new parameter value, run the command:

```
# drweb-ctl cfset Root.CoreEnginePath <new path>
```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Root.CoreEnginePath -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```
3. If the path is correct and the error persists after updating virus databases, reinstall the `drweb-bases` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *No virus databases*

Error code: `x106`

Internally used name: `EC_NO_VIRUS_BASES`

Description: Virus databases are missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Root.VirusBaseDir -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Process terminated by signal*

Error code: `x107`

Internally used name: `EC_APP_TERMINATED`

Description: A component has been shut down (possibly, owing to being idle or by a user request).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If the operation has not finished, repeat it. Otherwise, the shutdown is not an error.
2. If the component shuts down constantly, reset its settings to default using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands;
   - edit the configuration file manually (delete all parameters from the component section).
3. If this did not help, reset Dr.Web Industrial for Unix File Servers settings to default.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:
   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```
   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected process termination*

Error code: `x108`

Internally used name: `EC_APP_CRASHED`

Description: A component has been shut down because of a failure.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the terminated operation.
2. If the component constantly shuts down abnormally, reset its settings to default using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands;
   - edit the configuration file manually (delete all parameters from the component section).
3. If this did not help, reset Dr.Web Industrial for Unix File Servers settings to default.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:
   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```
   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```
4. If the error persists after resetting Dr.Web Industrial for Unix File Servers settings, reinstall the component package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Incompatible software detected*

Error code: `x109`

Internally used name: `EC_INCOMPATIBLE`

Description: One or several components of Dr.Web Industrial for Unix File Servers cannot operate properly. Their operation is impeded by software in your system.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Disable or reconfigure the conflicting software such as to prevent any interference in the Dr.Web Industrial for Unix File Servers operation.

If the error persists, contact our technical support and provide the error code.

Error message: *MeshD is not available*

Error code: `x114`

Internally used name: `EC_NO_MESHD`

Description: The Dr.Web MeshD component required for load balancing during file scanning is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-meshd` executable file. Change the path if necessary (the `ExePath` parameter in the `[MeshD]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow MeshD.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset MeshD.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset MeshD.ExePath -r
   ```

2. If the configuration does not contain Dr.Web MeshD settings or the error persists after entering the correct path, install or reinstall the `drweb-meshd` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *ScanEngine is not available*

Error code: `x119`

Internally used name: `EC_NO_SCAN_ENGINE`

Description: The Dr.Web Scanning Engine component required for threat detection is missing or has failed to start.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-se` executable file. Change the path if necessary (the `ExePath` parameter in the `[ScanEngine]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow ScanEngine.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset ScanEngine.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset ScanEngine.ExePath -r
   ```

2. If the error persists after entering the correct path:

   - Run the command:

     ```
     $ drweb-ctl rawscan /
     ```

   If the line "`Error: No valid license provided`" is output, a valid key file is missing. Register Dr.Web Industrial for Unix File Servers and receive a license. After receiving the license, check whether the key file is available and install it if necessary.

   - If your operating system uses SELinux, configure the security policy for the `drweb-se` module (see the section Configuring SELinux Security Policies).

3. If the configuration does not contain Dr.Web Scanning Engine settings or the previous steps did not help, install or reinstall the `drweb-se` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *FileCheck is not available*

Error code: `x120`

Internally used name: `EC_NO_FILE_CHECK`

Description: The Dr.Web File Checker component is missing or has failed to start.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log`

command.

Troubleshooting

1. Check the path to the `drweb-filecheck` executable file. Change the path if necessary (the `ExePath` parameter in the `[FileCheck]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow FileCheck.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset FileCheck.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset FileCheck.ExePath -r
   ```

2. If the error persists after entering the correct path:
   - If your operating system uses SELinux, configure the security policy for the `drweb-filecheck` module (see the section Configuring SELinux Security Policies).
3. If the configuration does not contain Dr.Web File Checker settings or the previous steps did not help, install or reinstall the `drweb-filecheck` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *ESAgent is not available*

Error code: `x121`

Internally used name: `EC_NO_ESAGENT`

Description: The Dr.Web ES Agent component required to connect to a centralized protection server is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-esagent` executable file. Change the path if necessary (the `ExePath` parameter in the `[ESAgent]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow ESAgent.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset ESAgent.ExePath <new path>
   ```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset ESAgent.ExePath -r
```

2. If the configuration does not contain Dr.Web ES Agent settings or the error persists after entering the correct path, install or reinstall the `drweb-esagent` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *NetCheck is not available*

Error code: `x123`

Internally used name: `EC_NO_NETCHECK`

Description: The Dr.Web Network Checker component required to scan files over the network is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-netcheck` executable file. Change the path if necessary (the `ExePath` parameter in the `[Netcheck]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Netcheck.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Netcheck.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Netcheck.ExePath -r
   ```

2. If the configuration does not contain Dr.Web Network Checker settings or the error persists after entering the correct path, install or reinstall the `drweb-netcheck` package.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected error*

Error code: `x125`

Internally used name: `EC_UNEXPECTED_ERROR`

Description: An unexpected error has occurred in operation of one or several components.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Industrial for Unix File Servers log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Industrial for Unix File Servers by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Errors without a code

Symptoms: One of the following errors occurs on FreeBSD 13, 14 or 15:

- Dr.Web ConfigD component fails to start.
- Paths to sockets specified by parameters `Root.AdminSocketPath` and `Root.AdminSocketPath` (by default, `/var/run/.com.drweb.public` and `/var/run/.com.drweb.admin` correspondingly) do not exist or these sockets are unavailable.

Description: Dr.Web Industrial for Unix File Servers operation is impossible because the `compat12x` compatibility package is not installed.

Troubleshooting

Install the `compat12x` compatibility package. An example of how to do this is provided below.

1. Start a console or a terminal emulator and run the commands:

   ```
   # pkg update
   # pkg search compat12x
   ```

   The output of the last command will provide the full package name from your OS repository.
2. Specify the package name and install it, for example:

   ```
   # pkg install compat12x-amd64-12.2.1202000.20210406_2
   ```

If the error persists, contact our technical support.

Symptoms: A web browser cannot establish connection to the Dr.Web management web interface, Dr.Web components are not in the list of running processes (`ps ax | grep drweb`), an attempt to run any command such as `drweb-ctl <command>`, except for `drweb-ctl rawscan`, results in one of the following errors:

`Error: connect: No such file or directory: "<path>/.com.drweb.public"`

or

`Error: connect: Connection refused: "<path>/.com.drweb.public".`

Description: Dr.Web Industrial for Unix File Servers cannot start as the Dr.Web ConfigD configuration daemon is not available.

Troubleshooting

1. Run the command:

   ```
   # service drweb-configd restart
   ```

   to restart Dr.Web ConfigD and Dr.Web Industrial for Unix File Servers in general.
2. If this command returns an error or has no effect, install the `drweb-configd` package separately.

   This also may mean that PAM authentication is not used in the system. If so, install and configure PAM since Dr.Web Industrial for Unix File Servers cannot operate correctly without it.
3. If the error persists, uninstall Dr.Web Industrial for Unix File Servers entirely and then reinstall it.

   For details on how to install and uninstall Dr.Web Industrial for Unix File Servers or its components, refer to sections Installing Dr.Web Industrial for Unix File Servers and Uninstalling Dr.Web Industrial for Unix File Servers.

If the error persists, contact our technical support.

10.7. Appendix G. List of Abbreviations

The following abbreviations were used in this manual without further interpretation:

Convention	Complete form
AD	Microsoft Active Directory
FQDN	Fully Qualified Domain Name
GID	Group ID (system user group identifier)
GNU	GNU project (GNU is Not Unix)
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure (via SSL/TLS)
ID	Identifier
IMA/EVM	Integrity Measurement Architecture and Extended Verification Module
IP	Internet Protocol
MBR	Master Boot Record
OID	(SNMP) Object ID
OS	Operating System
PID	Process ID (system process identifier)
PAM	Pluggable Authentication Modules
RPM	Red Hat Package Manager
RRA	Round-Robin Archive
RRD	Round-Robin Database
SMB	Server Message Block (file access protocol)
SNMP	Simple Network Management Protocol
SP	Service Pack
SSH	Secure Shell
SSL	Secure Sockets Layer

Convention	Complete form
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UID	User ID (system user identifier)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBR	Volume Boot Record

© Doctor Web, 2026