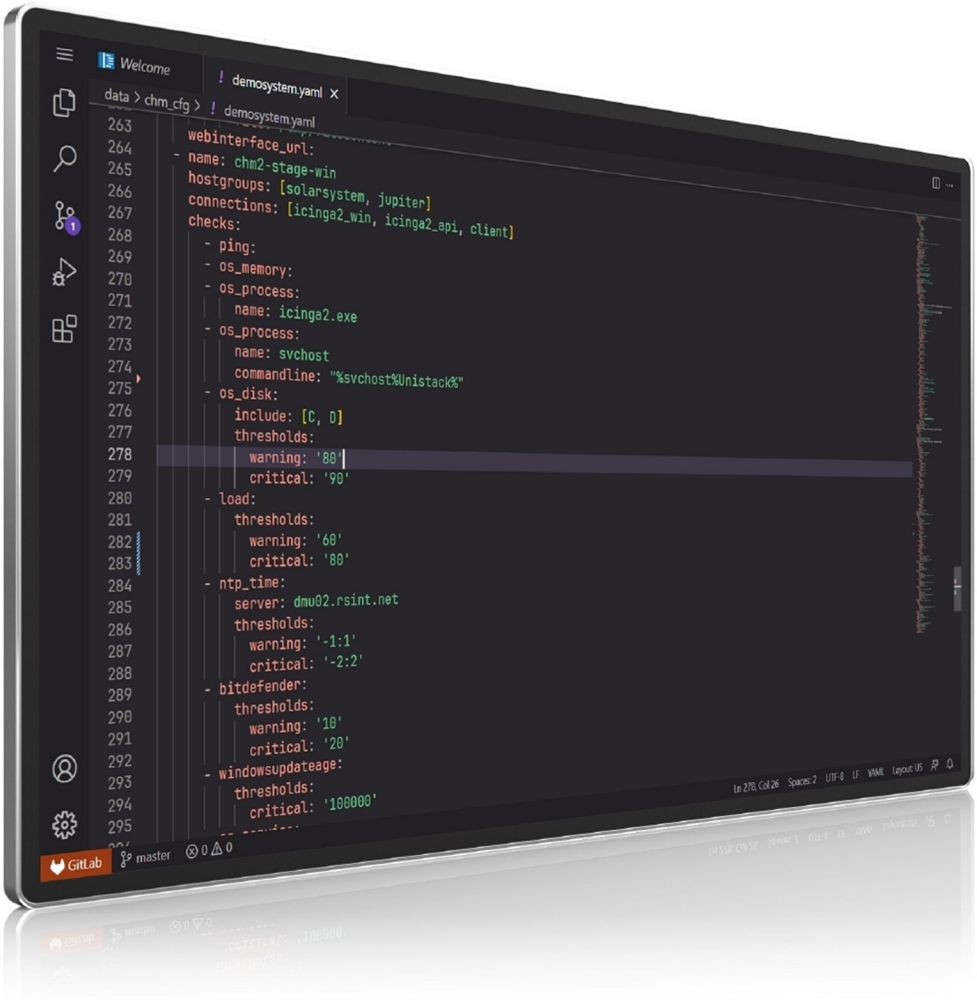
R&S®CHM
System Status Monitoring
Configuration
Configuration Manual



1179613702
Version 15

ROHDE&SCHWARZ
Make ideas real

This document describes implementation and configuration of the following software with version v2610 and higher:

- R&S®CHM, system status monitoring software (3067.6545.02)

© 2026 Rohde & Schwarz
Muehldorfstr. 15, 81671 Muenchen, Germany
Phone: +49 89 41 29 - 0
Email: info@rohde-schwarz.com
Internet: www.rohde-schwarz.com
Subject to change – data without tolerance limits is not binding.
R&S® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.
All other trademarks are the properties of their respective owners.

1179.6137.02 | Version 15 | R&S®CHM

Throughout this document, R&S® is indicated as R&S.

Contents

1 Welcome to R&S CHM 7

1.1 Key features 7

1.2 Documentation overview 7

1.2.1 Manuals 8

1.2.2 Brochure 8

1.2.3 Release notes and open source acknowledgment (OSA) 8

1.3 Using help 8

1.3.1 Navigating in the help 9

1.3.2 Using the search filter 9

2 What's new 11

2.1 Previous releases 11

2.1.1 R&S CHM v2607 12

2.1.2 R&S CHM v2604 12

2.1.3 R&S CHM v2601 13

2.1.4 R&S CHM v2511 14

3 Introduction 15

4 Installing R&S CHM 18

4.1 Installing the R&S CHM host without LCSM 19

4.2 Installing R&S CHM agents 20

4.2.1 Installing Windows agents 21

4.2.2 Installing Linux agents 22

4.2.3 Updating R&S CHM on agents and clients 22

4.3 Installing R&S CHM clients 23

4.3.1 Installing the client software 23

4.3.2 Extending the chm.yaml 24

4.3.3 Connecting the client with the R&S CHM host 24

4.3.4 Handling certificates 25

4.3.5 Starting the client for the first time 26

4.3.5.1 Configuring application logging 27

4.3.6 Setting up SSO 28

4.4 Firewall **30**

5 Deploying certificates **31**

5.1 Using the R&S CHM built-in CA **31**

5.2 Using CA-signed certificates **34**

5.3 Removing certificates **34**

5.4 Configuring a user-defined certificate location on Windows hosts **35**

5.5 Deploying certificates for distributed monitoring **35**

5.5.1 External CA 36

5.5.2 R&S CHM built-in CA 36

5.5.2.1 Deploying certificates with signing requests 37

5.5.2.2 Deploying certificates with tickets 38

6 Configuring status monitoring **40**

6.1 Introduction to the YAML syntax **41**

6.2 Understanding aggregated states **42**

6.3 Changing the configuration **43**

6.4 Configuring hosts **44**

6.5 Configuring web GUI users **61**

6.6 Configuring R&S CHM features **71**

6.7 Managing password identifiers **76**

6.8 Configuring R&S RAMON for monitoring **78**

6.8.1 Configuring the chmrd service 79

6.8.2 Configuring the System Control view 82

6.9 Configuring graphical system views (maps) **83**

6.10 Configuring the SNMP upstream interface **88**

6.10.1 Activating the interface 88

6.10.2 Configuring SNMPv2 traps 89

6.10.3 Configuring SNMPv3 traps/informs 90

6.11 Configuring generic monitoring of SNMP third-party devices **90**

6.11.1 YAML file layout 91

6.11.2 'get' check syntax 92

6.11.3 'expect:' operators 94

6.11.4 'on_pass:' and 'on_fail:' outcomes 96

6.11.5 Template variables 97

6.11.6 Expressions ('when', 'filter', 'derived')....99
6.11.7 'type: walk' – table-oriented checks....100
6.11.8 Full worked example – Synology NAS....104
6.11.9 Full worked example – IF-MIB ifTable....105
6.11.10 Configuring snmp_flex in the chm.yaml....106
6.11.11 Troubleshooting....107
6.12 Configuring distributed monitoring....108
6.12.1 Configuring high availability monitoring....109
6.12.1.1 Editing the YAML configuration for HA monitoring....110
6.12.1.2 Configuring R&S CHM agents for HA monitoring....111
6.12.2 Configuring subsystems....112
6.12.3 Configuring multi-level monitoring....113
6.12.3.1 Editing the YAML configuration for multi-level monitoring....115
6.12.3.2 Configuring agents for multi-level monitoring....117
6.12.4 Configuring multi-level HA monitoring....118
6.12.4.1 Editing the YAML configuration for multi-level HA monitoring....119
6.12.4.2 Configuring agents for multi-level HA monitoring....121
6.13 Configuring automatic system control....122
6.14 Using common keys....125
6.15 Using frequent keys....128
7 Configuring status checks....134
8 YAML configuration examples....191
8.1 R&S CHM host configuration....191
8.2 Linux host configurations....192
8.3 Example configuration for R&S CHM Windows agents....194
8.4 Example configuration for R&S CHM Linux agents....195
9 Troubleshooting....196
9.1 Web GUI is unavailable....196
9.2 Web GUI shows message Wrong SNMP PDU digest....196
9.3 Web GUI shows 404 error....196
9.4 Troubleshooting installation problems on Windows agents....197
9.4.1 Accessing the event log....197

9.4.2 Accessing the MSI log files 198

9.5 Contacting customer support 199

9.6 Reporting cybersecurity issues 199

Glossary: Abbreviations and terms 201

Glossary: Specifications 207

List of YAML keys 208

1 Welcome to R&S CHM

The R&S CHM software monitors status information from various system components that are connected to the network. The web-based user interface visualizes system state parameters, and lets you monitor and troubleshoot connected and configured Rohde & Schwarz instruments, devices with simple network management protocol (SNMP) interface, and other hosts.

Target audience

This manual familiarizes you with implementation and configuration of R&S CHM, including configuration of monitoring services. As a **system administrator** or **software integrator**, you install and configure R&S CHM on the R&S CHM host. The descriptions assume that you already have a comprehensive knowledge of system setup and configuration.

For information on using the R&S CHM web GUI, see the "R&S CHM System Status Monitoring" user manual.

1.1 Key features

R&S CHM system status monitoring provides the following high-level features:

- Run on a security-enhanced Linux® distribution (SELinux).
- Run on a hardened operating system according to DISA STIGs. For information, see https://public.cyber.mil/stigs/.
- Run unattended for a long period of time.
- Continuously monitor the status of hosts and services, e.g. used disk space.
- Allow configuration of device-specific monitoring services.
- Reduce downtime of system components.
- Troubleshooting problems.
- Encrypted communication between R&S CHM and monitored hosts.
- Secure password handling.

1.2 Documentation overview

This section provides an overview of the R&S CHM user documentation. Unless specified otherwise, you find the documents at:

www.rohde-schwarz.com/product/chm

1.2.1 Manuals

The manuals are provided in two formats. The PDF format is contained in the software delivery. An HTML5-based help format is available on the R&S CHM web GUI.

The latest versions of the manuals are available for download or for immediate display on the internet at:

www.rohde-schwarz.com/manual/chm

- **"R&S CHM System Status Monitoring" user manual**:
  Introduces the R&S CHM and describes how to start working with the web GUI that lets you monitor the "health status" of the system in detail.
- **"R&S CHM System Status Monitoring Configuration" configuration manual**:
  Provides a description of all configuration options and describes how you implement and set up R&S CHM on all system components.

To obtain help in the web GUI

1. On the left navigation area of the R&S CHM web GUI, select "Extras" > "User Manual".

   The help opens in the R&S CHM web GUI (English).

2. To show the manual in a different language, e.g. German or French, select the "DEU" or "FRA" tabs on the top of the "User Manual" area.

See also: Section 1.3, "Using help", on page 8

1.2.2 Brochure

The brochure provides an overview of the software and deals with the specific characteristics:

www.rohde-schwarz.com/brochure-datasheet/chm

1.2.3 Release notes and open source acknowledgment (OSA)

The release notes list new features, improvements and known limitations of the current software version, and contain a release history.

The open source acknowledgment document provides verbatim license texts of the used open source software.

Both documents are contained in the software delivery.

1.3 Using help

By default, the help opens in the main R&S CHM web GUI window.

To open the help in a separate window

You can read the help also in parallel to the R&S CHM web GUI.

1. In the browser, duplicate the "CHM Web" tab.
2. Drag the tab, so that it opens in a new window.

1.3.1 Navigating in the help

You can use the table of contents and the index on the left or the search text box on the top right to find the right piece of information. Navigation between help pages and table of contents is synchronized.

To navigate back or forward, use the commands on the shortcut menu of your browser.

1.3.2 Using the search filter

Use a search filter to narrow down the number of results. Currently, the following information type filters are assigned to the help pages:

- "Basics and concepts": Finds the search term in help pages with conceptual or descriptive information.
- "Graphical user interface": Finds the search term in all "GUI reference" help pages.
- "How to": Finds the search term in help pages with step-by-step instructions for completing tasks.
- "Troubleshooting": Finds the search term in the "Troubleshooting" help pages.

To set a search filter

As an example, we search for step-by-step instructions.

1. Type the search term in the "Search" box, e.g. *search range*.
2. Select ▼.
3. Select "How to".
4. Select 🔍.

   The result list only shows help pages that contain step-by-step instructions.
5. Select a result.

   The search terms are highlighted on the help page.

   Search tips
   - You can also first set the filter and then type the search term.
   - You can return to the search list by using "Back" on the shortcut menu.
   - You can change the filter on the current search result to obtain results from other categories.
   - You can reset the filter using "All Files".
   - Keep the search short and simple using as few words as possible. Each space is regarded as a logical *and*.

- The logical expressions *and*, *or* and *not* to combine several search terms. E.g., the expression *not available* in the search expression excludes *available* in the search results.

2 What's new

This section summarizes the most important changes and enhancements of version **v2610** compared to version **v2607**. For more information about latest product and documentation changes, restrictions and known issues, see the release notes.

R&S CHM built-in CA

You can now use the R&S CHM built-in certificate authority (CA) to deploy certificates in distributed setups. R&S CHM supports two variants: certificates with signing requests and certificates with tickets.

Read more: Section 5.5.2, "R&S CHM built-in CA", on page 36

Generic monitoring of SNMP third-party devices

R&S CHM now provides an option to configure SNMP device checks independent from the main `chm.yaml`. Instead, you can declare the SNMP values to poll and how to interpret them in separate YAML files. The `chm.yaml` only consists of a minimal configuration under the `snmp_flex` key where you specify the file path. Furthermore, R&S CHM can read-in existing YAML files from a specific directory.

Read more: Section 6.11, "Configuring generic monitoring of SNMP third-party devices", on page 90

New or enhanced configuration keys and status checks

- `rccm`
  Monitors a device via RCCM device access layer API.
  Read more: `RCCM` on page 177
- `snmp_flex`
  Runs the generic YAML-driven SNMP check.
  Read more: `snmp_flex` on page 179
- `mac_address`
  Defines a MAC address check for an SNMP-connected network device.
  Read more: `mac_address` on page 162

2.1 Previous releases

This section contains release information from some previous R&S CHM releases.

- R&S CHM v2607 12
- R&S CHM v2604 12
- R&S CHM v2601 13
- R&S CHM v2511 14

2.1.1 R&S CHM v2607

Configurable web server TLS certificate

Specifies the path to a TLS certificate and corresponding private key on R&S CHM hosts and satellites. This feature already has been released with R&S CHM v2604.

Read more: `webserver` on page 61

Reporting status information

You can now configure specific reports that are shown on the web GUI. This feature already has been released with R&S CHM v2604.

Read more: `reporting` on page 74

New or enhanced configuration keys and status checks

- `reporting`
  This new permission allows users to view reports on the web GUI.
  Read more: `authorization` on page 68 > `permissions`
- `snmp_connection`
  The SNMP upstream interface now supports SNMPv3. Several keys have been added to `snmp_connection` to enable secure communications.
  Read more: `snmp_connection` on page 128
- `xcp_ng`
  Supports now SSH connections.
  Read more: `xcp_ng` on page 189

2.1.2 R&S CHM v2604

Automatic system control

If you configure automatic system control, R&S CHM can automatically execute a predefined set of commands triggered by check result events from the logic network. For example, R&S CHM can shut down host systems and switch off power outlets when temperature checks are critical. This feature already has been released with R&S CHM v2601.

Read more: Section 6.13, "Configuring automatic system control", on page 122

New or enhanced configuration keys and status checks

- `ar60`
  Checks the global status of the AR60 microwave modem.
  Read more: `ar60` on page 135
- `argus`
  Checks status information of connected RF equipment and the operating users in ARGUS.

Read more: argus on page 135

- check_kerberos_auth
  Monitor that Linux agents can authenticate against Active Directory.
  Read more: check_kerberos_auth on page 136
- snmp_diskspace
  Checks the disk usage of a server via SNMP and reports partition names that are in critical state.
  Read more: snmp_diskspace on page 179
- snmp_processes
  Monitors the Linux processes running on the system via SNMP.
  Read more: snmp_processes on page 181
- xcp_ng
  Checks the disk usage and VM status of an XCP-NG system.
  Read more: xcp_ng on page 189
- seclevel
  This key in snmp_connection is no longer supported.
- chm_agent_connection
  Key renamed to chm_agent_conn. You can use both key names in the chm.yaml file.
  Read more: chm_agent_conn on page 136
- raritan_pdu
  Sensor data is added.
  Read more: raritan_pdu on page 174
- ups
  Examples enhanced.
  Read more: ups on page 186
- msr4
  Monitors the status and the temperature of the R&S CHM multipurpose satellite receiver via SNMP. The check also informs about the firmware version.
  Read more: msr4 on page 164

2.1.3 R&S CHM v2601

Enhanced application logging on R&S CHM clients

You can now enable Windows® event logging on R&S CHM clients.

Read more: Section 4.3.5.1, "Configuring application logging", on page 27

Start the web GUI on the R&S CHM client in full-screen mode (maximized)

To start the web GUI maximized, you can add the configuration in the JSON configuration file on the client.

Read more: Section 4.3.3, "Connecting the client with the R&S CHM host", on page 24

New or enhanced configuration keys and status checks

- `check_kerberos_auth`
  Monitor that Linux agents can authenticate against Active Directory.
  Read more: `check_kerberos_auth` on page 136

2.1.4 R&S CHM v2511

Resetting uptime data information

The `forget_states_on_restart` feature lets you reset the timing information for the "UP" value after a restart of the R&S CHM service or restart of the R&S CHM master.

Read more: `forget_states_on_restart` on page 73

New or enhanced configuration keys and status checks

- `snmp_connection`
  Added the information that SNMP passwords require a minimum length of 8 characters.
  Read more: `snmp_connection` on page 128
- `raritan_pdu`
  Checks the outlet status of a Raritan power distribution unit (PDU).
  Read more: `raritan_pdu` on page 174

3 Introduction

The R&S CHM system status monitoring software provides an integrated, system-wide solution to collect status information continuously in a local area network (LAN). The software continuously performs checks for monitored hosts and services and evaluates the results. If R&S CHM detects an error condition, it creates an alert. The following figure provides an overview of a monitored system.

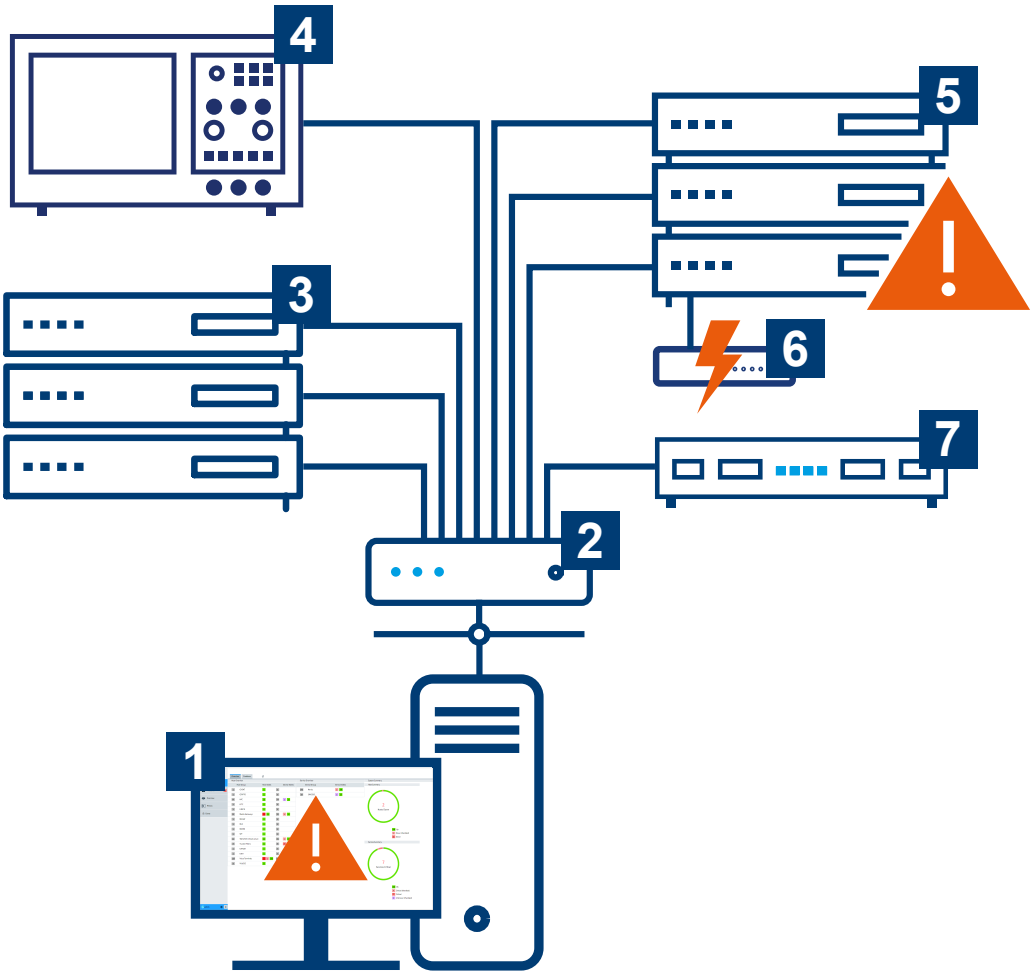


Figure 3-1: R&S CHM - status monitoring overview

1 = Computer with web-based user interface
2 = Network component (router, switch)
3 = Server hardware
4 = Rohde & Schwarz device
5 = Server hardware with error condition
6 = Uninterruptible power supply with error condition
7 = R&S CHM host that runs the status monitoring software

The R&S CHM software runs on a Linux server (**7**). You can access the web-based user interface on any standard computer in the network (**1**).

R&S CHM can fetch data from all connected and configured system components (**1** to **7**). Therefore, the operational state of the system is always under control. The downtime periods, due to maintenance operations or hardware failures, are reduced to a minimum.

Life of monitoring data

All monitoring data is retained for 90 days. Older data is purged from the database.

To monitor status information, system operators and administrators use the browser-based graphical user interface, in the following named as "web GUI".

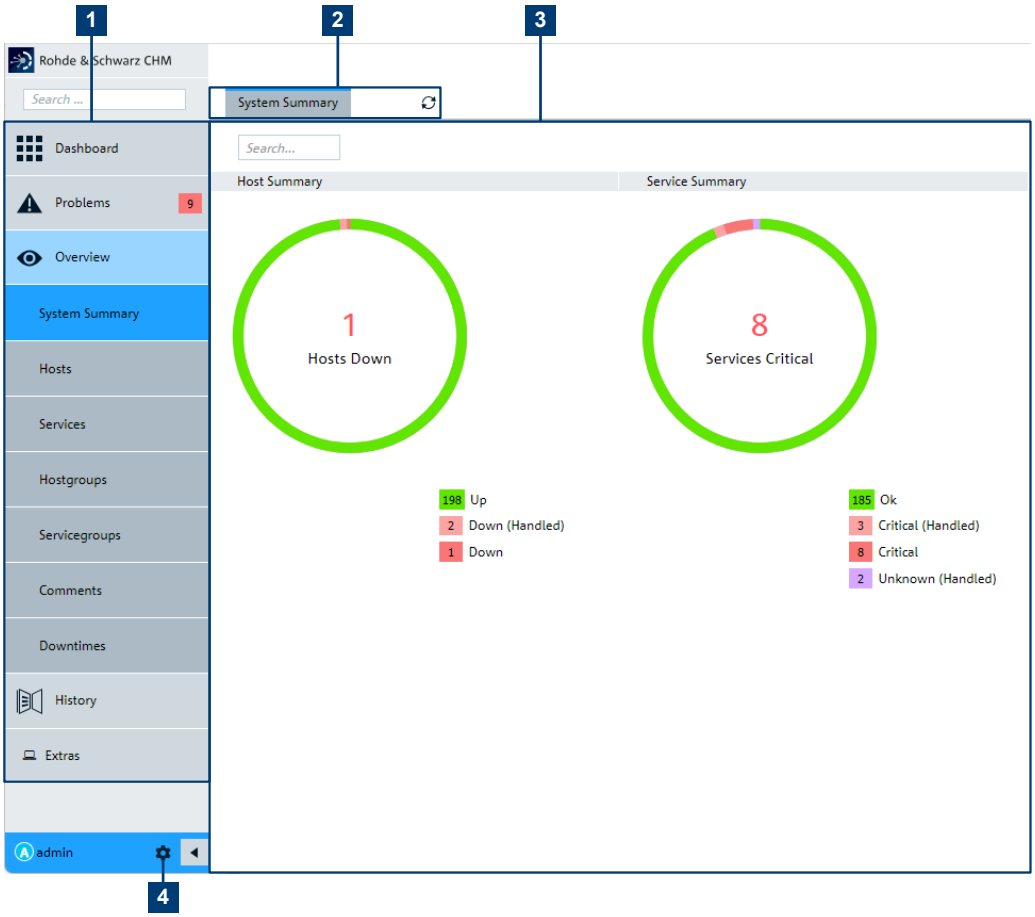


Figure 3-2: Web GUI for status monitoring

1 = Navigation and filter categories
2 = Additional filter categories
3 = Main area for status monitoring
4 = System-related pages and "Logout"

To configure the R&S CHM host from any computer in the LAN, system administrators can use an SSH client, such as PuTTY.

How to continue?

The next steps depend on your role as mentioned under "Target audience" on page 7.

- **Monitor system status information on the web GUI (operators and administrators)**
  Read the "R&S CHM System Status Monitoring" user manual.
- **Install and configure R&S CHM (system administrators, integrators)**
  These tasks address system administrators and software integrators:
  – Section 4, "Installing R&S CHM", on page 18
  – Section 5, "Deploying certificates", on page 31
  – Section 6, "Configuring status monitoring", on page 40
  – Section 7, "Configuring status checks", on page 134

Under Linux, run commands as `root` user or use `sudo`.

4 Installing R&S CHM

Software installation is divided into these main parts:

- **R&S CHM host installation**
  The R&S CHM host software runs on Linux. Use the Rohde & Schwarz lifecycle software manager (LCSM) for installation. If LCSM is not available, follow the description in Section 4.1, "Installing the R&S CHM host without LCSM", on page 19.
- **R&S CHM agent installation**
  The agent software runs on monitored Windows®- and Linux-based computers.
  – Section 4.2.1, "Installing Windows agents", on page 21
  – Section 4.2.2, "Installing Linux agents", on page 22

Before you start installation, review the minimum hardware and software requirements for the R&S CHM host and the agents.

Hardware and software requirements

You can install the R&S CHM host software on a server or a virtual machine (VM). Ensure that the R&S CHM host meets the minimum requirements listed in the following table. Keep in mind that the requirements increase with an increasing number of monitored system components and services.

Table 4-1: Requirements for R&S CHM hosts and Linux agents

Component	Minimum requirements
CPU	2 cores with 2 GHz
HDD	50 Gbyte
RAM	2 Gbyte Enable the swap partition for optimal system performance and stability.
LAN adapter	1 Gbit/s, RJ-45 connector
Operating system	Oracle Linux v8.x or later, optional with hardening according to DISA standard.

Table 4-2: Requirements for Windows agents

Component	Minimum requirements
CPU	2 cores with 2 GHz
HDD	50 Gbyte
RAM	2 Gbyte
Operating system	Windows 10 build 1809 and later

System time requirements

All devices in the system that you monitor need to be time-synchronized using NTP.

- Installing the R&S CHM host without LCSM 19
- Installing R&S CHM agents 20
- Installing R&S CHM clients 23
- Firewall 30

4.1 Installing the R&S CHM host without LCSM

The R&S CHM host runs on Oracle Linux. If you do not have a host running this operation system, we recommend downloading the Linux minimal version from the internet.

To install Oracle Linux

For comprehensive installation instructions of Oracle Linux, visit:

docs.oracle.com/en/operating-systems/oracle-linux/8/install/

This procedure only contains the main steps:

1. Visit the homepage: yum.oracle.com/oracle-linux-isos.html
2. Download the ISO image that suits the hardware architecture of your host, for example x86_64 for an Intel 64-bit server in version 8.x (or later).
3. Prepare the installation source.

   You can select from various options:
   - If you need a bootable physical media, prepare a DVD or a USB flash drive.
   - If you install Oracle Linux in a virtual machine, configure the virtual machine with at least the minimum requirements listed in Table 4-1. You can directly select the ISO image as the startup disk on your HDD.
   - If needed, you can also save the ISO image from a location on the network and boot it using NFS, FTP HTTP or HTTPS access methods.
4. Boot the installation media or ISO image.
5. Select "Install" in the boot menu and press [Enter].

   Anaconda, the Linux installer starts.
6. Follow the instructions on the screen.

   All installation options are properly configured, such as language, region, keyboard layout, date and time.
7. On the "INSTALLATION SUMMARY" screen, select "Begin Installation".

   Installation of Oracle Linux starts.

   Oracle Linux is installed on the host and ready for operation.

To install the R&S CHM host software

1. Perform a VM snapshot before you continue.

This measure lets you fall back to the fresh OS if you need to update the R&S CHM host software.

2. Ask your Rohde & Schwarz sales representative or applications engineer for providing the R&S CHM host software package.
3. Copy the `chm-<version>.tar.gz` archive to the R&S CHM host > `/root/`. For example, you can use WinSCP for this task.
4. Log in to the R&S CHM server, e.g. using SSH.
5. Change to the directory where the `chm-<version>.tar.gz` file resides.
6. Unpack the archive.

   ```
   tar xfvz chm-*.tar.gz
   ```

7. Change to the extracted `chm` directory:

   ```
   cd chm
   ```

8. Run the install script:

   ```
   ./install-chm-server
   ```

   Installation takes a while. Wait until the `Completed` message is shown.

   The R&S CHM host is up and running.
   Continue with Section 6.3, "Changing the configuration", on page 43.

To update the R&S CHM host

R&S CHM does not support update installations.

1. On R&S CHM hosts, back up the `chm.yaml` configuration file.
2. Reset the host to a VM snapshot before you install R&S CHM.
3. Install the new R&S CHM version as described previously.

4.2 Installing R&S CHM agents

The agent is a program that runs remotely on a Windows or Linux computer. It helps provide information to the R&S CHM host. Contained PowerShell modules are signed on Windows and the Rohde & Schwarz certificate is installed.

Obtaining installers

Ask your Rohde & Schwarz sales representative or applications engineer to provide the software package for R&S CHM Windows and Linux agents.

Once the agent is installed, configure the agent and integrate it with R&S CHM by following the instructions in Section 6, "Configuring status monitoring", on page 40.

4.2.1 Installing Windows agents

R&S CHM supports the AllSigned execution policy.

1. Check for the installation of the current certificates.

   The `CHM_Windows_Agent_<version>.exe` is signed via the Rohde & Schwarz code signing service. This fact means that the "DigiCert Trusted Root G4" is necessary. If this certificate, or an equally acceptable cross-signed one, is not present in the "Trusted Publishers" certificate store, the installer fails.

2. Check your hardening for potential issues.

   a) Is installation of Windows packages allowed? Open the registry editor ("regedit"). Check the following path:
   `Software\Policies\Microsoft\Windows\PowerShell`.

   b) Is Powershell script execution allowed? At least the following setting is necessary: "ExecutionPolicy": "AllSigned"
   Open the group policy editor ("gpedit").
   Open "Computer Configuration" > "Administrative Templates" > "Windows Components" > "Windows PowerShell".
   Enable the setting "Turn on script execution" and at least set it to "AllSigned".

3. Copy the `CHM_Windows_Agent_<version>.exe` installer to the Windows agent.

4. Run the `CHM_Windows_Agent_<version>.exe` installer.

   Note: If you need to monitor the CPU temperature of the Windows agent, select "Options" > "Install the PawnIO kernel driver".
   Also, you can run the installer from the command line with `WithPawnIo=1`.

5. Finish the `CHM_Windows_Agent_<version>.exe` installer.

6. If you install a Windows agent for gRPC-based R&S RAMON monitoring:

   a) Copy the `chmrd_<version>.msi` installer to the Windows agent.
   b) Run the `chmrd_<version>.msi` installer.

   The Windows agent is installed successfully.
   Continue with Section 5, "Deploying certificates", on page 31.

   See also:
   - Section 6.8, "Configuring R&S RAMON for monitoring", on page 78
   - Section 8.3, "Example configuration for R&S CHM Windows agents", on page 194
   - Section 9.4, "Troubleshooting installation problems on Windows agents", on page 197

To configure a user-defined location for the package cache (optional)

During installation of Windows software, the installers add files to a location named package cache on your PC. If necessary, you can change the location for R&S CHM software installers on a per-machine level using the Windows Registry Editor.

1. Select ⊞ "Start".

2. Type *registry editor*.
3. Select "Run as administrator".

   The Registry Editor opens.
4. Expand the "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\" key.
5. Create the following entries:
   a) Under "Policies", add the key "Wix".
   b) Under the "Wix" key, add the key "Burn".
      Resulting registry key:
      "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Wix\Burn"
   c) Under "Burn", add the string value "PackageCache".
   d) As a value, enter the path without using environment variables. For example, `C:\my_package_cache_location\chm`

   R&S CHM software installers now use the user-defined location as the package cache location.

4.2.2 Installing Linux agents

1. Perform a VM snapshot before you continue.

   This measure lets you fall back to the fresh OS if you need to update the R&S CHM host software.
2. Copy the `tar.gz` installer archive to the Linux agent.
3. Execute `tar xfvz xxx.tar.gz`.
4. Change to the extracted `chm` directory:
   `cd chm`
5. Execute `./install-chm-agent`

   The Linux agent is installed successfully.
6. Deploy the certificates as described in Section 5, "Deploying certificates", on page 31. Then, return to this procedure.
   a) Execute `systemctl status chm-agent` to check the status of the R&S CHM agent.
   b) Execute `systemctl restart chm-agent` to restart the R&S CHM agent.

See also: Section 8.4, "Example configuration for R&S CHM Linux agents", on page 195

4.2.3 Updating R&S CHM on agents and clients

To update R&S CHM on Linux agents, see "To update the R&S CHM host" on page 20.

To update R&S CHM on Windows agents and clients

1. On R&S CHM agents and R&S CHM clients, back up the configuration files. E.g., on R&S CHM hosts, the `chm.yaml` file and on R&S CHM clients, the `client_config.json` file.
2. Uninstall R&S CHM.
3. Install the new R&S CHM version.

4.3 Installing R&S CHM clients

An R&S CHM client (short: client) is an application to open the web GUI, including these additional features:

- Problem indication on a system tray icon .
- Autostart the application when logging on to the PC.
- Start the web GUI by using the tray icon or start the web GUI in a maximized window. No need to install an additional browser.

To get a client up and running

Summary of necessary tasks:

1. Installing the client software
2. Extending the chm.yaml
3. Connecting the client with the R&S CHM host
4. Handling certificates
5. Starting the client for the first time
6. Configuring application logging
7. Setting up SSO

Updating a client

If you need to update a client, see Section 4.2.3, "Updating R&S CHM on agents and clients", on page 22.

4.3.1 Installing the client software

R&S CHM supports the AllSigned execution policy.

1. Copy the `CHM_Client_<version>.msi` installer to a Windows agent or Windows PC.

2. Run the `CHM_Client_<version>.msi` installer.

   The client application is installed successfully.

4.3.2 Extending the chm.yaml

1. On the R&S CHM host, edit the `chm.yaml` file.
2. Enable the client interface and the check-logic.
   To do so, add the `system_state` key to the `checks` section of the R&S CHM host, here named `host1.de`.

```
hosts:
  - name: host1.de
    tags: [chm]
    checks:
      - ping:
      - system_state:
```

Figure 4-1: Code snippet - system_state check

See also: system_state on page 184

3. Specify the connection type for each client in the status monitoring system, e.g. the client named `host2.de`. The `connections: [client]` key ensures that the client can communicate with the R&S CHM host.

```
hosts:
  - name: host2.de
    connections: [client]
```

Figure 4-2: Code snippet - client connection type

   The client interface, the check logic and the client connection type are configured properly.

4.3.3 Connecting the client with the R&S CHM host

In addition to the previous configuration in the chm.yaml file, you configure the connection between the client and the R&S CHM host in a JSON configuration file on the client.

1. Create the `client_config.json` text file in one of the following locations:
   - **User-specific**
     `%appdata%\chm-client\client_config.json`
     For example:
     `C:\Users\Administrator\AppData\Roaming\chm-client\client_config.json`
   - **System-wide**

```
%programdata%\chm-client\client_config.json
```

Note: A client-specific configuration always takes precedence over a system-wide configuration. In this case, R&S CHM only considers the information that you provide in the client-specific configuration. A system-wide configuration is not considered.

2. Insert the following:

```
{
    "api": {
        "host": "<chm_host>"
    }
}
```

Figure 4-3: Client - JSON configuration file - minimal information

3. Substitute `<chm_host>` with the name of your R&S CHM host as specified in the `chm.yaml` file on the R&S CHM host. For example:

```
{
    "api": {
        "host": "host1.de"
    }
}
```

Figure 4-4: Client - JSON configuration file - example R&S CHM host name

4. Optional: Start the client in a maximized window, i.e. the web GUI in full screen, using `"window": { "start_maximized": true }`:

```
{
    "api": {
      "host": "host1.de"
    },
    "window": {
      "start_maximized": true
    }
}
```

Figure 4-5: Client - JSON configuration file - start the web GUI in a maximized window

Otherwise, the web GUI starts minimized. Users can open the web GUI using the client status icon in the Windows notification area.

4.3.4 Handling certificates

A client needs an SSL certificate for the secure communication with the R&S CHM host. Certificate usage depends on the implementation and certificate type.

Option 1: The client runs on a Windows agent

If the computer already runs the agent software, the same certificates are used by the client and found automatically.

See also: Section 5, "Deploying certificates", on page 31

Option 2: Using central PKI/central CA-signed certificates

If a central public key infrastructure (PKI) is used in your system and the certificates are generated and distributed via the central PKI, the client can use these certificates.

By default, the client checks for a valid certificate under `%appdata%\chm-client\` and in the certificate folder of the agent, see Section 5.2, "Using CA-signed certificates", on page 34.

The following files are necessary. The prefix <hostname> must exactly match the Windows OS host name of the machine, even if they have been initially created by using the FQDN of the machine as name:

- `<hostname>.key`
- `<hostname>.crt`

E.g. If the Windows host name is "SERVER-01", name the files `SERVER-01.key` and `SERVER-01.crt`.

If you want to use another certificate location, you can add this information to the JSON configuration file that you have created in Section 4.3.3, "Connecting the client with the R&S CHM host", on page 24.

Example:

JSON configuration file on the client. As a path separator, use double backslashes \\:

```
{
    "api": {
        "host": "host1.de",
        "client_crt": "C:\\temp\\someOtherCertificate.crt",
        "client_key": "C:\\temp\\someOtherCertificate.key"
    }
}
```

4.3.5 Starting the client for the first time

The following steps are only necessary if the HTTPS certificate, i.e. the CA root certificate, is not installed in the "Trusted Root Certification Authorities" store yet but in the "Intermediate Certification Authorities" store only and if the client application is installed here:

- On a stand-alone node (not on an agent).
- On an agent node where the SSO option is not configured in the configuration file of the client application.

1. Start the R&S CHM client application for the first time.

   Windows prompts you to import the HTTPS certificate.

2. Confirm the request to install the certificate.
3. Open the web GUI.

a) If necessary, select Windows notification area > "Show hidden icons".
b) Right-click > "Open".

4. On the log on page, enter your credentials.

   The client successfully connects to the R&S CHM host using the HTTPS protocol.

If you start the client without a valid configuration file, it automatically creates this file under `%appdata%\chm-client\`. Open this file and specify the right R&S CHM host name. See Section 4.3.3, "Connecting the client with the R&S CHM host", on page 24.

If there is no prompt request to import the HTTPS certificate at the initial R&S CHM client application start, double-check if the client has been started successfully without installing the HTTPS certificate. Also, ensure that with a stand-alone client the CA root certificate has got installed in the "Trusted Root Certification Authorities" store.

4.3.5.1 Configuring application logging

You can define to where a client logs to and the amount of logged information. To do so, specify an additional `"logging"` object in the JSON configuration file.

Example:

JSON configuration files on the client with optional `"logging"` settings.

(1) Console logging, warning log level 3

```
{
    "api": {
        "host": "host1.de"
    },
    "logging": {
        "logger": "Console",
        "log_level": 3,
     }
}
```

(2) Windows event logging, debug log level 1.

```
{
    "api": {
        "host": "host1.de"
    },
    "logging": {
        "logger": "EventLog",
        "log_level": 1,
     }
}
```

Log types

- `"logger": "Console"`
  Recommended logging method. Logs everything on the console in which the client is started (default). If started using the `*.exe`, the logs go unnoticed.

- `"logger": "File"`
  Logs everything in a file, without file rotation. By default, the file is located here if you do not specify the file path:
  `%appdata%\chm-client\chm_client_log.txt`
  If you specify a different file location, use double backslashes (\\) as path separator:
  `"file_path": "<drive>\\<folder>\\LogFileName.txt".`
- `"logger": "EventLog"`
  Logs "CHM client" events in the Windows Event Viewer > "Windows Logs" > "Application" ("Information", "Warning", "Error").

Log level

The log level `"log_level": <number>` specifies the amount and type of logged information. The levels meet the Microsoft log level standard.

Table 4-3: Log level overview

Log level	Meaning/description	
0	Trace	Logs contain the most detailed messages (default). These messages can contain sensitive application data. These messages are disabled by default. Never enable them in a production environment.
1	Debug	Logs are used for interactive investigation during development. These logs primarily contain information useful for debugging and have no long-term value.
2	Information	Logs track the general flow of the application. These logs have long-term value.
3	Warning	Logs highlight an abnormal or unexpected event in the application flow, but do not otherwise cause the application execution to stop.
4	Error	Logs highlight when the current flow of execution is stopped due to a failure. These messages indicate a failure in the current activity, not an application-wide failure.
5	Critical	Logs describe an unrecoverable application or system crash, or a catastrophic failure that requires immediate attention.
6	None	Not used for writing log messages. Specifies that a logging category does not write any messages.

4.3.6 Setting up SSO

To allow the client to use single sign-on, specify an additional `"browser"` object in the JSON configuration file.

1. Open the `client_config.json` text file in one of the following locations:
   - **User-specific**
     `%appdata%\chm-client\client_config.json`
     For example:
     `C:\Users\Administrator\AppData\Roaming\chm-client\client_config.json`
   - **System-wide**

```
%programdata%\chm-client\client_config.json
```

2. Insert the SSO configuration options as listed in Table 4-4. For an example, see Example "SSO browser configuration" on page 29.
3. If necessary, you can also add other configuration options as listed in the following table.

Table 4-4: SSO-specific browser configuration options

Configuration	Explanation	Example value	Default value
auth_negotiate_delegate_whitelist	A comma-separated list of servers for which integrated authentication is enabled. Then any URL ending with the given servers is considered for integrated authentication. Without the * prefix, the URL must match exactly.	"auth_negotiate_delegate_whitelist": "*.example.com, *.foobar.com, *.baz"	NULL
auth_server_whitelist	A comma-separated list of servers for which delegation of user credentials is required. Without the * prefix, the URL has to match exactly.	"auth_server_whitelist": "*.example.com, *.foobar.com, *.baz"	NULL

Example: SSO browser configuration

Add the following lines to the `client_config.json` file. Substitute the server examples accordingly.

```
{
  "browser": {
    "auth_negotiate_delegate_whitelist": "*.example.com, *.foobar.com, *.baz"
    "auth_server_whitelist": "*.example.com, *.foobar.com, *.baz"
  }
}
```

You can configure additional browser settings as necessary.

Table 4-5: Additional browser settings

Configuration	Explanation	Example value	Default value
disable_renderer_backgrounding	Prevent Chrome from lowering the priority of invisible pages' renderer processes. This flag is global to all renderer processes, if you only want to disable throttling in one window, you can take the hack of playing silent audio.	"disable_renderer_backgrounding": "true"	False
enable_logging	Prints Chrome's logging to stderr or a log file. For more information, see www.chromium.org/for-testers/enable-logging/	"enable_logging": "true"	False

Configuration	Explanation	Example value	Default value
lang	Set a custom locale.	"lang": "en_US"	SystemDefault
ignore_certificate_errors	Ignores certificate-related errors. **Note**: This setting removes the check for certificate validation. This measure reduces the overall system security and can lead to man-in-the-middle and similar attacks.	"ignore_certificate_errors": "true"	False

4.4 Firewall

The firewall rules are included in the software installer and thus set automatically. The following table informs about necessary connections.

Table 4-6: Firewall rules

Connection	Port	Protocol	Use case
R&S CHM host → R&S CHM host	4656	TCP	Transfer of system or host group summary states between R&S CHM hosts over a trusted filter device
R&S CHM host → SNMP monitored device	161	SNMP	Collect monitoring information
Maintenance PC → R&S CHM node	22	SSH	Optional SSH connection
PC → R&S CHM node	80	HTTP	Viewing R&S CHM website in the browser (redirection to HTTPS)
PC → R&S CHM node	443	HTTPS	Viewing R&S CHM website in the browser (encrypted connection)
R&S CHM node → VMWare ESXi/vCenter	443	HTTPS	Monitoring of VMWare ESXi/vCenter status
Monitored item Windows/Linux → R&S CHM node	5665	Icinga	Encrypted communication of Icinga (monitoring information)
PC with R&S CHM client → R&S CHM node	5665	Icinga	API connection
Monitored item Windows/Linux ←→ R&S CHM node	18005	Grpc	Encrypted communication of R&S CHM (monitoring and control information)

5 Deploying certificates

Certificates protect the connections between the R&S CHM host and the R&S CHM agents. Without certificates, R&S CHM cannot monitor the system state of connected R&S CHM agents. Thus, we recommend deploying certificates on all R&S CHM agents.

The following figure serves as a system configuration example. This configuration is used in the following procedures.

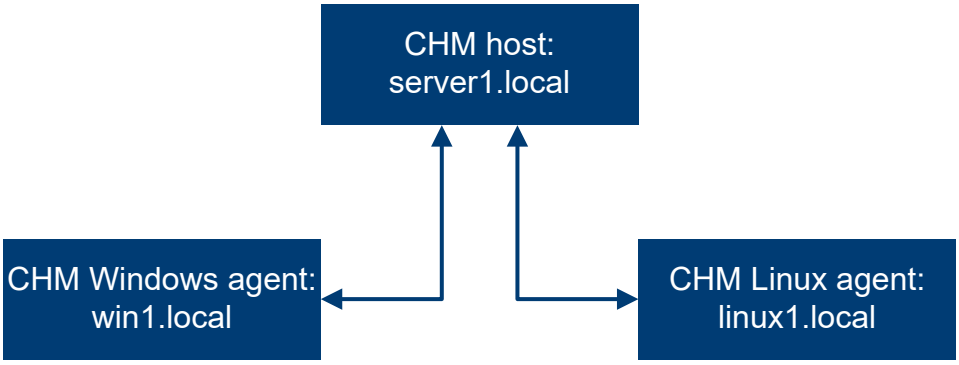


Figure 5-1: Example R&S CHM system

R&S CHM uses transport layer security (TLS) encryption to secure the communication between the R&S CHM host and the R&S CHM agents. By default, certificates are self-signed. Self-signed certificates are renewed automatically.

Also, you can use certificates that are provided by a central certificate authority (CA). If you want to use certificates from a central CA, contact your certificate manager. Self-signed certificates and a CA are generated automatically on the R&S CHM host during software installation.

Change all certificates before your system goes live.

- Using the R&S CHM built-in CA....31
- Using CA-signed certificates....34
- Removing certificates....34
- Configuring a user-defined certificate location on Windows hosts....35
- Deploying certificates for distributed monitoring....35

5.1 Using the R&S CHM built-in CA

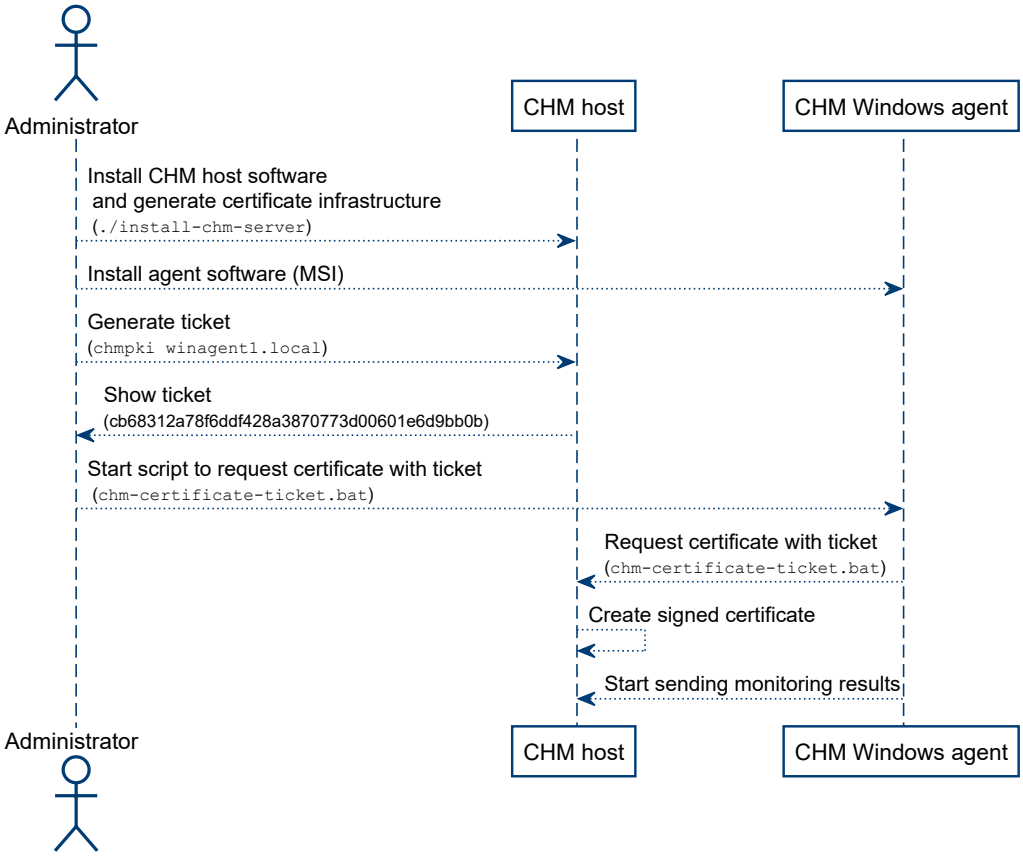
You can use the R&S CHM built-in CA as follows:

- With pregenerated tickets, see "To deploy certificates with tickets" on page 32.
- With certificate signing requests (CSR), see "To deploy certificates with signing request" on page 33.

As a prerequisite for creating certificates, the R&S CHM host must be installed and online.

To deploy certificates with tickets

The following figure shows the general workflow if you use the R&S CHM built-in CA with tickets.



1. Log in to the shell of server1.local using ssh.
2. Execute `chmpki win1.local`.
   win1.local must be the FQDN of the windows agent.

   A generated ticket is shown.
3. Note down that ticket.
4. Connect to the windows host.
5. Create certificates:
   - **On Windows**, run this batch file as an administrator:
     `%programfiles%\chm\chm-certificate-ticket.bat`
   - **On Linux**, issue this command:
     `chm_certificate_ticket`

   The script prompts you for the server you want to connect to.

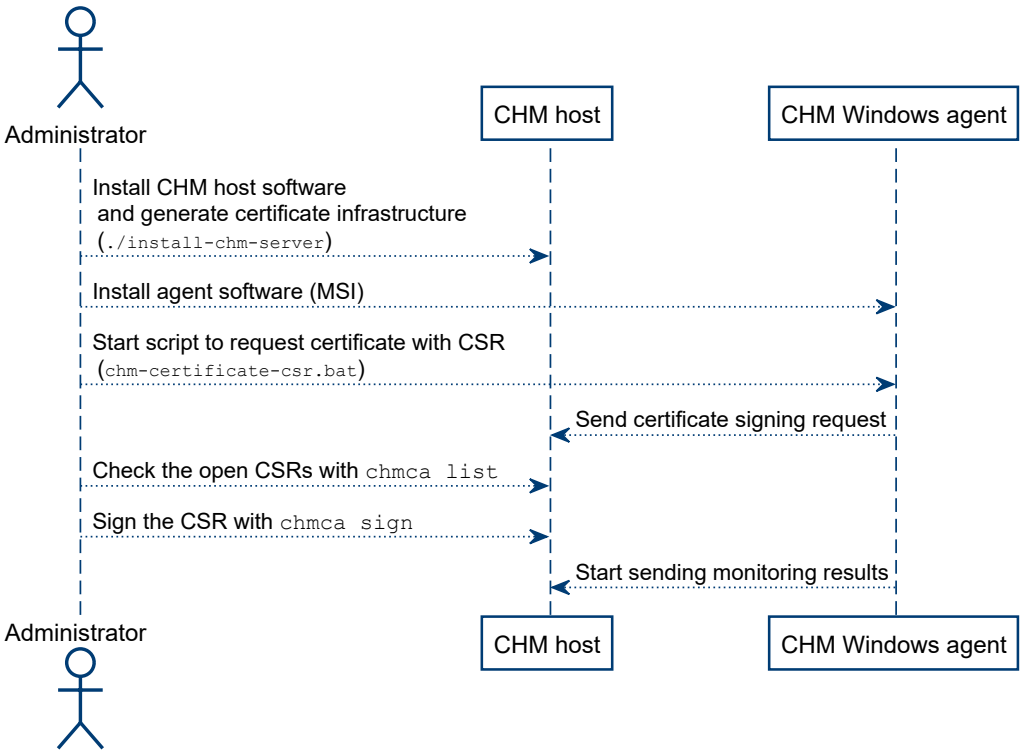
6. Enter `server1.local` and the ticket identifier.

   The script creates the necessary certificates and configuration.

   If necessary, you can call the script with command-line arguments to execute it silently:
   - **On Windows**, run the batch file with parameters, all on one line:
     `chm-certificate-ticket.bat server1.local cb68312a78f6ddf428a3870773d00601e6d9bb0b`
   - **On Linux**, run this command with parameters, all on one line:
     `chm_certificate_ticket server1.local cb68312a78f6ddf428a3870773d00601e6d9bb0a`

To deploy certificates with signing request

The following figure shows the general workflow if you use the R&S CHM built-in CA with certificate signing request (CSR).



1. Send the signing request.
   - **On a Windows agent**
     Execute `%programfiles%\chm\chm-certificate-csr.bat`
   - **On a Linux agent**
     Execute `chm-certificate-csr`

   The script prompts you for the server you want to connect to.

2. Type in `server1.local`.

   The script requests the certificate at the R&S CHM host and generates it.

3. Log in to the server via ssh.
4. Execute `chmca list`.

   All signing requests are shown, e.g.

```
Fingerprint                                                      | Timestamp           | Signed | Subject
-----------------------------------------------------------------|---------------------|--------|--------
403da5b228df384f07f980f45ba50202529cded7c8182abf96740660caa09727 | 2021/09/06 17:02:40 | *      | CN = win1.local
71700c28445109416dd7102038962ac3fd421fbb349a6e7303b6033ec1772850 | 2021/09/06 17:20:02 |        | CN = win2.local
```

5. Execute this command to approve the sign request from, e.g. win1.local.
   `chmca sign 403da5b228df384f07f980f45ba50202529cded7c8182abf96740660caa09727`

   Note: Ensure that the timestamp and the "CN" under the subject are correct to ensure that only valid requests are signed.

   The Windows agent can send its monitoring results to the R&S CHM host.

5.2 Using CA-signed certificates

As an alternative to the R&S CHM built-in CA, your company can use private certificate authorities to issue certificates for your internal servers.

We recommend using the following naming conventions:

- Certificate of the root CA: `ca.crt`
- Certificate of the server: `<fqdn>.crt`, where `fqdn` is the fully qualified domain name (FQDN).

1. Obtain the certificates from your certification authority.
2. Copy the certificates to these locations:

System component	Location
R&S CHM host	`/var/lib/icinga2/certs/`
Windows agent	`%programdata%\icinga2\var\lib\icinga2\certs\`
Linux agent	`/var/lib/icinga2/certs/`

5.3 Removing certificates

If necessary, you can remove all certificates on the R&S CHM host and the agents.

If you remove the certificates, system status monitoring is no longer possible.

► Execute these commands:

- **On the R&S CHM host**: `chm_clean_certificates`
- **On Windows agents**:
  `%programfiles%\chm\chm-clean_certificates.bat`
- **On Linux agents**: `chm_clean_certificates`

5.4 Configuring a user-defined certificate location on Windows hosts

You can configure a common folder with all the needed certificates and make Icinga use this folder instead of the default folder `C:\ProgramData\icinga2\var\lib\icinga2\certs\`. To reach this goal, we need a symbolic link that points to the common folder with the certificates.

To create the symbolic link

1. If certificates are already installed in the default folder, move them to the new location, i.e. the `Target` folder, e.g. `C:\Certificates`.
2. If existing, remove the (now empty) default folder:
   `C:\ProgramData\icinga2\var\lib\icinga2\certs`
3. Use one of these methods to create the symbolic link:
   - **In a Command Prompt window**
     Syntax:
     ```
     mklink /D Link Target
     ```
     Example:
     ```
     mklink /
     D "C:\ProgramData\icinga2\var\lib\icinga2\certs" "C:\
     Certificates"
     ```
   - **In the PowerShell**
     Syntax:
     ```
     New-Item -Path LINK -ItemType SymbolicLink -Value TARGET
     ```
     Example:
     ```
     New-Item -Path "C:\ProgramData\icinga2\var\lib\icinga2\
     certs" -ItemType SymbolicLink -Value "C:\Certificates"
     ```

5.5 Deploying certificates for distributed monitoring

You can select from the following deployment methods for certificates.

High availability (HA) masters cannot use the R&S CHM built-in CA.

- External CA.....36
- R&S CHM built-in CA.....36

5.5.1 External CA

Add the following for every R&S CHM instance, i.e. master, satellite and agent, to the directories listed in Section 5.2, "Using CA-signed certificates", on page 34:

- A copy of the root certificate.
- Its own certificate signed by the CA.
- Its own private key that corresponds to the signed certificate.

5.5.2 R&S CHM built-in CA

You can use the R&S CHM built-in certificate authority (CA) to deploy certificates in distributed setups. We demonstrate deployment using a multilevel setup with one master, one satellite and one windows agent. The following YAML code snippet describes this setup.

```
subsystems:
        - name: "master"
          hosts: [ "master.local.net" ]
        - name: "satellite"
          hosts: [ "satellite.local.net" ]
          parent_subsystem: "master"
        - name: "agent"
          hosts: [ "agent.local.net" ]
          parent_subsystem: "satellite"

hosts:
  - name: "master.local.net"
    tags: [ chm ]
    checks:
            - dummy:
  - name: "satellite.local.net"
    connections: [ "icinga2_linux" ]
    checks:
            - dummy:
  - name: "agent.local.net"
    connections: [ "icinga2_win" ]
    checks:
            - dummy:
```

5.5.2.1 Deploying certificates with signing requests

The following procedures assume that you already have added and deployed the previous YAML code to master and satellite.

To deploy certificates on the master

1. Clean existing certificates.
   `chm_clean_certificates --force`
2. Generate certificates for the master.
   `<CHM_install_folder>/install-chm-server --certonly`
3. Restart the CHM service.
   `systemctl restart chm`

To deploy certificates on the satellite

1. Clean existing certificates.
   `chm_clean_certificates --force`
2. Set up the satellite node.
   `chm_node_setup --subsys satellite --parent-subsys master --parent-chm master.local.net`
3. On the master, find and sign the satellite CSR.
   a) List pending certificate signing requests.
      `chmca list`
   b) Sign the satellite CSR. Example:
      `chmca sign 4415713a191055bf6941a450c6282fdc47838f8a7f0d863fb62b8b4d8baa1a88`
4. Restart the CHM service on the satellite.
   `systemctl restart chm`

To deploy certificates on the agent

Run the following steps in an elevated Windows PowerShell session in the R&S CHM installation directory `C:\Program Files\chm`.

On Linux agents, you can use the equivalent commands.

1. Clean existing certificates.
   `.\chm-clean-certificates.bat --force`
2. Set up the agent node
   `.\chm-node-setup.bat --subsys agent --parent-subsys satellite --parent-chm satellite.local.net`
3. On the master, find and sign the agent CSR.

a) List pending certificate signing requests:
`chmca list`
b) Sign the agent CSR. Example:
`chmca sign 56187c00f8b078dad6a18b03cf16a5c5899e8f9b3db4fa525b2bec872582d8c5`

4. Restart the agent services: `Restart-Service -Name "icinga2"`
`Restart-Service -Name "icingapowershell"`

5.5.2.2 Deploying certificates with tickets

The following procedures assume that you already have added and deployed the previous YAML code to master and satellite.

To deploy certificates on the master

1. Clean existing certificates.
`chm_clean_certificates --force`
2. Generate certificates for the master.
`<CHM_install_folder>/install-chm-server --certonly`
3. Restart the CHM service.
`systemctl restart chm`
4. Generate the tickets.
   - For the satellite.
   `chmpki satellite.local.net`
   Example ticket: `8414176ded78a5287af4baefba82a59100c1d4b6`
   - For the agent.
   `chmpki agent.local.net`
   Example ticket: `27272172a33d7865d335af7bafce5a635baf9d5d`

To deploy certificates on the satellite

1. Clean existing certificates.
`chm_clean_certificates --force`
2. Set up the satellite node with the ticket from the master.
`chm_node_setup --subsys satellite --parent-subsys master --parent-chm master.local.net --ticket 8414176ded78a5287af4baefba82a59100c1d4b6`
3. Restart the CHM service.
`systemctl restart chm`

To deploy certificates on the agent

Run the following steps in an elevated Windows PowerShell session in the CHM installation directory `C:\Program Files\chm`.

1. Clean existing certificates.
   ```
   .\chm-clean-certificates.bat --force
   ```
2. Set up the agent node with the ticket from the master.
   ```
   .\chm-node-setup.bat --subsys agent --parent-subsys
   satellite --parent-chm satellite.local.net --ticket
   27272172a33d7865d335af7bafce5a635baf9d5d
   ```
3. Restart the agent services.
   ```
   Restart-Service -Name "icinga2"
   Restart-Service -Name "icingapowershell"
   ```

6 Configuring status monitoring

Here, you can find all steps that are necessary to configure R&S CHM for system status monitoring. All data is contained in an editable configuration file. The configuration file is written in YAML v1.2 notation standard.

YAML is a human readable data serialization language for all programming languages. YAML is a case-sensitive language. It uses indentation with one or more spaces to represent the structure. Dashes (-) are used to represent the sequences (lists) and colons (:) are used to represent key-value pairs. The upper part of the configuration file on the R&S CHM host gives you an impression how this language looks like.

```
hosts:
  - name: host1.de
    displayname: CHM host
    tags: [chm]
    authentication:
      monitoring:
        - ldap:
            server: ldapserv.ourlocal.net
            port: 35636
            encryption: ldaps
            base_dn: ou=ldap_users,dc=ldapserv,dc=ourlocal,dc=net
            user_class: user
            user_name_attr: sAMAccountName
            bind_dn: service_user
            bind_pwd_path: ldap/service_user
    authorization:
[...]
```

Related information

- For more information about YAML, see Section 6.1, "Introduction to the YAML syntax", on page 41.
- For a YAML syntax reference, see the YAML website at https://yaml.org/refcard.html.

The following sections provide more details on R&S CHM configuration.

- Introduction to the YAML syntax 41
- Understanding aggregated states 42
- Changing the configuration 43
- Configuring hosts 44
- Configuring web GUI users 61
- Configuring R&S CHM features 71
- Managing password identifiers 76
- Configuring R&S RAMON for monitoring 78
- Configuring graphical system views (maps) 83
- Configuring the SNMP upstream interface 88
- Configuring generic monitoring of SNMP third-party devices 90
- Configuring distributed monitoring 108

- Configuring automatic system control 122
- Using common keys 125
- Using frequent keys 128

6.1 Introduction to the YAML syntax

The YAML syntax contains different kinds of data blocks:

- A **sequence** with values that are listed in a specific order. The sequence starts with a dash and a space, e.g. `- ping`.
- A simple **mapping** between key and value pairs. A key must be unique; the order does not matter.

A third type is called **scalar**, which is arbitrary data, such as strings, integers.

Data blocks can be written in block style or flow style.

Example: Sequence data blocks

A list of items in block style.

```
checks:
  - ping:
  - os_memory:
  - os_process:
```

A list of items in flow style.

```
host: [ping , os_memory , os_process]
```

Example: Mapping data blocks

```
snmp_connection:
  port: 161
  version: 2
  community: public
```

Example: Dictionary

This data block is a more complex collection of `key: value` pairs. Each pair can be nested with numerous options.

```
hosts:
  - domainname: chm-host.domain.net
    connections: [local]
    tags: [chm]
    hostgroups: [germany, bavaria]
    checks:
      - icinga2_cluster:
      - dhcp:
      - dns:
```

Table 6-1: Indicator characters - excerpt from the YAML syntax

Collection indicators	
:	Value indicator. In threshold configurations, the colon (:) indicates the edges of the interval, see also `thresholds` on page 132
-	Nested series entry indicator.
,	Separate in-line branch entries.
[ ]	Surround in-line series branch.
{ }	Surround in-line keyed branch.
Misc indicators	
#	Throwaway comment indicator.

Use single quotes (`' '`) in YAML if your string value includes special characters. For example, you possibly need single quotes around strings that contain these special characters:

`{, }, [, ], ,, &, :, *, #, ?, |, -, <, >, =, !, %, @, \.`

For details, see the YAML specification at https://yaml.org/spec in version v1.2.

This user documentation uses the identifier ***RST** to indicate the default values in the descriptions of the YAML keys.

6.2 Understanding aggregated states

Besides the individual status checks with their individual states, R&S CHM provides a state aggregation logic for the whole system and for host groups. The summarized states are accessible to different system components. Services that are acknowledged or in a downtime are excluded from state calculation. For more information about handling issues ("Acknowledge", "Schedule downtime"), see the "R&S CHM System Status Monitoring" user manual or help.

System state

The system state is defined as the worst state of all status checks within the system. The following figure shows the logic behind the system state and shows the components that use the system state. For example, the SNMP upstream interface, the gp2pp server check over an R&S trusted filter and the R&S CHM client can process the system state.

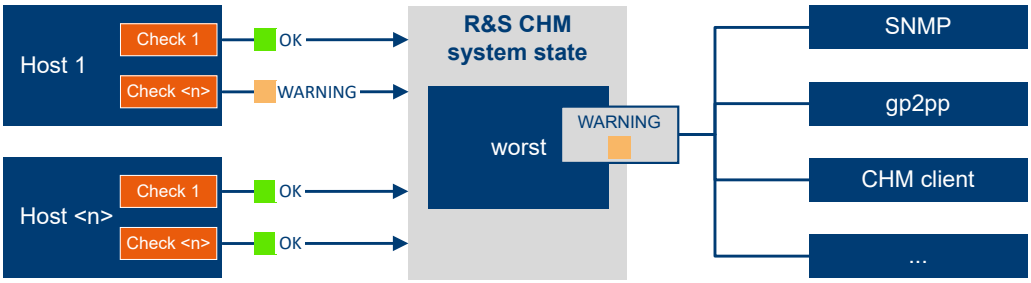


Figure 6-1: Aggregated state of hosts and checks

Host group state

Also, R&S CHM provides an aggregated host group state. The host group state is defined as the worst status of all status checks within that host group. For example, the SNMP upstream interface and the gp2pp server check over an R&S trusted filter can process the host group state.

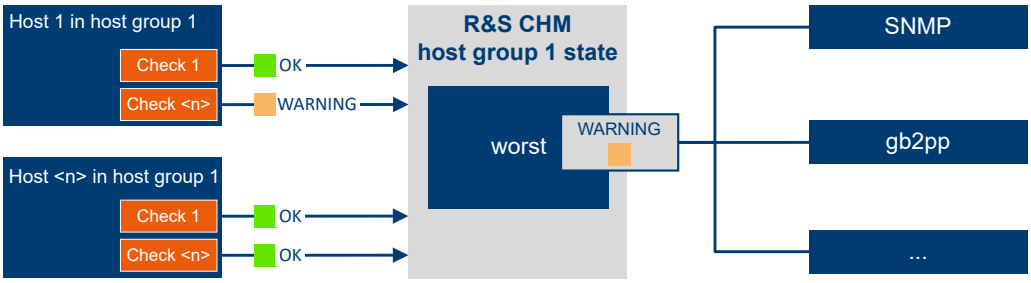


Figure 6-2: Aggregated state of hosts that belong to a host group

See also:

- Section 6.10, "Configuring the SNMP upstream interface", on page 88
- gb2pp on page 152

6.3 Changing the configuration

All configurations are defined in a single configuration file, which is the central configuration file for all objects that you want to monitor in the network.

To access the configuration file

► On the R&S CHM host, you can find the configuration file here:
`/etc/opt/rohde-schwarz/chm/chm.yaml`

You can edit the file locally. Alternatively, you can transfer the configuration file to another PC, e.g. using WinSCP with SFTP or FTPS protocols. If finished, transfer it back to its original location on the R&S CHM host.

To edit the configuration file

1. Open the chm.yaml file in an editor.
   - On the local R&S CHM host, you can use the vi editor:
     `vi chm.yaml`
   - On a remote Windows host, you can use Windows Notepad or a more comfortable text editor with YAML syntax highlighting, e.g. Notepad++.
2. In the editor, navigate to the sequence item.
3. Add the key-value pairs.
4. Save the file.
5. If necessary, transfer the file back to its location on the R&S CHM host (`/etc/opt/rohde-schwarz/chm/chm.yaml`).
6. Restart this service on the R&S CHM host to take the changes effect:
   `systemctl restart chm`

   R&S CHM checks the syntax. If the syntax checks failed, edit the configuration file again and correct all syntax errors.

   If the syntax check was successful, the changes are applied. For example, you can monitor newly configured services on the web GUI.

Check if this service is running:
`systemctl status chm`

6.4 Configuring hosts

Here, you find detailed information on host configuration, including host-specific keys in the `chm.yaml` file.

A **host** is an independent device in the system. It is addressed and monitored by R&S CHM. For example, a host is a Windows PC, a Linux virtual machine or a device that you monitor using SNMP.

Hosts are characterized by several attributes, and several **checks** are subordinated to them. Each check verifies the host for an intended status, e.g. available disk space, temperature or other hardware status.

hosts....45
dashboards....48
widgets....49
exports....50
logic....52
logging....57
webinterface_url....60
webserver....61

hosts (Hosts)

The `hosts` dictionary consists of a list of all host elements for the whole system to be monitored.

Here, you specify the configuration and the checks for all hosts where R&S CHM is installed or that are monitored by R&S CHM.

Parameters:

name — string

Name of the host, i.e. the name of the R&S CHM host, a R&S CHM agent or an SNMP device that is monitored. A host instance always starts with the `name` key.
The first host instance in the file always denotes an **R&S CHM host**.

displayname — string

Shows this name on the web GUI instead of the specified `name` (optional).
There is no explicit limit of the string length. R&S CHM supports at least 1024 characters.
R&S CHM supports at least the following special charactes:
`!@#$%^&*()_+=-`~[]\\{}|;':\",./<>?äöüÄÖÜß€„“»«§°²³µ`

dashboards — Configures the contents of the "Dashboard". See `dashboards` on page 48.

features — string

Lets you configure additional R&S CHM features. See Section 6.6, "Configuring R&S CHM features", on page 71.

notes — string

Specifies a text snippet for hosts and services (optional). You can add a detailed description of the location or details on how to handle errors. Use the `<br>` tag to write a multi-line note.
R&S CHM shows this text snippet on the web GUI > "Service"/"Host" tab > "Problem handling".

tags — [chm] | [icinga2_ha]

Assigns a role to a host in the status monitoring system.

[chm]
Assigns the role "master" to an exclusive R&S CHM host or the role "primary master" to one of the R&S CHM hosts for a high-availability status monitoring configuration. For more information about the roles, see Section 6.12, "Configuring distributed monitoring", on page 108.
An R&S CHM host that is tagged with `[chm]` starts a monitoring system in which all hosts are synchronized regarding monitoring state. All hosts that are specified beneath are part of this monitoring system.
The next R&S CHM host instance tagged with `[chm]` starts the next monitoring system, and so forth. In combination with `exports`, you can configure multiple monitoring systems. These hosts are not synchronized, because the R&S CHM hosts are separated from each other, e.g. by a security gateway.

[icinga2_ha]
Assigns the role "secondary master" to a second R&S CHM host in a high-availability status monitoring system. All hosts that are tagged `[icinga2_ha]` belong to the same overall status monitoring system as the associated "primary master". Both R&S CHM hosts, primary master and secondary master, are fully synchronized regarding monitoring state.
For usage scenarios, see Section 6.12.1, "Configuring high availability monitoring", on page 109 and Section 6.12.4, "Configuring multi-level HA monitoring", on page 118.

logic
Combines status values from multiple checks to a single, aggregated status value. See `logic` on page 52.

logging
Configures the severity and the facility for event logging on the R&S CHM host. See `logging` on page 57.

exports
Configures an R&S CHM host so that it sends status monitoring information to another R&S CHM host (optional). See `exports` on page 50.

authentication
Configures LDAP-based user authentication. See `authentication` on page 64.

authorization
Configures user authorization. See `authorization` on page 68.

webinterface_url
string
Configures a hyperlink to the management web interface of the host. See `webinterface_url` on page 60.

connections
[local] | icinga2_win] | [icinga2_linux] | [snmp] | [client] | [gb2pp]
Defines how R&S CHM communicates with this host.

[local]
If you configure an operating system-dependent check on a master, specify `[local]`. This setting ensures that the right check plugin is used for all checks that depend on the operating system (Windows or Linux). For example, `load` is such a check.

[icinga2_win]
Denotes Windows agent. The checks run on this agent. The agent sends the check results to the master.

[icinga2_linux]
Denotes a Linux agent. The checks run on this agent. The agent sends the check results to the master.
Check plugin for Linux agents and satellites in multi-level monitoring configurations.

[snmp]
Denotes a host that is monitored by the R&S CHM host by using SNMP. The host is not installed on an R&S CHM agent. All checks are performed on the R&S CHM host and the check results are obtained by active checks.

[client]
Denotes an R&S CHM client. Specify `[client]` for a host that runs the R&S CHM client application.
How to: Section 4.3, "Installing R&S CHM clients", on page 23

[gb2pp]
Denotes that this R&S CHM host is configured as a gb2pp server.
For the necessary checks and for examples, see `gb2pp` on page 152.

checked_by — string

Specifies the R&S CHM instance that monitors this host (optional). You can specify this key for all hosts that are not a master, a satellite or an agent. For an example, see Example "YAML configuration: multi-level HA monitoring" on page 120.
Without this key, the default applies. Hence, hosts are monitored by default by the R&S CHM instance that is located in the same subsystem and that is not configured as a high availability host.

hostgroups — [comma-separated list of strings]

List of groups the host belongs to. The groups help identify the host on the web GUI.

snmp_connection — Activates the SNMP upstream interface (optional).
How to: Section 6.10, "Configuring the SNMP upstream interface", on page 88

webserver — Specifies the path to a TLS certificate and corresponding private key on R&S CHM hosts and satellites.
How to: `webserver` on page 61

checks

Parent key for all status checks. The first check is always a host check that checks if the host is reachable. All following checks are service checks. These checks provide information about the health states of the checked resources. See Section 7, "Configuring status checks", on page 134.

For detailed R&S CHM host configuration examples, see Section 8.1, "R&S CHM host configuration", on page 191 and Section 8.2, "Linux host configurations", on page 192.

Example:

Single R&S CHM host configuration with some high-level keys.

```
hosts:
  - name: host1.de
    displayname: CHM master
    tags: [chm]
    logic:
      aggregation1:
        function: worst
        ins: [input1, input2, input 3]
    logging:
      severity: info
      facility: local0
    authentication:
    authorization:
    webinterface:
    connections: [icinga2_linux]
    hostgroups: [monitoring, control]
    checks:      # The checks for this host
```

dashboards (Main elements on the web GUI > "Dashboard")

Configures the start page of the web GUI, the "Dashboard" (**1**). You can configure individual dashboard tabs (**2**) and the widgets (**3**) on these dashboards.

The `dashboards` key is optional. If it is not specified, the web GUI shows the default dashboards "Overview" and "Problems".

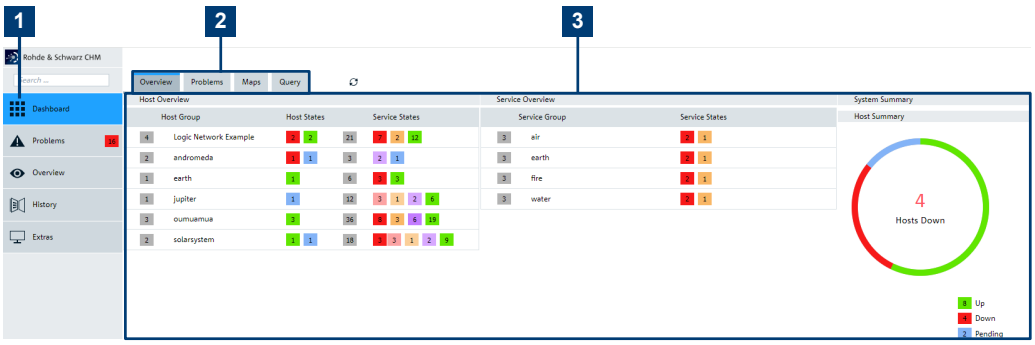


Figure 6-3: System-specific dashboard configuration

1 = "Dashboard" menu
2 = Multiple configured dashboard tabs
3 = Multiple configured widgets

Related parameters

- `Graphical system view (maps)` on page 86
- `widgets` on page 49

Parameters:

name — string

Label of a dashboard tab (**2**). The `name` must not contain dots (`.`).

widgets — The areas on an individual dashboard (**3**). For details, see `widgets` on page 49.

Example:

```
hosts:
  # First host list entry. (1)
  - name: host1.de
    displayname: CHM host
    dashboards:
      - name: Maps
        widgets:
          - name: Overview Map
            content: "Map: Overview1"
          - name: A custom query
            content: "Query: filter_pattern" # (2)
          - content: Host Problems - unhandled
      - name: Overview
        widgets:
          - content: Service Problems - unhandled
          - name: Overwrite the name with a custom name
            content: Host Problems - unhandled
```

(1) By convention the first hosts entry is the R&S CHM host.

(2) In the above example, the *`filter_pattern`* looks like this:

`monitoring/list/hosts?hostgroup_name=andromeda&sort=host_severity`

---

widgets (Areas on a dashboard)

Configures the areas of an individual dashboard. An individual widget consists of a name and content.

Related parameters

- `dashboards` on page 48
- `Graphical system view (maps)` on page 86

Parameters:

name — string

Heading of an area on a dashboard tab (optional for `content: <preset_value>`). If you configure the `name` in combination with `content: <preset_value>`, this configuration results in a user-defined heading that overwrites the default heading.
The `name` is mandatory in combination with `content: "Map: <map_name>"` and `content: "Query: <pattern>" Query`).
The `name` must not contain dots (`.`).

content — Host Overview | Service Overview | Status View | System Summary | Recent Events | Host Problems - unhandled | Service Problems - unhandled | "Map: <map_name>" | "Query: <pattern>"

Contents of the widget.

Map: <map_name>
Name of the map as specified in `Graphical system view (maps)` on page 86 > `name`. Due to the colon (`:`), enclose the whole string in quotation marks.

Query: <filter_pattern>
Query pattern. Due to the colon (`:`), enclose the whole string in quotation marks.

Example: Filter for a host name "Master" in the "Hosts" view:
`monitoring/list/hosts?host=%2AMaster%2A`

`%2A` is the HTML code for the asterisk (*) as a universal placeholder in a filter pattern.

Example: Filter for a host group name "andromeda" in "Hostgroups" view:
`monitoring/list/hosts?hostgroup_name=%2Aandromeda%2A`

To create queries with complex filter patterns is reserved for R&S CHM experts. For suitable filter patterns, ask your Rohde & Schwarz support engineer.

For an example in the `dashboards` context, see `dashboards` on page 48.

---

exports (Export of status information)

If two R&S CHM systems are separated by a security gateway, you can configure this key to send status information from one R&S CHM host to the other R&S CHM host.

To do so, you configure the target R&S CHM host and the data format that is used by R&S CHM for sending status monitoring information.

The following figure explains the basic principles. R&S CHM sends status information from a **Domain B** to a separated **Domain A**. On its way, the status information is filtered by a security gateway. R&S CHM host (A) can monitor its own items and display the monitored items from R&S CHM host (B).

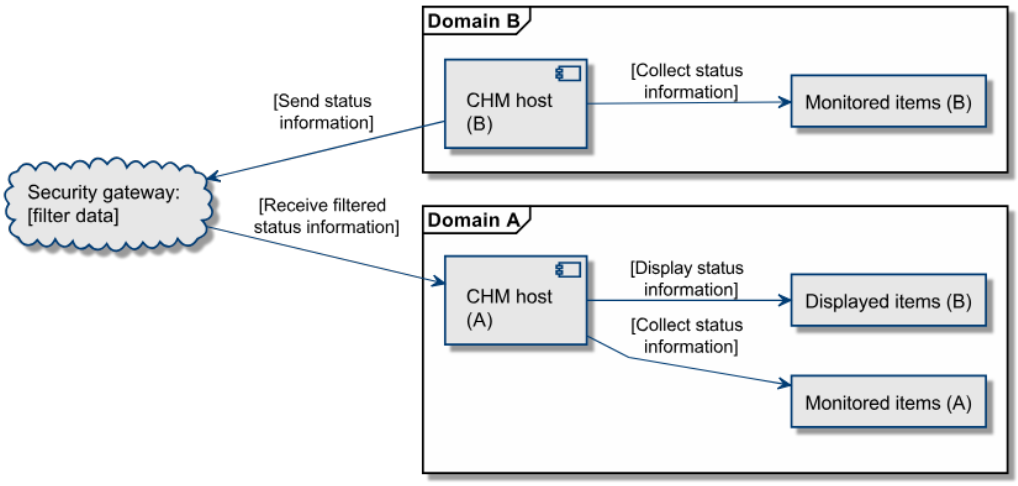


Figure 6-4: Exporting status information form domain B to domain A

Prerequisite

Both R&S CHM hosts need identical `chm.yaml` files. So, first change the file on one host. Then, transfer the file to the other host, e.g. using SSH. Example **2** at the end of this description shows the high-level structure of the `chm.yaml` file.

Parameters:

xmlhttp	Interface used for sending status information. This interface uses HTTP with content type application/xml on TCP port 5669.
target	Name of the R&S CHM host that receives the status information.
http_proxy	URL The gateway acts as an HTTP proxy (optional). Start the URL with `http://` because other formats are not supported.

Example: Configuration with two R&S CHM hosts and some high-level keys, including `exports` configuration.

```
hosts:
# First R&S CHM host
  - name: chm-k130-domain-A
    tags: [chm]
    connections: [local]
    logging:
      severity: debug
      facility: local0
    checks:
      - load:
      - os_process:
          name: icinga2
# Second R&S CHM host
  - name: chm-k130-domain-B
    tags: [chm]
    connections: [local]
    logging:
      severity: debug
      facility: local0
    exports:
      - xmlhttp:
          target: chm-k130-domain-A
          http_proxy: http://theproxy.myorg.net:5669
    checks:
      - load:
      - os_process:
          name: icinga2
```

logic (Combine logic status values)

R&S CHM system status monitoring lets you combine status values from multiple checks to a single, aggregated status value.

The following figure visualizes an example using the `worst` function. We assume that you monitor three components of a device, and DKN device 2 is defective. The `worst` function now takes the most critical value and hands it over to, e.g. the web GUI. The overall status indication that you configure using the `logic` key is ■ "WARNING".

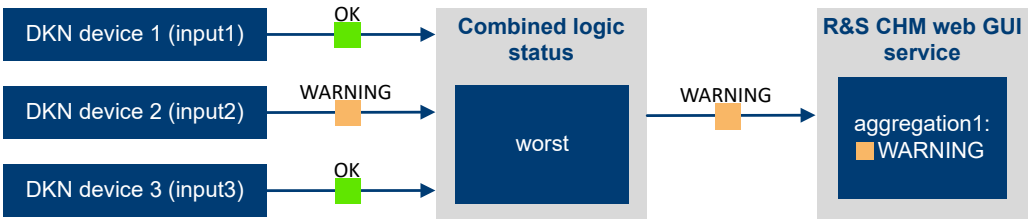


Figure 6-5: Combined logic status - "worst" example

The following figure adopts the previous example and explains how the specified keys are used to configure the logic function and to promote the aggregated status to the web GUI.

```
hosts:
  - name: chm2-staging-disa.rsint.net
    connections: [local]
    tags: [chm]
    checks:
      - passive:
          src_logic_id: aggregation1
      - dkn:
          logic_id: input1
          snmp_version: 2
          snmp_community: dkn
          port: 1234
          type: device_ready
          id: 7
      - dkn:
          logic_id: input2
          snmp_version: 2
          snmp_community: dkn
          port: 1234
          type: device_status
          id: 7
      - dkn:
          logic_id: input3
    logic:
      aggregation1:
        function: worst
        ins: [input1, input2, input 3]
```

1 2

Figure 6-6: Usage of involved keys

1 = Check with configured logic function instance
2 = Logic function instance used to promote the aggregated state to the web GUI

In detail, the previous configuration reads as follows: A logic identifier is assigned to each of the devices (`input1`, `input2`, `input3`). For logic function instance `aggregation1`, the logic function `worst` combines all input monitoring states and determines the most severe status as the *check result*. The `passive` key adopts the *check result* and shows it on the web GUI.

The next figure visualizes an example using the `best` function. We assume that you monitor two hosts and **host 2** (demodevice2.example.net) is defective. The `best` function now takes the best value and hands it over to the web GUI. The overall status indication that you configure using the `logic` key is ■ "UP".

Figure 6-7: Combined logic status - "best" example

For a "best" coding example, see example "2a) Example (best function)" at the end of this section.

Parameters:

<log_funct_inst> string

Instance of a logic function. You can specify multiple instances that you configure using the following `function` key.
You can also specify such a logic function instance for a check using `logic_id` on page 126.

function worst | best

The logic that is used for aggregation of status values.

worst
From all checked status values, the most severe status value determines the aggregated status value.

best
From all checked status values, the best status value determines the aggregated status value.

ins [<logic_function_instance 1>, <logic_function_instance 2>, ..., <logic_function_instance n>]

List of input values that are evaluated by the logic function.
See also: `logic_id` on page 126.
You can list logic function instances that are specified here, under `logic`. Also, you can list logic function instances that you have specified for a dedicated check using the `logic_id` on page 126 key.

Example:

1) DKN example (worst function)

This example is copy ready. It is identical to the example in Figure 6-6.

```
hosts:
  - name: chm2-staging-disa.rsint.net
    connections: [local]
    tags: [chm]
    checks:
      - passive:
          src_logic_id: aggregation1
      - dkn:
          logic_id: input1
          snmp_connection:
            version: 2
            community: dkn
            port: 1234
          type: device_ready
          id: 7
      - dkn:
          logic_id: input2
          snmp_connection:
            version: 2
            community: dkn
            port: 1234
          type: device_status
          id: 7
      - dkn:
          logic_id: input3
    logic:
      aggregation1:
        function: worst
        ins: [input1, input2, input 3]
```

Example:

2a) Example (best function)

```
- name: chm-server.example.net
  displayname: "CHM Server"
  tags: [chm]
  ...
  logic:
    logic_path_available:
      function: best
      ins: [device1, device2]

...

- name: Overall network path
  checks:
    - passive:
        src_logic_id: logic_path_available

- name: demodevice1.example.net
  checks:
    - ping:
        logic_id: device1

- name: demodevice2.example.net
  checks:
    - ping:
        logic_id: device2
```

Example: **3) NAVICS example (worst function)**
This example uses the result of the `navics` status check also as host result. The result is redirected using a logic function.

```
hosts:
  - name: chm-demo.rsint.net
    displayname: "Central CHM"
    connections: [icinga2_api, local]
    tags: [chm]
    checks:
      - ping:
    logic:
      copy_of_navics_VT1:
        function: worst
        ins: [navics_VT1]                          # (1)

    - name: VT1.navics
        connections: [snmp]
        checks:
          - passive:
              src_logic_id: copy_of_navics_VT1     # (2)
          - navics:
              logic_id: navics_VT1                 # (3)
              health_host: navics_server.local
              type: cwp
              eqid: VT1


    - name: navics_server.local
        connections: [snmp]
        snmp_connection:
          community: public
        checks:
          - ping:
```

(1) defines the logic function.
(2) receives the result from the logic function as host check result.
(3) sends the result to the logic function.

See also:

`passive` on page 172

`logic_id` on page 126

---

logging (System logging)

R&S CHM components send their log events into the Linux journal of the R&S CHM host. The journal is a binary, ring-buffer like database.

By default, Linux keeps the journal in volatile memory. You can persist messages to text files by using the `syslog` service. It reads the journal and exports to text files by some filter rules.

Note: Currently, R&S CHM does not provide means to change these export settings. If you use the `syslog` service, R&S CHM logging can cause high IO and CPU load and can degrade flash memory (SSDs). Ensure that only a subset of messages is exported, e.g. warning and higher.

For possible Linux logging options, see the related man pages.

Logging configuration

You configure the logging level in the `chm.yaml` file under the hosts key, see `hosts` on page 45.

Viewing logs

You can view the logs using the `journalctl` command.

Log events can originate at different components. For identification of the component, see Table 6-2.

Parameters:

severity — emerg | alert | crit | err | warning | notice | info | debug

Specifies the severity level, i.e. the importance of the message. For severity details, see Table 6-3.
If you change the severity, e.g. to `err`, only messages with severity `err` or higher are logged (`crit`, `alert`, `emerg`). For normal operation, we recommend severity `info`.

*RST: info

facility — local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7

Specifies the type of system that is logging the message according to RFC 5424. Messages with different facilities can be handled differently.

local0
Locally used facility code. All R&S CHM components send their logs as facility `local0`.

*RST: local0

icinga2_log_duration — time

Defines how long the replay log is stored. If a satellite cannot connect to the master, usually all collected data is stored for one day, i.e. 84600 s. After a reconnection, data is transmitted to the master.
For systems with small bandwidth or limited disk capacity, we recommend turning off the replay log. This measure avoids that all data is transmitted after a reconnection and overloads the WAN. To turn off the replay log, set the value to `0`.

*RST: 84600
Default unit: s

Example: **Logging configuration** under the `hosts` key. The severities with numerical code "0" to "5" are logged:

```
logging:
  severity: notice
  facility: local0
```

Example: Parameter **icinga2_log_duration** – replay log turned off:

```
logging:
  icinga2_log_duration: 0
```

Example: **Query of a specific component** with `journalctl -t <Identity>` or `journalctl SYSLOG_IDENTITY=<Identity>`:
For example, query the monitoring web UI and the web server status.

```
journalctl -t chm-monitoring-webui -t chm-httpd
```

Example: **Query of successful login, logout and failed login at the web interface**:

```
journalctl | grep "User logged in"

journalctl | grep "User logged out"

journalctl | grep "User failed to authenticate"
```

Example: **Filter for certain facilities** using `journalctl SYSLOG_FACILITY=<facility_code>`:

```
journalctl SYSLOG_FACILITY=16
```

For a list of facility codes and their meaning, see RFC 5424.

Example: **Filter messages by severity** with `journalctl -p <severity_or_severity_rage>` or `journalctl PRIORITY=<numerical code>`:

```
journalctl PRIORITY=6
```

Example: **Filter for automatic system control** with `journalctl -t chm LOGGER=chm.asc`.

Example: **Message output**:

```
Oct 07 11:25:48 test.local chm-httpd[10476]:
Thu Oct 07 11:25:48.797427 2021] [ssl:info]
pid 121267] [client 172.27.18.70:56854]
AH01964: Connection to child 2 established
(server test.local.net:443)
```

Table 6-2: Functional components

Component identity	Description
icinga2	Monitoring core
chm-monitoring-webui	Monitoring web UI
chm-monitoring-webui-audit	User login events at the monitoring web UI

Component identity	Description
chm-httpd	Web server status
chm-httpd-req	Web server request and responses

Table 6-3: Logging levels (severities) in order of decreasing importance

Parameter value	Numerical code	Description
emerg	0	Emergency - the system is unusable
alert	1	Alert - immediately act
crit	2	Critical conditions
err	3	Error conditions
warning	4	Warning conditions
notice	5	Normal, but significant, condition
info	6	Informational message
debug	7	Debug-level message

webinterface_url (Hyperlink to management web interface)

Configure a hyperlink to the management web interface of the monitored host. R&S CHM shows the hyperlink on the web GUI.

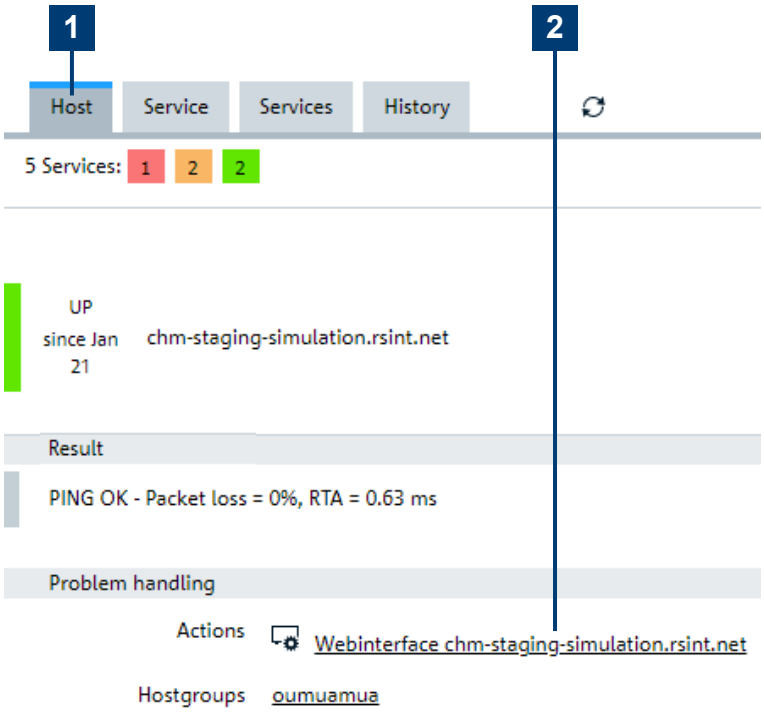


Figure 6-8: Hyperlink to a web interface

1 = "Hosts" tab
2 = Hyperlink to the web interface of the host

Configuration details

Select from the following options:

- Compose the link automatically from the host name. This mechanism requires that the host name is specified as a fully qualified domain name. R&S CHM system status monitoring automatically adds `https://` in front of the host name to compose the hyperlink, e.g. `https://chm-staging-simulation.rsint.net`.
- Specify a dedicated URL, e.g. `https://rohde-schwarz.com`. The web GUI shows this hyperlink.
- Omit the parameter from the configuration to omit the entry on the web GUI.

HTTP or HTTPS web address of the web interface of the host. If the name of the host is configured as a URI, CHM automatically composes the hyperlink, e.g. `https://chm-staging-simulation.rsint.net`.

Example: Automatically compose hyperlink:

```
- name: chm-staging-simulation.rsint.net
  webinterface:
```

Resulting hyperlink:
`https://chm-staging-simulation.rsint.net`

Example: Specific hyperlink:

```
- name: chm-staging-simulation.rsint.net
  webinterface: https://rohde-schwarz.com
```

---

webserver (Settings for usage by the R&S CHM webserver)

Specifies the path to a TLS certificate and corresponding private key on R&S CHM hosts and satellites (optional). The certificate is then used exclusively by the web server in TLS handshakes with web clients on port 443.

Parameters:

cert_file — The path to the TLS certificate.
Default unit: string

key_file — The path to the private key.
Default unit: string

Example:

```
webserver:
  cert_file: "/etc/pki/tls/certs/my_cert_file.crt"
  key_file: "/etc/pki/tls/certs/my_key_file.key"
```

6.5 Configuring web GUI users

You can select from the following configuration methods to access the web GUI:

- Default local user database, see "Local user database (method 0)" on page 62.

- LDAP-based or Kerberos-based authentication method, without or with fallback to the local user database, see "LDAP and single sign-on authentication methods" on page 63.

Local user database (method 0)

This method is the default log in method. It works without external dependencies to an authentication server like Active Directory.

You can use the local web GUI users "admin" and "operator" if the following applies:

- Authentication is not configured in the `chm.yaml` file.
- The `builtin` key is configured as an authentication method.

Both local users and their permissions are predefined. The "admin" user gets all permissions (acknowledge, check, comment, downtime, monitoring, maps, manual). The "operator" user only gets the monitoring and the maps permissions.

For information on permissions details, see `authorization` on page 68 > `permissions`.

To manage users in the local user database

You can list, add and delete users from the local user database.

► Use the following commands:

- `# chm_userlist`: Lists all currently existing users.
- `# chm_useradd`: Adds a new user with password. The password is hidden on the command line while typing.
- `# chm_userdel`: Deletes a user.

To change the passwords of existing users

1. Delete the user.
   `# chm_userdel`
2. Add the user again with the new password. Use a unique and strong password that complies with the security policies in your company.
   `# chm_useradd`
3. Configure the user permissions for the newly added user in the `chm.yaml` file. See the following procedure ("To control the permissions of local web GUI users" on page 63.

Example:

The following examples show how to use the commands for user management.

List all users	Add a new user with password	To delete a user

`# chm_userlist`
` name`
`----------`
` admin`
` operator`
` test1`
` test2`
` ee`
` uh`
`(6 rows)`
	`# chm_useradd`
	`new user: testuser`
	`new password:`
	`INSERT 0 1`
	`completed`
		`# chm_userdel`
		`user to remove: testuser`
		
		`DELETE 1`
		`completed`

To control the permissions of local web GUI users

You can use the local users "admin" and "operator" without further configuration. However, you can assign specific permissions to them, e.g. to the "operator".

1. Under the first `hosts` list entry, add the `authorization` key.
2. Configure the `permissions` for specific `roles`. For example, add `check` and `acknowledge` for `operators`.
   For all permissions and configuration details, see `authorization` on page 68 > `permissions`.

   You have configured specific permissions of the local web GUI user.

In a distributed system with several R&S CHM hosts, the local user database is not synchronized between the instances.

LDAP and single sign-on authentication methods

You can select between LDAP-based user authentication or Kerberos-based single sign-on (SSO) user authentication methods. R&S CHM then uses the configured method to restrict permissions and users that can access the web GUI. By using one of these methods, you can manage users or user groups centrally and enhance security.

Usage of all SSO methods requires several 3rd-party services, e.g. LDAP and Kerberos. Implementation and configuration of these services are not covered by this user guide. The services also require configuration of the web GUI users in the central user management of your organization. Ask the local system administrator for support.

Here, we describe the configuration of the `chm.yaml` and the requirements for using 3rd-party services.

R&S CHM supports user and group management with these directory services to log in to the web GUI:

- **LDAP with dedicated bind user (method 1)**
  LDAP bind uses the credentials of dedicated bind-users defined in the password store. Users log in with a password as stored in a central LDAP-service.
- **LDAP anonymous bind (method 2)**

LDAP bind does not require credentials. Users log in with a password as stored in a central LDAP-service.

- **Kerberos SSO with SSSD for group information (method 3)**
  Users are logged in automatically with a ticket that they receive and cache during Windows- or Linux desktop login. Also, "role to group mapping" supports groups from a central LDAP-service.
- **Kerberos SSO with users only (method 4)**
  Users are logged in automatically with a ticket that they receive and cache during Windows- or Linux desktop login.
- **LDAP-based or Kerberos-based authentication (methods 1 to 4), with fallback to the local user database**
  If you configure R&S CHM for exclusively using LDAP and single sign-on authentication methods, the local users are no longer available on the web GUI. Only LDAP users or KDC users can access the web GUI for system status monitoring. However, you can configure a fallback method to the local user database if you specify the `builtin` key as an additional authentication method.

For detailed configuration examples, see the following `authentication` and `authorization` syntax descriptions.

To configure LDAP and single sign-on authentication

1. Under the host with the `[chm]` tag, configure the `authentication` key.
2. Configure user **authentication** as described under `authentication` on page 64.
3. Configure user **authorization** as described under `authorization` on page 68.

   You have configured R&S CHM for LDAP or SSO support. Web GUI users can log in to the web GUI with the users or user groups that are already configured on your network.

authentication....64
monitoring....64
builtin....65
gssapi....65
ldap....66
authorization....68

---

authentication (Authentication)

Configures the authentication method for all web GUI users.

Parameters:

monitoring	All authentication keys are specified under this key. See `monitoring` on page 64.

---

monitoring (Authentication methods)

All authentication keys are specified under this key.

Parameters:

builtin	Built-in authentication method (fallback), see `builtin` on page 65.
gssapi	GSSAPI-based authentication methods, see `gssapi` on page 65.
ldap	LDAP-based authentication methods, see `ldap` on page 66.

---

builtin (Builtin authentication method)

Enables the local, built-in user database. If you specify this key, you can log in with the users "admin" and "operator" from the local user database when authentication using LDAP or SSO is not possible.

Default credentials

- **Operator**: *operator*, password *chmoperator*
- **Administrator**: *admin*, password *chmadmin*

If you only specify `builtin` without other authentication details, the configuration is equivalent to leaving out the `authentication` configuration at all, i.e. only the local users are available.

Example: **Local user database (method 0)**

Usage of the local user database (`builtin` specified):

```
authentication:
  monitoring:
    - builtin:
```

Usage of the local user database (`builtin` not specified):

```
authentication:
  monitoring:
```

Both authentication methods are equivalent.

---

gssapi (GSSAPI-based authentication method)

Configures the exchange of tokens as used for authentication methods "Kerberos SSO with SSSD for group information (method 3)" and "Kerberos SSO with users only (method 4)".

Parameters:

keytab <file_path>

Specifies the path to the key table (keytab) file.

Example: **SSO authentication variants**

SSO authentication only:

```
authentication:
  monitoring:
    - gssapi:
        keytab: /etc/opt/rohde-schwarz/chm/HTTP.keytab
```

SSO authentication with fallback to the local user database:

```
authentication:
  monitoring:
    - builtin:
    - gssapi:
        keytab: /etc/opt/rohde-schwarz/chm/HTTP.keytab
```

Example: **Kerberos SSO with SSSD for group information (method 3)**

This method retrieves user information from Kerberos tickets. The LDAP group information is requested using the POSIX command `id <user>`. This command version uses the name service switch (NSS) to query group information through the privileged system security services daemon (SSSD) from the LDAP.

Only the SSSD reads the secret key table (keytab file) that contains a key for service principal. Only the SSSD connects to LDAP.

Specify the path to a valid `*.keytab` file. In this example, also the fallback login method `builtin` is configured:

```
authentication:
  monitoring:
    - builtin:
    - gssapi:
        keytab: /etc/opt/rohde-schwarz/chm/httpd.keytab
```

Example `keytab` file contents:

```
ktutil:  read_kt HTTP.chmserver.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------
   1    2 HTTP/chmserver.your.org@YOUR.ORG
```

---

ldap (LDAP-based authentication method)

Obtains the credentials from a centrally maintained LDAP server.

Parameters:

server — <FQDN> | <IP_address>

Specifies the address of the LDAP server, either its fully qualified domain name or its IP address. You can specify two redundant LDAP servers to enhance availability of this authentication method.

encryption — ldaps | starttls

Configures the encryption method that is used to secure the communication between the LDAP server and the R&S CHM host.
The LDAP server must support your choice.

ldaps
Configures the *LDAP over SSL* protocol.

starttls
Configures the *LDAP over TLS* protocol.

base_dn — string

Specifies the LDAP distinguished name (DN) of the branch of the directory where the searches for users start from. The DN uniquely identifies an object in the Active Directory.

user_class — string

Specifies the LDAP class of user objects.

user_name_attr — string

Specifies the LDAP attribute that holds the user's name that is used for the login.

bind_dn — string

Specifies the DN used to bind to the server when searching for users. Only necessary for authentication method "LDAP with dedicated bind user (method 1)", see following example.

bind_pwd_path — string

Path of the LDAP password within the R&S CHM password store. Only necessary for authentication method "LDAP with dedicated bind user (method 1)".
See also: Section 6.7, "Managing password identifiers", on page 76

Example:

LDAP with dedicated bind user (method 1)
This method requires a user name and the path of the authentication password for the bind operation. A dedicated bind user authenticates itself against LDAP.
Specify `bind_dn` and `bind_pwd_path`:

```
authentication:
  monitoring:
    - ldap:
        server: [ldapserv.ourlocal.net, ldapserv2.ourlocal.net]
        encryption: ldaps
        base_dn: ou=Foo_Users,dc=foo,dc=bar,dc=baz
        user_class: user
        user_name_attr: sAMAccountName
        bind_dn: user
        bind_pwd_path: ldap/icinga_ldap_user
```

Example:

LDAP anonymous bind (method 2)
This method does not require user credentials at all and accesses the LDAP as anonymous. To use this method, it is necessary that you explicitly allow this binding method on the LDAP server.
Omit both keys `bind_dn` and `bind_pwd_path` or leave them empty:

```
authentication:
  monitoring:
    - ldap:
        server: [ldapserv.example.net, ldapserv2.example.net]
        encryption: ldaps
        base_dn: ou=ldap_users,dc=ldapserv,dc=ourlocal,dc=net
        user_class: user
        user_name_attr: sAMAccountName
```

---

authorization (Authorization)

Configures the authorization method for all web GUI users.

Parameters:
monitoring

roles — string

Specifies and configure the user roles that are available.
You can choose the names freely, e.g. `administrators` and `operators`. The specified roles are generated on the R&S CHM host.

permissions — acknowledge | check | checkdetails | comment | downtime | graphs | manual | maps | reporting | statusview | systemcontrol

List of permissions that are assigned to the role (optional).

acknowledge
Acknowledge hosts or service problems by selecting "Acknowledge" on the web GUI.

check
Start a check immediately by selecting "Check now" on the web GUI.

checkdetails
Shows more detailed information on the web GUI about check execution, such as command, check source, last update, next update and check attempts. This information can be useful for administrators.

comment
Leave a comment for a host or service by selecting "Comment" on the web GUI.

downtime
Schedules a downtime by selecting "Downtime" ⏻ on the web GUI. Host or service problems do not show up for the dedicated host or service during the downtime.

graphs
Shows graphs with performance data for supported services on the web GUI. Web GUI users can select the period of time.
See also: `graphs` on page 72

manual
In combination with a passive check configuration, this permission creates a button for "passive" checks on the web GUI. See also: `manual` on page 162

maps
Shows maps on the web GUI. See Section 6.9, "Configuring graphical system views (maps)", on page 83.

reporting
Shows reports on the web GUI. See `reporting` on page 74.

statusview
Adds a "Status View" page to the web GUI > "Overview" menu. Thus, you can provide a page on the web GUI that sorts all the hosts via the host group. Configure the hosts groups for your hosts accordingly, see `hosts` on page 45 > `hostgroups`.

systemcontrol
Access the "System Control" view on the web GUI. To configure the management functions that are provided under "System Control", see `chm_remote_grpc` on page 137.
See also: Section 6.8.2, "Configuring the System Control view", on page 82

users — list of users

List of users to which R&S CHM applies the role, e.g. `admin`, `operator`, `john` (optional).

groups — list of groups

List of user groups to which R&S CHM applies the role, e.g. `company_chm_admins` (optional).

Example: **LDAP with dedicated bind user (method 1)**

Example with LDAP `users`:

```
authorization:
  monitoring:
    roles:
      operators:
         permissions:
           - acknowledge
           - comment
           - check
         users:
           - chm_operator
           - chm_monitor
```

Example with LDAP `groups`:

```
authorization:
  monitoring:
    roles:
      administrators:
         permissions:
           - acknowledge
           - comment
           - downtime
         groups:
           - company_chm_admins
```

Example: **Kerberos SSO with users only (method 4)**

This method retrieves user information from Kerberos tickets. You require a configured key distribution center (KDC) in your system.

Group information is not included in Kerberos tickets. As a consequence, you cannot use groups for authorization if only Kerberos without LDAP is available due to security restrictions.

Specify `permissions` for `users`:

```
authorization:
  monitoring:
    roles:
      operators:
        permissions:
          - acknowledge
          - comment
          - check
        users:
          - chm_operator@domain.org
          - chm_monitor@domain.org
```

Example: **Combined authentication method: Kerberos SSO with SSSD for group information (method 3) with fallback to local user database (method 0)**

This example also configures the `builtin` fallback authentication method. The user `admin` is the fallback user.

```
authentication:
   monitoring:
     - builtin:
     - gssapi:
         keytab: /etc/opt/rohde-schwarz/chm/HTTP.keytab
authorization:
   monitoring:
     roles:
       commenter:
         permissions:
           - comment
         users:
           - johndoe@RSINT.NET
           - admin
       downtimer:
         permissions:
           - downtime
         groups:
           - domainoperators
```

Example: **Local user database (method 0)**

`builtin` authentication method in combination with authorization:

```
authentication:
  monitoring:
    - builtin:
authorization:
  monitoring:
    roles:
      commenter:
        permissions:
          - comment
        users:
          - operator
```

6.6 Configuring R&S CHM features

Configures additional R&S CHM features. You can enable or disable features and adapt the feature-specific configuration.

Supported features:

- `graphs`

Configures the performance data processing that is received by the checks and prepares this data for visualization in graphs on the web GUI.

- `forget_states_on_restart`
  Resets the timing information for the "UP" value after a restart of the R&S CHM service or restart of the R&S CHM master.
- `reporting`
  Configures the amount of information that is provided in reports on the web GUI.

---

graphs (Performance data processing, received by checks)

Configures the performance data processing that is received by checks and needed for display on the web GUI. Configuration has an impact on the disk usage and on the data shown in the graphs on the web GUI. To make graphs visible on the web GUI, also specify the `graphs` permission under `authorization` on page 68.

Parameters:

retentions — string

Configures retention times. You can specify multiple retentions. Separate retention value-pairs in the format *<frequency>:<history>* by commas. Specify multiple retentions from *most-precise:least-history* to *least-precise:most-history* and consistent in time *frequency(n) <= history(n-1)*.
Frequencies and histories are specified using the suffixes specified in Table 6-4.

*RST: 1m:30d
The default values (1m:30d) have the following meaning: Data is stored in 60 s accuracy for 30 days, which results in a file size of 518.428 kbyte.
Calculation of the disk space requirements
Storage file size = storage-meta + per-retention-meta + data
In more detail:
*Storage file size = 16 byte + <number of retentions> * 12 bytes + <number of data points> * 12 byte*

enabled — true | false

Enables or disables the graphs feature.

Example: Host configuration for graphs with multiple retentions.

```
- name: MyHost
  ...
  features:
    graphs:
      retentions: "60s:1d,5m:30d,1h:3y"
      enabled: true
  ...
```

The data is stored in 60 s accuracy for 1 day, 5-minute accuracy for 30 days and 1 h accuracy for 3 years.
A single graph with this setting uses up to 436.372 kbyte of storage. An R&S CHM check can have more or less than a single graph. E.g., a system with 100 hosts and 10 checks per host and 2 graphs per check needs around 873 Mbyte of disk space to store all graphs with this setting. The disk space requirements are different on your system depending on the configuration and size.

Table 6-4: Suffixes for frequencies and histories

Suffix	Meaning
s	Second
m	Minute
h	Hour
d	Day
w	Week
y	Year

Data file handling

The data files do not automatically shrink if you reduce the retentions. If you rename hosts and services, the data files are not deleted automatically. For a clean start, remove existing storage files and restart the carbon-cache service:

```
rm -rf /var/lib/carbon/whisper/icinga2
```

```
systemctl restart carbon-cache.service
```

---

forget_states_on_restart (Reset uptime data after restart or reboot)

Resets the timing information for the "UP" value by setting all R&S CHM host states to "DOWN" and the service states to "UNKNOWN" after a restart of the "R&S CHM" service or restart of the R&S CHM master.

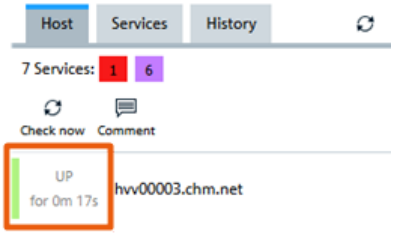


Figure 6-9: UP timing information

If the key is not configured, the timing information not reset.

Parameters:

enabled — true | false

Resets the uptime if set to `true`.

---

reporting (Configuring reports)

Defines the top level key to configure reporting in R&S CHM. Users with `permission` "`reporting`" can access the "Reporting" view on the web GUI.

See also: `authorization` on page 68

Parameters:

reports — string

Defines the reports on the web GUI > "Reporting" (optional).
See `reports` on page 75.

retention_period — string

Retention period of all reports (optional).
For the meaning of the time suffixes, see the following table.

*RST: 30d

Example:

```
features:
  reporting:
    retention_period: 60d
```

Table 6-5: Time suffix overview

Suffix	Meaning
no suffix | `s`	Second
`m`, `min`	Minute
`h`	Hour
`d`	Day
`w`	Week

reports (Defining reports templates)

Defines available report templates and reports that are available on the web GUI > "Reporting" > "Reports" for download. The time formats used in the keys are based on ISO 8601.

Parameters:

name — string

The name that is displayed in the report template and the report on the web GUI.

type — string

The type defines the template that is used. It serves for internal identification purposes and is not shown on the web GUI. Currently, only `type: ArgusReport` is supported.

parameters — The parameters that make the report: `timeframe`, `host_scope`, `frequency` and `start`.

timeframe — string

Time interval that is used for creating reports. For supported values, see Table 6-6. The timeframe is also used in report names besides date and time.

current -month

host_scope — string

The hosts that are included in the report (optional).

*****

Includes all hosts.

<host name>*
Includes all hosts whose host names start with <host_name>.

<host name A> | <host name B>
Includes host A and host B.

frequency — string

Defines how often R&S CHM generates a report (optional). Use `frequency` together with `start`.

<time value><time unit>
For supported values and units, see Table 6-6.

start — string

Start time of report generation (optional).

<YYYY-MM-DD>T<hh>:<mm><UTC time difference>

Example:

```
reporting:
  reports:
    - name: Argus Report
      type: ArgusReport
      parameters:
        timeframe: current-month
        host_scope:*        all hosts where selected
        frequency: 1h
        start: "2026-07-27T13:16"
```

Table 6-6: Timeframe values

1h	1 hour
1d | last-day	Last day
current-day	Current day
1w	One week
current-week	Current week
last-week	Last week
1m	One month
current-month	Current month
last-month	Last month
1y	One year
current-year	Current year
last-year	Last year

6.7 Managing password identifiers

All passwords for communication between R&S CHM and an LDAP server or R&S CHM and the monitored services are encrypted using GPG. To ease password handling, R&S CHM provides a **password manager**.

The password manager lets you safely specify necessary password identifiers for communication of R&S CHM via the following interfaces:

- LDAP simple authentication password
- SNMP
- SSH
- Proprietary interfaces, e.g. VMware

Change all passwords in the password store before your system goes live.

To list all password identifiers

1. Log in to the R&S CHM host.
2. Enter the following command:
   `chmpass ls`

   The currently defined password identifiers are listed. For an example output, see the following example.

Example: List configured password identifiers

```
$ chmpass ls
Password Store
├── tiger
├── bumblebee
└── ant
```

To add a password identifier

1. Log in to the R&S CHM host.
2. Type the following command:
   `chmpass insert <password_identifier>`.
   Example: `chmpass insert tiger`
3. Enter the password identifier.
4. Repeat the password identifier.

   You successfully added the password identifier.

To remove a password identifier

1. Log in to the R&S CHM host.
2. Enter the following command:
   `chmpass rm <password_identifier>`
3. Confirm deletion.

   You successfully removed the specified password identifier.

Example: Delete a password identifier

The name of the identifier is "tiger".

```
$ chmpass rm tiger
Are you sure you would like to delete tiger? [y/N] y
removed '/var/opt/chm/password-store//tiger.gpg'
```

To set a password identifier in the configuration file

1. Access the `chm.yaml` file.
   See also: "To access the configuration file" on page 43

2. Under the `check:` key for the resource, add the key-value pair:
   `<identifier>: <password_identifier>`
   Examples
   - For `snmpv3`: `snmp_connection` > `secname: tiger`
   - For `vmware`: `user: lion`
   - For `xcp_ng`: `ssh_connection` > `username: peter`

   R&S CHM can access the checked resources via the set password identifier.

6.8 Configuring R&S RAMON for monitoring

This monitoring method uses a gRPC-based R&S CHM service called `chmrd`. It replaces the deprecated Windows SNMP service.

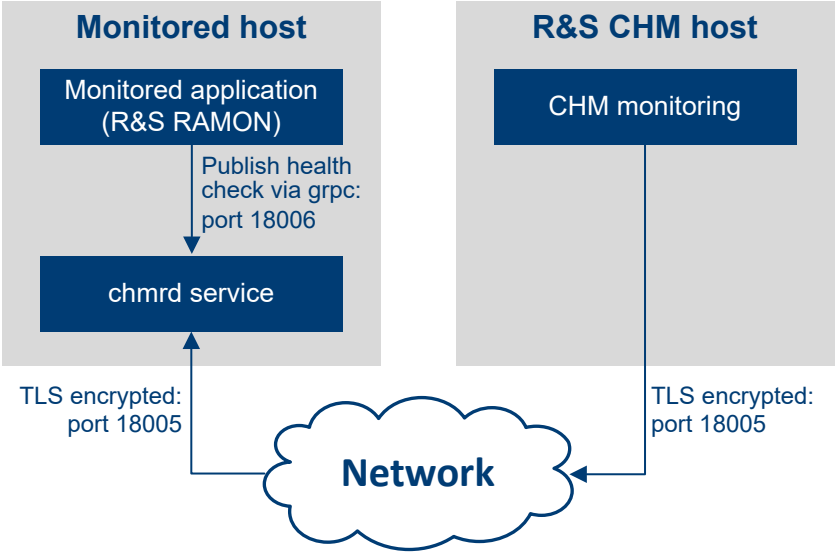


Figure 6-10: Monitoring of applications, e.g. R&S RAMON

The following description explains the monitoring steps visualized in the previous figure.

Monitored application and R&S CHM

Applications "publish" monitoring data to `chmrd`. R&S CHM fetches the monitoring data from `chmrd`.

The chmrd service

The service `chmrd` gathers monitoring data sent by applications and makes it available to R&S CHM instances. Currently, it has to be installed on the same Windows host that also runs the monitored application. It is necessary that you install `chmrd_<version>.msi` on the agent that runs R&S RAMON, see Section 4.2.1, "Installing Windows agents", on page 21.

Interface definition

The service provides a gRPC interface that can be used to both send and query monitoring data. The interface is defined in a protobuf file. This file describes the services provided by `chmrd` and the data model that is used for communication and even how this data is serialized on the wire. The file thus takes the role of a serialization document.

Security aspects

The `chmrd` service uses two separate TCP ports:

- For communication with clients on the same host: local port, default port number 18006
  On the local port, the service only listens for connections from localhost. There is no encryption or authentication or authorization when using the local port. Its main use case is for communication between `chmrd` and the monitored application.
- For clients on remote hosts: remote port, default port number 18005.
  When communicating over the remote port, `chmrd` enforces TLS encryption and client authentication using X.509 certificates to secure network communication. There is no authorization mechanism in place yet which means an authenticated client is allowed to both send and query monitoring data without any restrictions.

6.8.1 Configuring the chmrd service

The only officially supported way of configuring `chmrd` is to pass command-line arguments to the service.

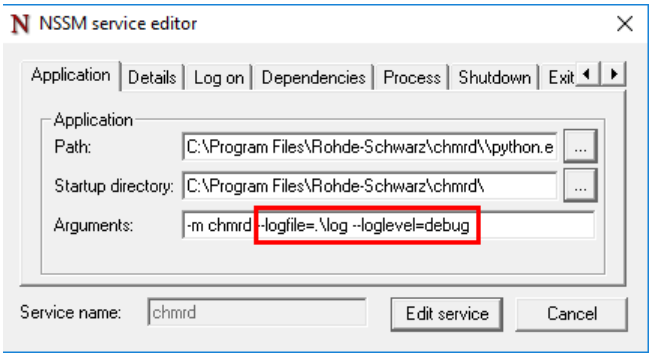
Typically, you can use the default `chmrd` configuration. However, if you need to change the configuration, continue as described in the following procedure.

To configure the chmrd service

This procedure assumes that the `chmrd` software is already installed.

1. Open the installation directory:
   `C:\Program Files\Rohde-Schwarz\chmrd\`
2. Open a command prompt window in the installation directory.
3. Run the following command:
   `.\nssm.exe edit chmrd`

   The "NSSM service" editor opens.



4. Configure the desired arguments as listed in Table 6-7.

   Note: Always keep the "-m chmrd" argument. This information tells the python interpreter which module to use to start the service.

Table 6-7: Command-line arguments for configuring the chmrd service

Argument (short)	Argument (long)	Default value	Description
"-a"	"--address"	"0.0.0.0"	IP address the server runs on
"-p"	"--port"	"18005"	Port for connections from remote hosts
"-P"	"--local-port"	"18006"	Port that clients on localhost can use without needing to authenticate themselves
"-d"	"--cert-dir"		Directory with certificates and keys
"-C"	"--server-cert"		Server certificate path
"-R"	"--server-root-cert"		Server root certificate path
"-K"	"--server-priv-key"		Server-private key path
"-c"	"--client-root-cert"		Client root certificate path
	"--insecure"		If set, disable encrypted message transport and server/client authentication (without a value)
	"--loglevel"	"info"	One of "debug", "info", "warning", "error", "critical"
	"--logfile"		Logfile path
	" -- logfilemode"	"w"	"a" or "w" "a" for appending to log file. "w" for truncating log file and starting a new one when the service is restarted

Example:

The following arguments set specific ports and how the log file is treated.

"-m chmrd -p=18007 -P=18008 --logfilemode=a"

About certificates and keys

All certificates and keys used for `chmrd` have to be PEM encoded.

To achieve encrypted and authenticated network communication, `chmrd` needs the following:

- **A server certificate chain**
  A certificate chain is a list of certificates where the issuer of each one of them matches the subject of the following. Also each certificate - except for the last - is signed with the secret key corresponding to the next certificate. The last certificate in the chain is self-signed, which makes it a root certificate.
  Usually, this chain consists of a certificate issued for the host on which the service is running. This certificate is followed by some root CA's certificate that was used to issue the certificate of the host. You can specify these two parts of the certificate chain by using the "--server-cert" and the "--server-root-cert" arguments, including the path to the corresponding files.
  For the uncommon use case that the chain consists of more than two certificates, you can split up the certificates to the two files specified by "--server-cert" and "--server-root-cert". Make sure that the resulting chain fulfills the criteria for a certificate chain described above.
- **A server-private key**
  This key is the private key corresponding to the certificate on the server, i.e. the monitored host used for encrypting network communication. The file containing the key can be specified by using the "--server-priv-key" argument.
- **A client root certificate**
  `chmrd` expects remote clients to provide a certificate chain to authenticate themselves. You can specify a file containing one or more root certificates for these chains by using the "--client-root-cert" argument.

Default paths for certificates and keys

There are different possible combinations of how to use command-line arguments in `chmrd`. The following tables list the defaults that are used in the different cases **A**, **B** and **C**.

A) If no command-line arguments are specified, the defaults use the fully qualified domain name (FQDN) of the host, see the following table.

Table 6-8: No command-line arguments are specified

Argument (long)	Default value
"--server-cert"	`C:\ProgramData\icinga2\var\lib\icinga2\certs\<FQDN>.crt`
"--server-priv-key"	`C:\ProgramData\icinga2\var\lib\icinga2\certs\<FQDN>.key`
"--server-root-cert"	`C:\ProgramData\icinga2\var\lib\icinga2\certs\ca.crt`
"--client-root-cert"	`C:\ProgramData\icinga2\var\lib\icinga2\certs\ca.crt`

The default file locations here correspond to the certificate settings you usually already made for the R&S CHM Windows agent, see Section 5, "Deploying certificates", on page 31. Thus, no extra configuration is necessary for the `chmrd` service. Also,

the `chmrd` service expects that both server and clients to use same root certificate by default.

B) If you specify "--cert-dir", you can set a custom location for all certificates and key, see the following table.

Table 6-9: Only --cert-dir is specified

Argument (long)	Default value
"--server-cert"	`<CERT_DIR>\<FQDN>.crt`
"--server-priv-key"	`<CERT_DIR>\<FQDN>.key`
"--server-root-cert"	`<CERT_DIR>\ca.crt`
"--client-root-cert"	`<CERT_DIR>\ca.crt`

C) If one or all "--server-cert", "--server-root-cert", "--server-priv-key", "--client-root-cert" are specified, you can always specify a customized, absolute path to certificates and key.

For information about configuration of the check in the `chm.yaml` file, see `chm_remote_grpc` on page 137.

6.8.2 Configuring the System Control view

You can configure a "System Control" view that is available on the web GUI for R&S RAMON components that are connected via gRPC. See `chm_remote_grpc` on page 137.

You can select from these management functions for web GUI users that have got the permission `systemcontrol`:

- "Self-Test"
- "Reboot"
- "Shutdown"

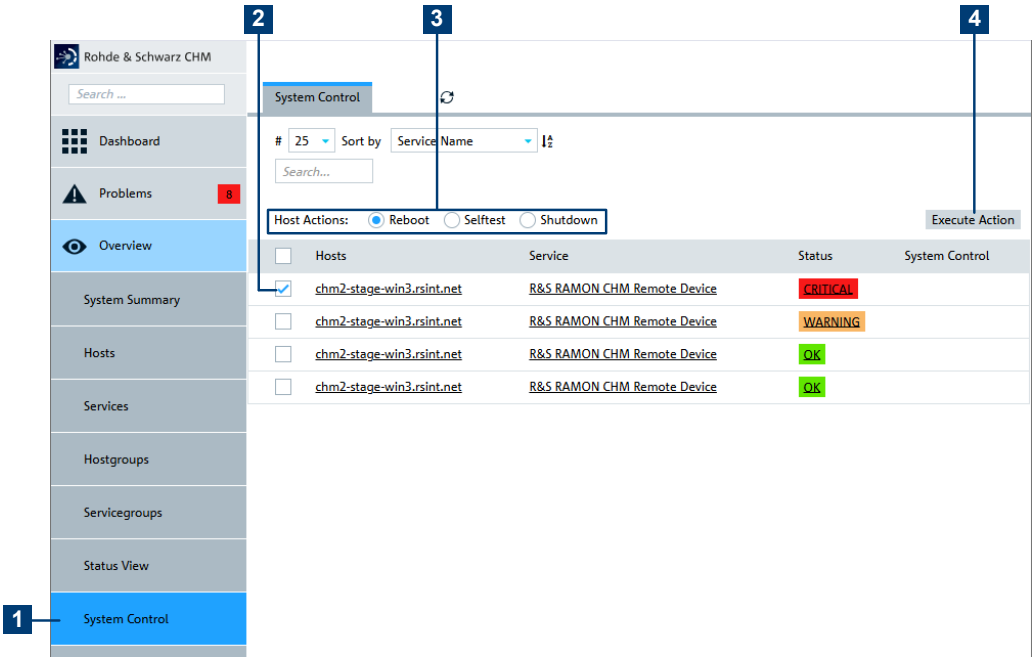


Figure 6-11: System Control on the R&S CHM client GUI

1 = Menu entry
2 = Selected device
3 = Configured functions
4 = Execute selected action

6.9 Configuring graphical system views (maps)

You can add and configure graphical elements to R&S CHM. These elements let you visually track the system's status on customizable maps, providing a more intuitive and comprehensive understanding of the system's operation. After the configuration in the `chm.yaml` file, you can find all configured graphical system views on the web GUI under "Maps" (**1**, **2**). R&S CHM lets you visualize the status of hosts, services, host groups or service groups.

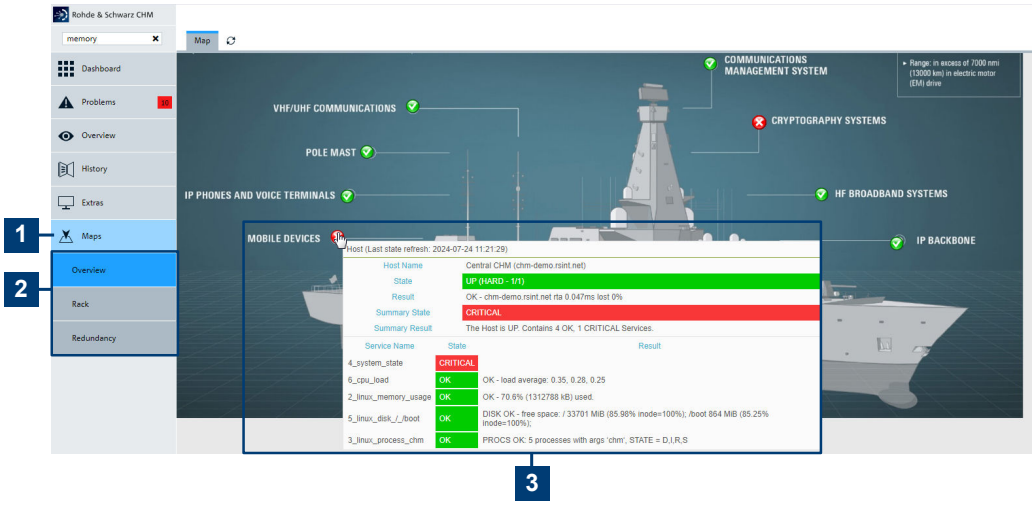


Figure 6-12: Graphic system views (maps)

1 = "Maps" main menu
2 = Individual "Maps" views
3 = Mouse over on the status icon provides details. Select the status icon to navigate to the configured host or service.

- You cannot configure maps if the web GUI users are using authentication method SSO in combination with `builtin`, i.e. the fallback option to the local user database.
- All example figures here explain the general behavior but do not reflect true systems or subsystems.

To prepare the background images

Each map is composed of a background image, a status icon and an optional label. To determine the coordinates for the status icons on an image, you can use almost any image editor. The labels are automatically filled with the `displayname` or `name` of that host or service. The label is shown to the right of a status icon.

1. Save the background images as pixel graphics of type PNG or JPEG in the correct final size and resolution. We recommend using images in 96x96 pixels resolution.

   Note: R&S CHM does not modify or adapt the image size.

2. Upload the images to the R&S CHM host. To do so, you can use, e.g. WinSCP or LCSM.
   `/etc/opt/rohde-schwarz/chm/maps/`

3. Determine the (x,y)-coordinates for the **status icons** that you want to show on the images. The following example shows how you use Microsoft Paint to determine the coordinates for the status icons on the graphic.

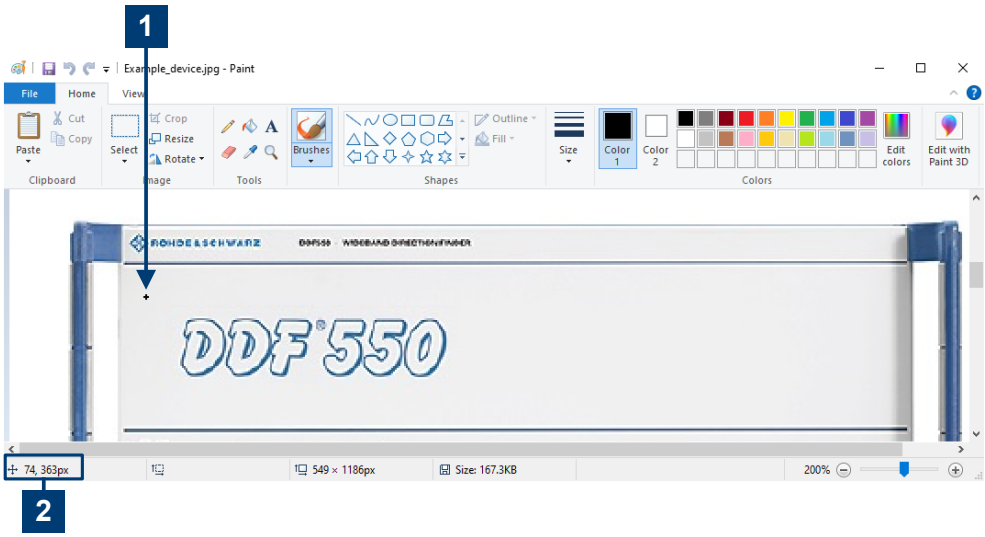


Figure 6-13: Coordinates of a cursor position

1 = Pointer location
2 = Status bar with the coordinates, e.g. "74, 363px"

a) Open the graphic in the editor.
b) If necessary, turn on the status bar.
c) Point to the location where you want to insert status icon (**1**). The location marks the top-left position of the status icon.
The first value is the x-value and the second is the y-value (**2**).

To configure graphical system views (maps)

1. Open the `chm.yaml` file. See Section 6.3, "Changing the configuration", on page 43.

2. Enter the coordinate value pairs (see step 3) in the `maps` > `x` and `y` keys of the corresponding hosts and services. Optionally, you can configure item-specific label formats.
For syntax details, see `maps` on page 126.

3. Configure the top-level `maps` key in the `chm.yaml` file. Here, you specify the background image and label layout.

   You can configure the label layout for each map separately:
   - The name that appears on the R&S CHM GUI
   - The image filename
   - The label format: background color, label border and label style. Also, you can hide all labels on a map.

   For syntax details, see `Graphical system view (maps)` on page 86.

   When finished, you can view the result on the web GUI.

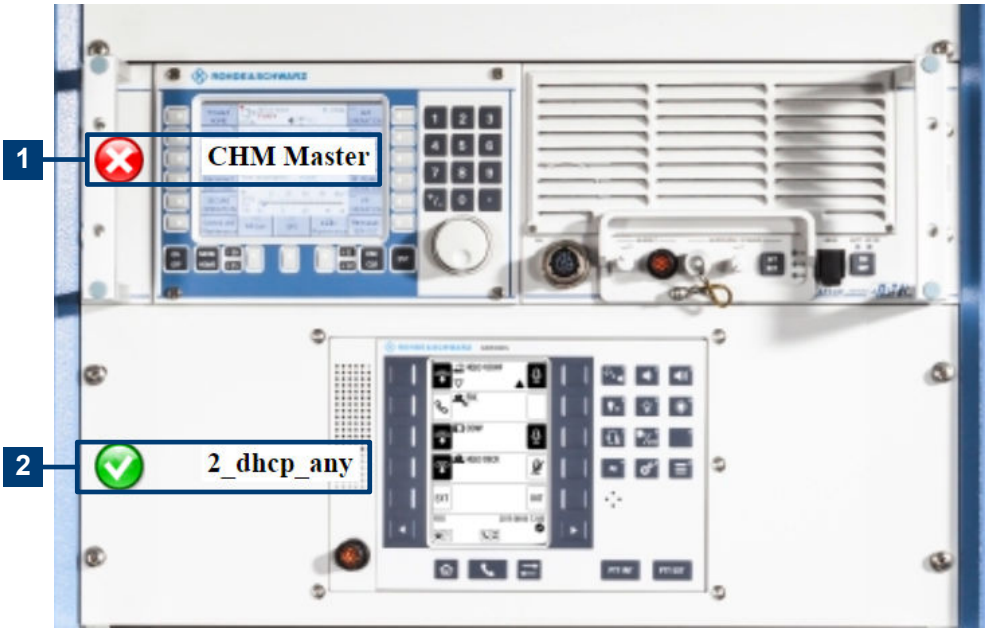


Figure 6-14: Map example with status icons and labels

1 = "CHM Master" host, status "CRITICAL"
2 = "2_dhcp_any" service, status "OK"

Graphical system view (maps) maps

Configures all graphical elements for visualization of the system status on the R&S CHM GUI. To view the maps on the web GUI, users need the `maps` permission.

Related parameters

- `maps` on page 126
- Section 6.5, "Configuring web GUI users", on page 61

Parameters:

name	string Name of the individual subsystem on the GUI. For an example, see Figure 6-12.
background_image	file name Filename of the background image, e.g. `my_system.jpg`. R&S CHM supports the file types PNG or JPEG.
label_show	True | False Visibility of labels on the GUI (optional). If no key is specified, the labels are shown (`True`). **False** Hides the labels on the GUI.

label_background — hex string

Map-specific background color of the label (optional).
Specifies the colors for graphical elements on the GUI in Hex code values. Table 6-10 lists basic colors that you can start with (from the West Library/Texas Weslyan University, page retrieved 2024-02-06). On the internet, you can find multiple pages that let you pick the color codes, e.g. HTML color codes.

label_border — hex string

Map-specific border color of the label (optional).

label_style — string

Map-specific font family, font weight and font size of the label text (optional).
Standard CSS font families: `serif`, `sans-serif`, `cursive`, `system-ui`.
Standard font weights: `normal`, `bold`, `lighter`, `bolder`, `<font-weight-absolute>` (numeric values between 1 and 1000)

Example: Three individual `maps` configurations. Specify the `maps` key on the same indention level as the `hosts` key.

```
maps:
  - name: Overview
    background_image: ship1.jpg
    label_show: False
  - name: Rack
    background_image: rack1.jpg
    label_background: "#FFFFFF"
    label_border: "#FFFFFF"
    label_style: "font-family:sans-serif;\
color:#000000;font-weight:bold;font-size:20px;"
  - name: Redundancy
    background_image: redundancy1.jpg
    label_background: "#FFFFFF"
    label_border: "#FFFFFF"
    label_style: "font-family:sans-serif;color:#000000;\
font-weight:bold;font-size:20px;"
```

Table 6-10: Basic color codes

Color	Hex code
Black	#000000
White	#FFFFFF
Red	#FF0000
Lime	#00FF00
Blue	#0000FF
Yellow	#FFFF00

Color	Hex code
Cyan/Aqua	#00FFFF
Magenta/Fuchsia	#FF00FF
Silver	#C0C0C0
Gray	#808080
Maroon	#800000
Olive	#808000
Green	#008000
Purple	#800080
Teal	#008080
Navy	#000080

6.10 Configuring the SNMP upstream interface

R&S CHM provides an SNMP-based interface that lets you query an aggregated system state from upstream monitoring solutions. The SNMP upstream interface provides information about the names, states and last-change timestamps for the overall system and its host groups. R&S CHM listens for incoming SNMP requests on port 161/udp on all available network interfaces. R&S CHM supports SNMP versions 2 and 3. The system name is "RS-CHM" (fixed).

6.10.1 Activating the interface

You can activate the SNMP upstream interface by adding the `snmp_connection` key to the master R&S CHM host instance in the `chm.yaml` file. The only available setting is the value for the SNMP `community` string, which must be passed to R&S CHM to retrieve data from the interface.

You can then establish SNMPv3 connections by using the `snmp_connection` key > `v3_users` keyword. See Section 6.10.3, "Configuring SNMPv3 traps/informs", on page 90.

For details about the state aggregation logic, see Section 6.2, "Understanding aggregated states", on page 42.

For a detailed description of the management information, see the management information base (MIB) files. All MIBs are contained in the delivery.

Example: Configuration of the SNMP upstream interface

Here, the value `testcommunity` must be used to retrieve data from the interface. The checks are omitted.

```
hosts:
  - name: host1.de
    tags: [chm]
    hostgroups: [saturn]
    snmp_connection:
      community: testcommunity
    checks:
      ...
```

In your master R&S CHM host configuration, substitute this example community string by a custom community string.

See also: `snmp_connection` on page 128

6.10.2 Configuring SNMPv2 traps

You can configure R&S CHM to inform a list of SNMP notification receivers about system status changes via SNMPv2 traps. To do so, add the `snmp_connection` > `trapreceivers` key to the master R&S CHM host instance in the `chm.yaml` file. The traps contain the system state and all host group states.

For a detailed description of the management information, see the management information base (MIB) files. All MIBs are contained in the delivery.

Example: Configuration of SNMPv2 traps

For each SNMP notification receiver, specify the host and port to which R&S CHM sends the trap. Also, specify the SNMP community that is expected by the SNMP notification receiver to accept the trap.

```
hosts:
  - name: host1.de
    displayname: CHM master
    tags: [chm]
    snmp_connection:
      community: foo
      trapreceivers:
        - host: host2.de
          port: 162
          community: bar
        - host: host3.de
          port: 1162
          community: baz
```

See also: `snmp_connection` on page 128

6.10.3 Configuring SNMPv3 traps/informs

For detailes about the SNMP keys, see `snmp_connection` on page 128.

Example: Configuration of SNMPv3 traps

SNMP configurations with one user specified under `v3_users` and multiple SNMPv2/v3 `trapreceivers`.

```
snmp_connection:
  snmp_engine_id: 80004fb80543484d63686d01
  community: public
  v3_users:
    - secname: chm_ro
      authproto: SHA-256
      authpass: VAULT:chm/snmp/chm_ro_auth
      privproto: AES-128
      privpass: VAULT:chm/snmp/chm_ro_priv
  trapreceivers:
    - host: delta-1.cobra.net # Send SNMPv2 traps
      port: 1162
      community: bar
    - host: delta-2.cobra.net # Send SNMPv2 traps
      port: 1164
      community: bar
    - host: delta-3.cobra.net # Send SNMPv3 traps
      port: 1166
      version: 3
      inform: false
      secname: chm_trap_user_v3 # Send SNMPv3 informs
      authproto: SHA-256
      authpass: VAULT:chm/snmp/trap_auth
      privproto: AES-128
      privpass: VAULT:chm/snmp/trap_priv
    - host: delta-4.cobra.net
      port: 1168
      version: 3
      inform: true
      secname: chm_trap_user_v3
      authproto: SHA-256
      authpass: VAULT:chm/snmp/trap_auth
      privproto: AES-128
      privpass: VAULT:chm/snmp/trap_priv
```

6.11 Configuring generic monitoring of SNMP third-party devices

R&S CHM provides an option to configure SNMP device checks independent from the main `chm.yaml`. Instead, you can declare the SNMP values to poll and how to

interpret them in separate YAML files. The `chm.yaml` only consists of a minimal configuration under the `snmp_flex` key where you specify the file path, see `snmp_flex` on page 179. Furthermore, R&S CHM can read-in existing YAML files from a specific directory, see Section 6.11.10, "Configuring snmp_flex in the chm.yaml", on page 106.

Using this option, R&S CHM can perform SNMP get and walk operations. It can also cast values, apply thresholds, emit performance data, and build the summary and long output. For convenience, R&S CHM installs the YAML examples. The following table lists some examples.

File	Style	Illustrates	Runnable against tests?
`generic_snmp.yaml`	Single GET, equality check	Minimal check + `on_pass`/ `on_fail`	Yes (`tests/ test_snmp_flex.py`, cisco_hardware fixture)
`gude.yaml`	Two GETs, metrics	`cast`, `scale`, perfdata, message templates	Yes (gude fixture)
`synology.yaml`	Mixed GET + multi-col walk	Enums, `status_map`, row-level checks	Yes (synology fixture)
`mac_address.yaml`	IF-MIB ifDescr + ifPhysAddress	`when` modes, `cast: mac`, field `derived:`	Yes (`nw_interface` fixture)
`interfaces_advanced.yaml`	Walk over IF-MIB	`when`, `slice`, `derived`, `filter`, `aggregate` in one file	No — needs a live switch (IF-MIB)

6.11.1 YAML file layout

A configuration is a YAML mapping with five top-level keys. Only `checks:` is necessary.

```
name: "Human-readable label"     # Optional. Prepended to the summary line.

chm_check_name: "short_name"     # Optional. Used as the chm.yaml keyword when
                                 # the file is distributed under
                       # /usr/share/rohde-schwarz/rs-monitoring-plugins/snmp_flex/.
                       # Falls back to the file's base name (no extension).
                       # See section Configuring snmp_flex in the chm.yaml.

vars:                            # Optional. Defaults for ${var} substitution.
    warn: 70
    crit: 85

enums:                           # Optional. Named integer to string maps.
    DiskStatus:
        1: NORMAL
        2: EMPTY
        3: NOT_INITIALIZED
        4: DAMAGED
        5: CRASHED
```

```
checks:                          # Mandatory, non-empty list of check blocks.
    - name: ...
      type: get | walk
      ...
```

R&S CHM validates the schema at load time. Mistakes fail before R&S CHM contacts the device. A validation error returns `UNKNOWN` with a message, such as `Schema error at 'checks/0/expect': ....`

Variable precedence

You can substitute the `vars:` block here using the `vars:` in the `chm.yaml` > `snmp_flex` check, see `snmp_flex` on page 179.

6.11.2 'get' check syntax

The `type: get` check reads a single scalar OID. The evaluation pipeline is:

```
- name: "Human label"            # Required. Used in error messages.
    type: get                    # Required.
    oid: ".1.3.6.1.4.1..."       # Required. Supports ${var} substitution.

    when: "hostname != ''"       # Optional. Skip check if expression is false.
    cast: int                    # int | float | str | hex | mac |
                                 # auto (default: auto).
    scale: 0.1                   # Optional numeric multiplier.
    enum: DiskStatus             # Optional. References a top-level enum.
    derived:                     # Optional. Extra names for expect/templates.
        wanted: 'iface_mac.replace(":", "").lower()'

    expect: { equals: 1 }        # Optional. See operator reference below.

    metric:                      # Optional.
                                 # Emits 'label=value[uom];warn;crit;min;max'.
        label: "temperature"
        uom: "C"
        warning: "70"            # pynag threshold range strings,
                                 #${var}-substituted.
        critical: "85"
        min: "0"
        max: "100"

    on_pass:                     # Optional. Applied when expect passes (or if
        summary: "OK: ${value} C"  # no expect was defined).
        long_output: "..."

    on_fail:                     # Optional. Applied when expect fails.
        status: critical           # ok | warning | critical | unknown
        summary: "Too hot: ${value} C"
```

Example: Minimal check

```
checks:
    - name: "Cisco stack switch state"
      type: get
      oid: ".1.3.6.1.4.1.9.9.500.1.2.1.1.6.1000"
      cast: int
      expect: { equals: 4 }
      on_pass: { summary: "DEVICE OK" }
      on_fail: { status: critical, summary: "DEVICE NOT OK" }
```

Output on a healthy device: "OK - DEVICE OK"

Example: Metric-only check (no expect)

```
checks:
    - name: "Gude temperature"
      type: get
      oid: ".1.3.6.1.4.1.28507.87.1.6.1.1.2.1"
      cast: float
      scale: 0.1
      metric:
        label: "temperature"
      on_pass:
        summary: "Temperature ${value} C"
```

Output: "OK - Temperature 31.6 C | 'temperature'=31.6;;;;"

If you want thresholds enforced by the YAML configuration (not displayed here), populate `metric.warning:` and `metric.critical:`. They accept the standard pynag threshold ranges (`70`, `70:`, `10:90`, `@0:9`, ...) and are `${var}`-expanded before use:

```
metric:
    label: "temperature"
    warning: "${warn}"
    critical: "${crit}"
```

Combined with `--var warn=20 --var crit=40`, a reading of 31.6 C emits "WARNING" and puts the thresholds into perfdata:

"WARNING - Temperature 31.6 C | 'temperature'=31.6;20;40;;"

Table 6-11: Cast values

`cast:`	**Result**
`auto` (default)	Leave the SNMP value as the helper returned it (`int` for integer, `str` for octet string).
`int` / `float` / `str`	Coerce with Python `int()` / `float()` / `str()`. A value that cannot be coerced surfaces as `UNKNOWN - Check error: cannot cast ....`

cast:	Result
hex	Encode the value as lowercase hexadecimal, unseparated (e.g. 000bab9e4234). bytes are hex-encoded directly; str is encoded as latin-1 first. Use this for PhysAddress-style columns when you want a compact hex string.
mac	Same encoding as hex, but with a : between every octet (e.g. 00:0b:ab:9e:42:34). Prefer this over hex when the output will be displayed to operators or compared against MAC strings typed in the familiar colon-separated form. Use this for ifPhysAddress: the helper's str() of an OctetString keeps the raw bytes (including NULs), which are not comparable to a colon-separated MAC.

Field-level 'derived:'

On a get check or a walk row check, derived: runs **after** cast: / scale: / enum: and **before** metric: / expect:. Each entry is an expression evaluated in the sandbox, see Section 6.11.6, "Expressions ('when', 'filter', 'derived')", on page 99. The result is stored under that name in the check context so ${var} templates and expect: operands can use it.

That is how mac_address.yaml normalizes iface_mac (lowercase, strip a leading 0x, replace -, ., and spaces with :) before comparing it to the mac-cast ifPhysAddress. Walk-level derived: (on the type: walk block itself) is different: it runs once per row **before** filter: and the row checks. See Section 6.11.6, "Expressions ('when', 'filter', 'derived')", on page 99.

6.11.3 'expect:' operators

Exactly one operator key per expect: block. Values on both sides go through the cast:/scale:/enum: pipeline first, so comparisons happen on native Python types.

Operator	Argument	Semantics
equals	any	value == expected (numeric-aware for int/float operands)
not_equals	any	Inverse of equals
in	list	value in expected
not_in	list	Inverse of in
lt	number/string	value < expected (numeric)
le	number/string	value <= expected (numeric)
gt	number/string	value > expected (numeric)
ge	number/string	value >= expected (numeric)
range	2-element list	lower <= value <= upper (inclusive)
regex	string	re.search(pattern, str(value)) is not None

Example: Multiple 'expect:' examples

```
expect: { in: [1, 2, 4] }
expect: { range: [10, 40] }
expect: { lt: "${threshold}" }          # threshold=5 -> "value < 5"
expect: { regex: "^[A-Z][A-Z0-9]+$" }
```

Interaction with 'enum:'

When a field declares `enum:`, both the raw integer (`${value}`) and the resolved enum name (`${value_name}`) are available. The operator picks which one to compare against based on the operand's type and not the operator name.

Operand shape	What is compared against `expected`
Single string (e.g. `"NORMAL"`)	`${value_name}` (the enum name)
Single number/bool (e.g. `1`, `true`)	`${value}` (the raw value)
List where **every** item is a string	`${value_name}`
List with any non-string item, or empty	`${value}`

So `in:` and `not_in:` **do support enums**, provided you keep the list homogeneous. Both of these are correct against a field with `enum: DiskStatus`:

```
expect: { in: [NORMAL, EMPTY] }  # all-string list -> compares to ${value_name}
expect: { in: [1, 2] }           # all-number list -> compares to ${value}
```

That is exactly what `synology.yaml` relies on: the disk `status` column has `enum: DiskStatus`, and `expect: { in: [NORMAL, EMPTY] }` matches the enum names, not the integers.

The three cases below are **not** what most users intend:

```
# Mixed list: ambiguous → falls back to raw value comparison, so "NORMAL"
# will never match (raw value is an int).
expect: { in: [1, "NORMAL"] }

# All-string list of digits: compares to ${value_name}, but the enum names
# are things like "NORMAL", not "1" — this always fails.
expect: { in: ["1", "2"] }

# Numeric operators never route through the enum name (enum names aren't
# numeric): "READY" is not evaluated against 10 here.
expect: { lt: 10 }                # always compares ${value} < 10
```

`lt`, `le`, `gt`, `ge`, and `range` always compare against `${value}` because enum names are not numeric.

Operands are `${var}`-expanded *before* this routing decision, so `in: ["${wanted}", "OK"]` behaves as an all-string list once `${wanted}` is substituted, and it compares against `${value_name}` on an enum field.

6.11.4 'on_pass:' and 'on_fail:' outcomes

Both blocks have the same shape:

```
on_fail:
    status: critical                # or ok | warning | unknown
    status_map:                     # Alternative to status:. See below.
        NORMAL: ok
        DAMAGED: critical
        CRASHED: critical
        default: warning
    summary: "Disk ${row_index}: ${value_name}"
    long_output: "..."
```

Rules:

- Only one of `status:` or `status_map:` is used per outcome. `status_map:` is consulted first; it looks up `${value_name}` (or `${value}` if there is no enum), falling back to `default:`.
- Status escalation only. The overall check starts at "OK" and only ever moves toward the worst status seen (pynag semantics). So `on_pass: { status: warning }` *can* nudge "OK" -> "WARNING", but it can never demote a prior "CRITICAL".
- Omitting `status:`/`status_map:` means that the outcome only emits messages. It does not touch the status. This behavior is typical for `on_pass:` in metric-only checks.
- `summary:` appends to the first output line; `long_output:` appends to the multi-line detail section.

'status_map' in practice

`status_map:` is convenient for encoding vendor-specific severity tables:

```
- field: status
    cast: int
    enum: DiskStatus
    expect: { in: [NORMAL, EMPTY] }
    on_fail:
      status_map:
          DAMAGED: critical
          CRASHED: critical
          default: warning
      summary: "Disk ${row_index}: ${value_name}"
```

A `DAMAGED` disk emits "CRITICAL", `NOT_INITIALIZED` emits "WARNING" (default) and both share the message template.

6.11.5 Template variables

Every string field that supports templating uses the same `${name}` syntax:

- `on_pass.summary`, `on_pass.long_output`, `on_fail.summary`, `on_fail.long_output`
- `metric.label`, `metric.uom`, `metric.warning`, `metric.critical`, `metric.min`, `metric.max`
- `oid:` on `get` and `walk` checks (including each entry of `columns:`)
- Any string operand of an `expect:` operator, including items inside `in:` / `not_in:` lists and either bound of `range:`
- `row.label` and `row.long_output` inside a walk

Rules that apply everywhere:

- Placeholder identifiers follow Python names (`[A-Za-z_][A-Za-z0-9_]*`). Dotted paths, indexing, and expressions are not supported here – use `derived:` if you need computation before rendering.
- Unknown names, and names whose value is `None`, expand to the **empty string**. Templates never raise a `KeyError`. This is deliberate so a missing `--var` shows up as a visibly empty message instead of crashing the check.
- Substitution happens after `cast:` and `scale:` (and after field-level `derived:`) but before the value is handed to `expect:` and `metric:`, so `${value}` always reflects the final Python-native value.

Scope reference

The available names depend on where the template is evaluated. The following table groups them by scope.

Scope	Name	Type	Meaning
Always (top-level context)	`hostname`	str	Value of `-H` on the CLI. Empty string if `-H` was omitted.
Always	any `vars:` key	str	Every key from `vars:` and `--var KEY=VALUE` (CLI wins). Values are strings until `cast:` runs.
`get` check *and* walk row	`value`	any	Current field's value after `cast:` and `scale:`. Also the SNMP raw value if neither is set.
`get` check *and* walk row	`value_name`	str / None	Enum-resolved name for `value`. Present only when the check declares `enum: <Enum>`. `None` if the value is not in the enum.
`get` check *and* walk row check	any field-level `derived:` key	any	Result of a `derived:` expression on that `get` or row check. Computed after `cast:`/`scale:`/`enum:`, before `expect:`.
Walk row (any template)	`row_index`	int	1-based row number *after* `slice:`. First surviving row is 1.

Scope	Name	Type	Meaning
Walk row (any template)	any column name	any	Raw SNMP value from the column named in `columns:` (or `column:` for the single-column form).
Walk row (any template)	any walk-level `derived:` key	any	Result of a `derived:` expression on the `type: walk` block. Computed once per row, before `filter:` and row checks run.
Walk row *after* a row-check ran	`<column>`	any	The column value is overwritten with the row check's cast/scaled `value`.
Walk row *after* a row-check ran	`<column>_name`	str	Enum-resolved name for that column, if its row check declared `enum:`.
Walk row (`row.long_output`)	`row_label`	str	Rendered `row.label` template. Only set if `row.label:` is defined.
`aggregate.count_passing.on_pass` / `on_fail`	`passing_count`	int	Number of rows where every row check passed.
`aggregate.count_passing.on_pass` / `on_fail`	`value`	int	Alias of `passing_count`, so generic `${value}` templates work here too.

Walk overwrite behavior

Each one reads its column via `field:`, and when it finishes the possibly cast or scaled `value` is written back to the row context under the column name. That is why `row.long_output: "... ${status_name} ..."` works in `synology.yaml` – by the time the row's `long_output` renders, the earlier `status` row check has already published `${status_name}`.

Worked rendering example

Given this YAML fragment inside a walk over a NAS disk table, and given `--var min_temp=15`:

```
name: "NAS"
enums:
    DiskStatus: { 1: NORMAL, 4: DAMAGED }
vars:
    warn_c: 60
checks:
    - name: "Disks"
      type: walk
      columns:
        status: "1.3.6.1.4.1.6574.2.1.1.5"
        temp:   "1.3.6.1.4.1.6574.2.1.1.6"
      row:
        label: "Disk ${row_index} on ${hostname}"
        long_output: "${row_label}: ${status_name} / ${temp} C"
        checks:
```

```
  - field: status
    cast: int
    enum: DiskStatus
    expect: { equals: NORMAL }
    on_fail:
      summary: "${row_label} is ${value_name}"
  - field: temp
    cast: int
    metric:
      label: "${row_label} temp"
      uom: "C"
      warning: "${warn_c}"
      critical: "70"
```

For the second row of a device polled with `-H nas1.example.net`, where SNMP returned `status=4, temp=68`, the templates resolve as listed in the following table.

Template	Renders to	Why
`row.label`	`Disk 2 on nas1.example.net`	`row_index=2`, `hostname` from `-H`.
`on_fail.summary` (status)	`Disk 2 on nas1.example.net is DAMAGED`	`row_label` from previous step; `value_name=DAMAGED` (enum lookup on 4).
`metric.label` (temp)	`Disk 2 on nas1.example.net temp`	Same `row_label` reused.
`metric.warning` (temp)	`60`	`${warn_c}` came from top-level `vars:`.
`row.long_output`	`Disk 2 on nas1.example.net: DAMAGED 68 C`	After the status row check, `${status_name}` is available; `${temp}` was overwritten with the cast-to-int value.

6.11.6 Expressions ('when', 'filter', 'derived')

These fields accept a Python-like expression, evaluated in a sandbox.

- `when:` on a top-level check – skip the whole check when it evaluates false.
- `filter:` on a `walk` – skip rows where it evaluates false.
- `derived:` on a `walk` – compute extra row-context variables before `filter:` and the row checks.
- `derived:` on a `get` or row check – extra names **after** `cast:`/`scale:`/`enum:`, available to `expect:` and the outcome templates. Not the same as walk-level `derived:`. See the `mac_address.yaml` file for a field-level example.

The sandbox exposes a small allow list:

- **Functions:** `abs`, `bool`, `float`, `int`, `len`, `max`, `min`, `round`, `str`.
- **`str` methods:** `startswith`, `endswith`, `lower`, `upper`, `strip`, `lstrip`, `rstrip`, `split`, `replace`, `count`, `find`, `isdigit`, `isalpha`, `isalnum`.
- **Everything else** (attribute access, `import`, `open`, ...) is rejected with a `Config error: ...` message.

Example:

Example patterns:

```
- name: "Only on this appliance"
    type: get
    when: 'hostname.startswith("pdu-")'
    oid: "..."

- name: "Disks"
    type: walk
    columns:
        status: "1.3.6.1.4.1.6574.2.1.1.5"
        temp:   "1.3.6.1.4.1.6574.2.1.1.6"
    filter: 'int(status) != 2'          # skip EMPTY slots
    derived:
        overheating: 'int(temp) > 55'
```

Field-level `derived:` (on a `get` or row check) runs later, after that field's `cast:`/`scale:`/`enum:`. The `mac_address.yaml` file uses it to normalize a MAC before `expect:`:

```
- name: "ifPhysAddress match"
    type: get
    oid: "1.3.6.1.2.1.2.2.1.6.${iface_no}"
    cast: mac
    derived:
      wanted_mac: \
        'iface_mac.lower().replace("0x", "").replace("-", ":") \
        .replace(".", ":").replace(" ", ":")'
    expect: { equals: "${wanted_mac}" }
```

6.11.7 'type: walk' – table-oriented checks

Walks turn one or more SNMP columns into per-row checks. The columns are zipped positionally into rows, so every column must return the same number of values. Otherwise, "UNKNOWN" is shown on the web GUI with a misalignment message instead of pairing up unrelated indices.

Walk field reference

Every top-level field of a `type: walk` check is listed below. Only `name`, `type`, and one of `oid` or `columns` are required. The rest is optional.

Field	Required?	Purpose
`name`	yes	Human label. Used in the misalignment error and elsewhere.
`type`	yes	Must be the literal `walk`.
`oid`	one of oid/columns	Single-column form. Combined with optional `column:` to name the value in the row context (default column name: `value`).

Field	Required?	Purpose
`column`	no (with `oid`)	Name under which the walked value is exposed in each row. See "Single-column form" on page 102.
`columns`	one of oid/columns	Multi-column form: mapping of `name -> OID`. Every column is walked and the results are zipped positionally into rows.
`when`	no	Expression evaluated against the top-level context. If false, the whole walk is skipped (device is not polled).
`slice`	no	`{ offset: N, max_rows: M }`. Applied to the raw walk result before row processing. See "Slicing rows" on page 103.
`filter`	no	Expression evaluated against the row context. Rows where it is false are skipped (nothing emitted, not counted).
`derived`	no	Mapping of `name -> expression`. Each expression is evaluated against the row context and its result stored under `name`.
`row`	no	Per-row rendering + row-level `checks:` list. See "Multi-column form" on page 102.
`aggregate`	no	Optional post-loop verdict driven by the number of passing rows. See "Aggregate: count passing rows" on page 103.

`when:`, `filter:`, and each `derived:` expression share allowed functions, same allowed `str` methods, same rejection rules.

Execution order

For a single `type: walk` check, the engine runs the following steps in this exact order. Naming the order matters because `slice:` narrows the row set before `filter:` sees it, and `derived:` is available to `filter:` and to every row check that follows.

1. Evaluate `when:` against the top-level context. If false, skip the whole check — no SNMP traffic.
2. Walk each column via SNMP (`oid` or every entry of `columns:`). Every column is walked once, in schema order.
3. Assert that the columns are aligned (same row count). "UNKNOWN" otherwise.
4. Apply `slice:` (`offset` first, then `max_rows`) to the raw column data.
5. For each surviving row, in walk order:
   a) Build the row context: `${row_index}` (1-based, 1 = first row after `slice:`), one entry per column, plus any `hostname` / `vars:` / `--var` names from the outer context.
   b) Evaluate every `derived:` expression against the row context and store each result under its name in the row context.
   c) Evaluate `filter:` against the row context. If false, skip the rest of this row (no `row.label`, no row checks, no `row.long_output`, does not count toward `passing_count`).
   d) If `row.label:` is set, render it and store the string as `${row_label}`.

e) For each entry in `row.checks[]`, in declared order: cast → scale → enum → field-level `derived:` → metric → expect → `on_pass` or `on_fail`. Fold the (possibly cast/scaled) value back into the row context under the column name; if the row check used `enum:`, publish `<column>_name` too. Field-level `derived:` names stay on that field's context; they are not written back to the row.
f) If `row.long_output:` is set, render it against the final row context.
g) If every row check passed (or if there were no row checks), increment `passing_count`.

6. If `aggregate.count_passing:` is set, evaluate its `expect:` against `passing_count` and emit `on_pass:` or `on_fail:`. `${passing_count}` (and its alias `${value}`) are available in these outcomes.

For a single file that exercises `when:`, `slice:`, `derived:`, `filter:`, and `aggregate:` together, with heavy line-by-line comments, see the `interfaces_advanced.yaml` file.

Single-column form

```
- name: "Interface admin status"
    type: walk
    oid: ".1.3.6.1.2.1.2.2.1.7"
    column: state              # exposes ${state} in the row context
    row:
        checks:
            - field: state
                cast: int
                expect: { equals: 1 }
                on_fail:
                    status: warning
                    summary: "Interface ${row_index}: admin down"
```

Multi-column form

```
- name: "Disks"
    type: walk
    columns:
        status: "1.3.6.1.4.1.6574.2.1.1.5"
        temp:   "1.3.6.1.4.1.6574.2.1.1.6"
    row:
        long_output: "Disk ${row_index}: ${status_name}, ${temp} C"
        checks:
            - field: status
                cast: int
                enum: DiskStatus
                expect: { in: [NORMAL, EMPTY] }
                on_fail:
                    status_map:
```

```
                    DAMAGED: critical
                    CRASHED: critical
                    default: warning
                summary: "Disk ${row_index}: ${value_name}"
        - field: temp
            cast: int
            metric:
                label: "Disk ${row_index} Temperature"
```

Note the row-scoped fields:

- `row.label:` – a rendered template stored as `${row_label}` and reusable in later row templates.
- `row.long_output:` – appended once per row after all row checks have run.
- `row.checks[].field:` – the column name to read (from `column:` or `columns:`). The rest of the field mirrors a `get` check.

After a row check runs, the possibly cast or scaled value is written back to the row context. If the field had an `enum`, `${<field>_name}` is also available – that is why `long_output: "... ${status_name} ..."` works above.

Slicing rows

Some tables are large but only the first N entries matter:

```
slice:
    offset: 0
    max_rows: 24
```

Both keys are optional (`offset` defaults to 0). `slice:` runs *before* the `filter:` expression.

Aggregate: count passing rows

Use `aggregate.count_passing` to fail based on how many rows survived all their row checks:

```
- name: "Fans"
    type: walk
    oid: ".1.3.6.1.4.1.9.9.13.1.4.1.3"
    column: state
    row:
        checks:
            - field: state
                cast: int
                expect: { equals: 1 }
    aggregate:
        count_passing:
            expect: { ge: "${min_fans}" }
            on_fail:
                status: critical
                summary: "Only ${passing_count} fan(s) healthy, need ${min_fans}"
```

`${passing_count}` is exposed only inside `count_passing` outcomes.

6.11.8 Full worked example – Synology NAS

The shipped `synology.yaml` file combines every feature that you are likely to need in a production check. Annotated view:

```
name: "Synology NAS"           # Prefix for the summary line.

enums:                          # Reusable name table for status codes.
    DiskStatus:
        1: NORMAL
        2: EMPTY
        3: NOT_INITIALIZED
        4: DAMAGED
        5: CRASHED

checks:
    # --- Single GET, equality guard: escalate to critical on failure.
    - name: "System Partition"
      type: get
      oid: "1.3.6.1.4.1.6574.1.1.0"
      cast: int
      expect: { equals: 1 }
      on_fail: { status: critical, summary: "System Partition Status: Failed" }
      on_pass: { long_output: "System Partition Status: Normal" }

    # --- Single GET, published as perfdata; no expect, so it's a
    # metric-only check.
    - name: "System Temperature"
      type: get
      oid: "1.3.6.1.4.1.6574.1.2.0"
      cast: int
      metric:
        label: "System Temperature"
      on_pass: { long_output: "System temperature is ${value} C" }

    # --- Multi-column walk with enum, status_map, and row-level long_output.
    - name: "Disks"
      type: walk
      columns:
        status: "1.3.6.1.4.1.6574.2.1.1.5"
        temp:   "1.3.6.1.4.1.6574.2.1.1.6"
      row:
        checks:
          - field: status
            cast: int
            enum: DiskStatus
            expect: { in: [NORMAL, EMPTY] }
```

```
      on_fail:
        status_map:
          DAMAGED: critical
          CRASHED: critical
          default: warning
        summary: "Disk ${row_index} Status: ${value_name}"
    - field: temp
      cast: int
      metric:
        label: "Disk ${row_index} Temperature"
  long_output: "Disk ${row_index} Status: ${status_name}, \
         Temperature: ${temp} C"
```

Running it against a NAS with one CRASHED disk yields:

```
Critical - Synology NAS. Disk 8 Status: CRASHED | 'System Temperature'=42;;;; ...
System Partition Status: Normal
System temperature is 42 C
Disk 1 Status: NORMAL, Temperature: 31 C
...
Disk 8 Status: CRASHED, Temperature: 42 C
```

6.11.9 Full worked example – IF-MIB ifTable

The shipped `mac_address.yaml` polls **only** `ifDescr` and `ifPhysAddress` from IF-MIB `ifTable` (RFC 2863). The mode is selected with `--var`; `debug` and `iface_no` / `iface_mac` are mutually exclusive.

Table 6-12: IF-MIB ifTable

`--var`	**SNMP**	**Status**
`debug=1`	Walk `ifDescr` + `ifPhysAddress`; every row is long output	always "OK"
`iface_no=N`	GET `ifDescr.N` and `ifPhysAddress.N`	always "OK"
`iface_no=N` and `iface_mac=...`	Same GET, then compare the MAC	"OK" if equal "CRITICAL" if not
`iface_mac` without `iface_no`	none (mode error)	"UNKNOWN"
`debug` together with `iface_no` or `iface_mac`	none (mode error)	"UNKNOWN"
neither `debug` nor `iface_no`	none (mode error)	"UNKNOWN"

`iface_no` is the IF-MIB **ifIndex** (the instance suffix on the OID), not the 1-based walk `${row_index}`. `iface_mac` comparison is case-insensitive; a leading `0x` is stripped and `-`, `.`, and spaces are normalized to `:`.

`ifPhysAddress` is a `PhysAddress` (raw octets). The config uses `cast: mac` so the value is rendered as colon-separated lowercase hex (e.g. `00:0b:ab:9e:42:34`), and

a field-level `derived:` normalizes the operator-supplied MAC to the same form before `expect:`.

6.11.10 Configuring snmp_flex in the chm.yaml

There are two ways to invoke a generic YAML config from the `chm.yaml` file.

See also: snmp_flex on page 179

Distributed configs – one keyword per file (recommended)

R&S CHM scans every YAML in the following directory on startup and registers it as its own service:

`/usr/share/rohde-schwarz/rs-monitoring-plugins/snmp_flex/`

The keyword you write in `chm.yaml` file equals the file's `chm_check_name:`. Otherwise, if that key is absent, it equals the file's base name without extension. The display name defaults to `name:` (fallback: `chm_check_name`).

For example, `mac_address.yaml` declares:

```
name: "Network interface MAC addresses"
chm_check_name: "mac_address"
vars:
    debug: ""
    iface_no: ""
    iface_mac: ""
```

So, `chm.yaml` addresses the check by the short name `mac_address` and passes overrides as arguments:

```
checks:
    - mac_address:
        iface_no: 2
        iface_mac: "00:11:22:33:44:55"
```

Rules:

- Only keys declared in the config's `vars:` block are accepted. An unknown key surfaces as `Plugin <chm_check_name>: Unknown plugin attribute '<key>'` when R&S CHM regenerates its config, so typos fail fast instead of silently being dropped.
- If two distributed configs resolve to the same `chm_check_name` – or if one would collide with a built-in R&S CHM system status monitoring service – the later config is skipped with a warning; the built-in keyword always wins.
- SNMP transport and credentials (community, version, SecName, ...) are taken from the host's SNMP connection, exactly like every other SNMP-based R&S CHM service. There is no per-check credential override.

Base 'snmp_flex' service – arbitrary paths and inline configs

For configs that are not distributed (custom paths under `/etc`, inline YAML embedded directly in `chm.yaml`, ...), use the built-in `snmp_flex` keyword with either `config:` (path) or `config_content:` (inline mapping):

```
checks:
    # Absolute path on the monitoring node.
    - snmp_flex:
        config: /etc/rohde-schwarz/snmp_flex/mydevice.yaml
        vars:
            - warn=70
            - crit=85

    # Bare file name -> resolved under the distributed config dir.
    - snmp_flex:
        config: mac_address.yaml
        vars:
            - iface_no=2

    # Inline YAML; CHM writes the file and points the service at it.
    - snmp_flex:
        config_content:
            name: "My device"
            vars:
                threshold: 90
            checks: []
        vars:
            - threshold=80
```

`config:` and `config_content:` are mutually exclusive; exactly one is required. `vars:` on this R&S CHM service is a list of raw `KEY=VALUE` strings passed straight through as `--var`, so the R&S CHM service does not validate them against the config's declared `vars:`. Prefer the per-file keyword form above when the config is distributed.

6.11.11 Troubleshooting

Symptom	Cause
"UNKNOWN - Schema error at 'checks/0/expect': ..."	The YAML doesn't match the schema (typo, wrong type). The path points at the offending block.
"UNKNOWN - Config error: Unknown expect operator: '...'"	Operator name misspelled; see Table 6-12.
"UNKNOWN - Config error: Reference to undefined enum '...'"	The check references an enum name that is not in the top-level `enums:` map.
"UNKNOWN - Check error: cannot cast value 'foo' to int: ..."	Device returned a non-numeric value; either fix `cast:` or add a `when:` guard.
"UNKNOWN - Check error: cannot cast value ... to hex/mac: ..."	`cast: hex` or `cast: mac` could not latin-1-encode the Octet-String; the value is not a PhysAddress-like byte string.

Symptom	Cause
"UNKNOWN - Walk '...' returned misaligned columns (a=8, b=7)"	The walked columns do not cover the same rows; use `columns:` that share an index, or narrow.
"UNKNOWN - Unresolved placeholder in OID template: '...'"	`${var}` in an OID references a name that is not in `vars:/--var/hostname`.
"UNKNOWN - ... debug and iface_no/iface_mac are mutually exclusive"	`mac_address.yaml`: do not set `debug` together with `iface_no` or `iface_mac`.
"UNKNOWN - ... iface_mac requires iface_no"	`mac_address.yaml`: `iface_mac` is only valid together with `iface_no`.
"UNKNOWN - ... set debug or iface_no"	`mac_address.yaml`: pick a mode; neither variable was set.
MAC compare is always "CRITICAL", values look nothing like a MAC	Missing `cast: hex` or `cast: mac` on `ifPhysAddress`; `str()` of the OctetString keeps the raw bytes as latin-1 and will not match a hex/MAC string.
Fine YAML, but no perfdata	The check has no `metric:` block, or `label:` is empty.

For live debugging, start from a copy of the example nearest to your device and iterate. The JSON Schema validator gives you a JSON-Pointer-style location on every failure, so you can jump straight to the misconfigured node.

6.12 Configuring distributed monitoring

This chapter helps you configure different variants of system status monitoring. Distributed monitoring means that you configure multiple R&S CHM instances that either monitor other hosts or devices, or that send monitoring results to R&S CHM hosts.

Such configurations can comprise a second, redundant R&S CHM host or multiple R&S CHM hosts that are distributed over different subsystems. A subsystem is at least one R&S CHM node that is grouped with any number of non-R&S CHM hosts or devices, or both. Each R&S CHM host instance in a subsystem provides its own web GUI.

In the following description, involved R&S CHM instances are named by their role in the status monitoring system.

- A **master** is an R&S CHM host instance that is located in the top-level subsystem. A master receives the results of all checks that are executed by itself and the check results from subordinated satellites and agents.
- A **satellite** is an R&S CHM host instance that is not placed in the top-level subsystem. A satellite sends its check results to configured master instances. The web GUI of the satellite only shows the check results of the satellite and subordinated agents. Check results from other satellite instances or master instances are unavailable.
- An **agent** is an R&S CHM agent instance on Linux or Windows hosts. An agent only checks itself and sends the results to a parent satellite or master. An agent does not provide an own web GUI.

Simple monitoring configuration

Typically, you configure a monitoring setup that comprises a single master R&S CHM host and multiple Linux and Windows agents.

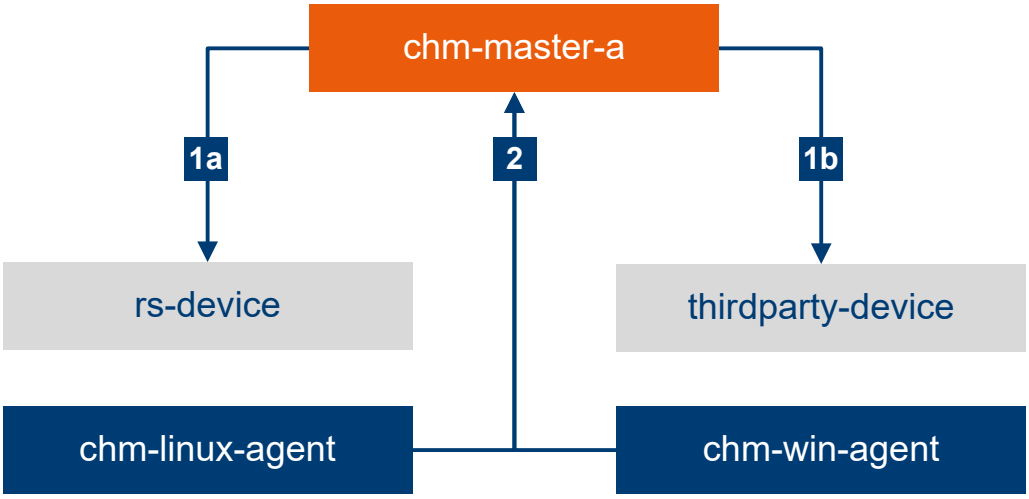


Figure 6-15: Simple monitoring configuration

1a, 1b = Master monitors devices.
2 = Agents send monitoring results to master.

The previous figure shows a monitoring configuration where a master, i.e. the R&S CHM host, monitors all kinds of devices and hosts that are not acting as an agent (**1a, 1b**). The agents monitor the hosts on which they are installed. The agents send their monitoring results to the R&S CHM host (**2**). In this configuration, the web GUI of the R&S CHM host shows all monitored hosts and services.

The following chapters explain how you configure monitoring variants that use multiple R&S CHM host instances in parallel.

See also: Section 5.5, "Deploying certificates for distributed monitoring", on page 35

- Configuring high availability monitoring.....109
- Configuring subsystems.....112
- Configuring multi-level monitoring.....113
- Configuring multi-level HA monitoring.....118

6.12.1 Configuring high availability monitoring

For high availability (HA) monitoring, you configure a second R&S CHM host as a **secondary master**. Such a configuration provides the following features:

- **Data synchronization**: Both masters synchronize all monitoring data between each other, and they let you view to whole system state independently.
- **Data duplication**: Both masters save all monitoring data to their own local database. Due to this mechanism, you can profit from an automatically created backup.
- **Failover**: If one master becomes unavailable, the R&S CHM agents automatically send their monitoring data to the remaining, intact master.

The automatic failover procedure avoids a single point of failure for receiving the check results from R&S CHM agents at master level.

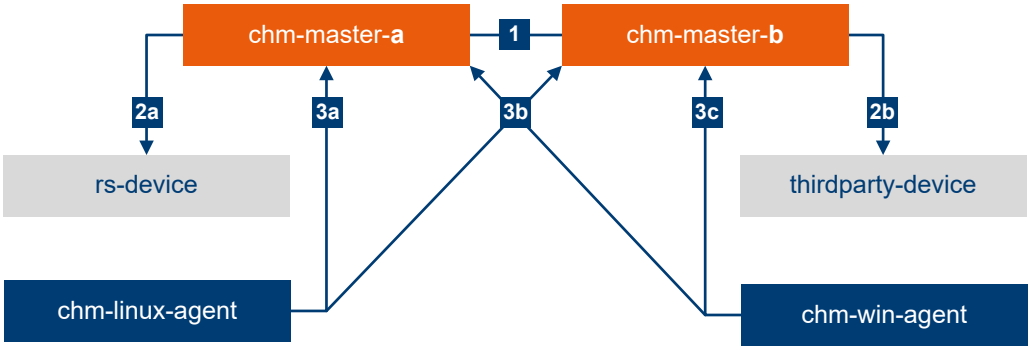


Figure 6-16: High availability monitoring

1 = Synchronization of monitoring results between masters.
2a, 2b = Masters monitor devices.
3a, 3b, 3c = Agents send monitoring results to masters.

To set up a HA monitoring system

1. On both masters, install the R&S CHM host software.
   How to: Section 4, "Installing R&S CHM", on page 18
2. Install certificates and keys.
   How to: Section 5.5, "Deploying certificates for distributed monitoring", on page 35
3. Edit the `chm.yaml` file to describe the HA monitoring configuration.
   How to: Section 6.12.1.1, "Editing the YAML configuration for HA monitoring", on page 110.
4. Both masters require an identical `chm.yaml` file. Save this file here:
   `/etc/opt/rohde-schwarz/chm/`
5. Restart the chm service on both masters to take the changes effect:
   `systemctl restart chm`.

6.12.1.1 Editing the YAML configuration for HA monitoring

HA monitoring configurations require two R&S CHM host instances, one instance serves as the primary master the other instance serves as the secondary master.

1. Specify the entries in the `chm.yaml` file for the HA monitoring configuration.
   a) Under the `hosts` configuration of the primary master, add the host configuration of the secondary master.
   b) Configure the secondary host as the high availability master:
   `tags: ["icinga2_ha"]`
2. Except for masters or agents, you can configure a relationship to the secondary master. To do so, add this key:

```
    checked_by: "<HA-master-fqdn>"
```

Example: YAML configuration: HA monitoring

This example:

- Uses the host names from Figure 6-16
- Omits any checks for clarity

```
hosts:
  # primary master
  - name: "chm-master-a"
    tags: ["chm"]

  # secondary master
  - name: "chm-master-b"
    tags: ["icinga2_ha"]

  # linux agent
  - name: "chm-linux-agent"
    connections: ["icinga2_linux"]

  # windows agent
  - name: "chm-win-agent"
    connections: ["icinga2_win"]

  # devices
  - name: "rs-device"
  - name: "thirdparty-device"
    checked_by: "chm-master-b"
```

6.12.1.2 Configuring R&S CHM agents for HA monitoring

It is necessary that you inform the agents about the existence of both masters.

► Run these scripts to complete agent configuration:

- On Linux agents, run the `chm_node_setup` shell script.
- On Windows agents, run the `chm-node-setup.bat` batch script.

For parameterization see the following examples that use the FQDNs from Figure 6-16.

Example:

Script on the Linux agent **chm-linux-agent**:

```
chm_node_setup \
--subsys chm-linux-agent \
--parent-subsys chm-master-a \
--parent-chm chm-master-a \
--second-parent-chm chm-master-b
```

Example:

Script on the Windows agent **chm-win-agent**:

```
"C:\Program Files\chm\chm-node-setup.bat" \
--subsys chm-win-agent \
--parent-subsys chm-master-a \
--parent-chm chm-master-a \
--second-parent-chm chm-master-b
```

6.12.2 Configuring subsystems

You can subdivide a status monitoring system into multiple subsystems for multi-level monitoring purposes. Subsystems then define the structure of the overall system. Also, subsystems define the relations between hosts, i.e. the hosts that are directly monitored by an R&S CHM host and the R&S CHM hosts that synchronize check results.

How to configure subsystems:

- Section 6.12.3, "Configuring multi-level monitoring", on page 113
- Section 6.12.4, "Configuring multi-level HA monitoring", on page 118

In the `chm.yaml` file, you add subsystems on the same indentation level as hosts.

---

subsystems (Subsystems for multi-level monitoring)

Defines the subsystems of the status monitoring system.

Parameters:

name	string Specifies the name of the subsystem.
hosts	list of strings Specifies the members of a subsystem using their host names.
parent_subsystem	For subordinated subsystems only, specify the related higher-level subsystem.

Example: This example shows a small excerpt for orientation purposes.

```
subsystems:
  - name: "A"
    hosts:
      - "chm-master-a"

  - name: "B"
    hosts:
      - "chm-satellite-b"
      - "rs-device"
    parent_subsystem: "A"

  - name: "C"
    hosts:
      - "chm-satellite-c"
      - "thirdparty-device"
    parent_subsystem: "A"
```

For comprehensive examples, see the following chapters.

6.12.3 Configuring multi-level monitoring

For multi-level monitoring, you configure more than one R&S CHM instance in a tree-like structure with three or more monitoring levels. A multi-level configuration provides these features:

- **Subsystem monitoring**: Split up the system into subsystems. Each R&S CHM node only monitors its subtree of the system.
- **Monitoring of remote systems**: For distant system components, a tree-like configuration reduces network traffic between remote locations and also helps reduce the load on the top-level master.

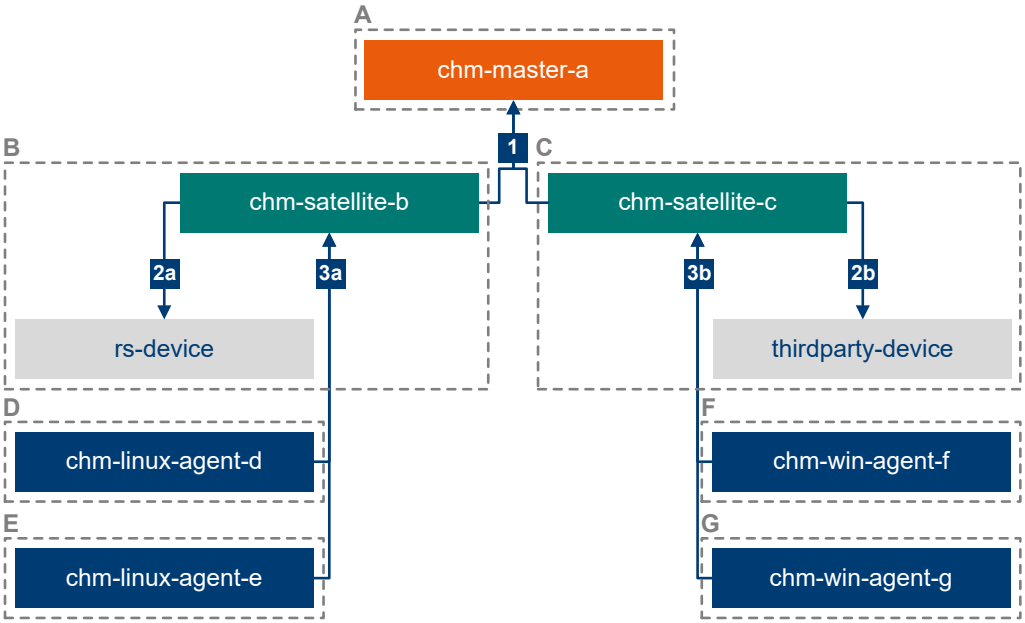


Figure 6-17: Multi-level (three-level) monitoring example

1 = Satellites send subsystem monitoring results to master.
2a, 2b = Satellites in subsystems monitor devices.
3a, 3b = Agents send monitoring results to satellites.

The status monitoring system in the previous figure is subdivided in subsystems **A** to **G**.

In sum, the system contains three R&S CHM host instances, i.e. one each in subsystems A, B and C. The R&S CHM hosts adopt the following roles:

- The **master** is the R&S CHM host instance in top-level subsystem **A**.
- The **satellites** are R&S CHM host instances in second-level subsystems **B** and **C**.

Each R&S CHM host instance provides its own web GUI. Thus, you can monitor the following on these web GUIs:

- chm-master-a (subsystem **A**): Monitor the whole system.
- chm-satellite-b (subsystem **B**): Monitor subsystems **B**, **D** and **E**.
- chm-satellite-c (subsystem **C**): Monitor subsystems **C**, **F** and **G**.

The remaining R&S CHM nodes are four R&S CHM agents, i.e. one each in subsystems **D**, **E**, **F** and **G**.

To set up a multi-level monitoring system

1. On the master and the satellites, install the R&S CHM host software.
   How to: Section 4, "Installing R&S CHM", on page 18

2. On each other host that masters or satellites cannot monitor with external checks, install the agent software.
   How to: Section 4.2, "Installing R&S CHM agents", on page 20

3. Install certificates and keys.

How to: Section 5.5, "Deploying certificates for distributed monitoring", on page 35

4. Edit the `chm.yaml` file to describe the multi-level monitoring architecture.
   How to: Section 6.12.3.1, "Editing the YAML configuration for multi-level monitoring", on page 115.
5. All masters and satellites require an identical `chm.yaml` file. Save this file here: `/etc/opt/rohde-schwarz/chm/`
6. Restart the chm service on all masters and satellites to take the changes effect:
   `systemctl restart chm`
   We recommend starting the service in sequence on the master and then on the satellites.
7. On each agent, run the node setup scripts with options that describe the multi-level system. See Section 6.12.3.2, "Configuring agents for multi-level monitoring", on page 117.

6.12.3.1 Editing the YAML configuration for multi-level monitoring

Multi-level monitoring configurations require that you configure the `subsystems` key above the `hosts` key.

► Specify the entries in the `chm.yaml` file for the multi-level monitoring configuration:

- The names of all subsystems
- The members of the subsystems, i.e. masters, satellites or agents, or monitored hosts or devices
- Exactly one parent subsystem except for the top-level subsystem

Example: YAML configuration: multi-level monitoring

A satellite always requires a R&S CHM host installation on Linux. Thus, the satellites require the `connections: ["icinga2_linux"]` key.

This example:

- Uses the host names from Figure 6-17
- Omits any checks for clarity

```
subsystems:
  - name: "A"
    hosts:
      - "chm-master-a"

  - name: "B"
    hosts:
      - "chm-satellite-b"
      - "rs-device"
    parent_subsystem: "A"

  - name: "C"
    hosts:
      - "chm-satellite-c"
      - "thirdparty-device"
    parent_subsystem: "A"

  - name: "D"
    hosts:
      - "chm-linux-agent-d"
    parent_subsystem: "B"

  - name: "E"
    hosts:
      - "chm-linux-agent-e"
    parent_subsystem: "B"

  - name: "F"
    hosts:
      - "chm-win-agent-f"
    parent_subsystem: "C"

  - name: "G"
    hosts:
      - "chm-win-agent-g"
    parent_subsystem: "C"

hosts:
  # master in A
  - name: "chm-master-a"
    tags: ["chm"]
```

```
# satellite in B
- name: "chm-satellite-b"
  connections: ["icinga2_linux"]

# satellite in C
- name: "chm-satellite-c"
  connections: ["icinga2_linux"]

# linux agent in D
- name: "chm-linux-agent-d"
  connections: ["icinga2_linux"]

# linux agent in E
- name: "chm-linux-agent-e"
  connections: ["icinga2_linux"]

# linux agent in F
- name: "chm-win-agent-f"
  connections: ["icinga2_win"]

# linux agent in G
- name: "chm-win-agent-g"
  connections: ["icinga2_win"]

# devices
- name: "rs-device"
- name: "thirdparty-device"
```

6.12.3.2 Configuring agents for multi-level monitoring

It is necessary that you inform the agents about these relations:

- The own subsystem.
- The parent subsystem.
- The connection to master or satellite.

To configure the agents

► Run these scripts to complete agent configuration:

- On Linux agents, run the `chm_node_setup` shell script.
- On Windows agents, run the `chm-node-setup.bat` batch script.

For parameterization, see the following examples that use the FQDNs from Figure 6-17.

Example:

Script on the Linux agent **chm-linux-agent-d** (subsystem **D**):

```
chm_node_setup \
--subsys D \
--parent-subsys B \
--parent-chm chm-satellite-b
```

Example:

Script on the Windows agent **chm-win-agent-f** (subsystem **F**):

```
"C:\Program Files\chm\chm-node-setup.bat" \
--subsys F \
--parent-subsys C \
--parent-chm chm-satellite-c
```

6.12.4 Configuring multi-level HA monitoring

For this advanced usage scenario, you combine multi-level and high availability monitoring. This combination lets you realize, for example, a primary master that synchronizes all information with a distant secondary master. This usage scenario combines the features from the "pure" multi-level or HA monitoring configurations.

The following figure shows an example for such a multi-level, HA monitoring configuration.

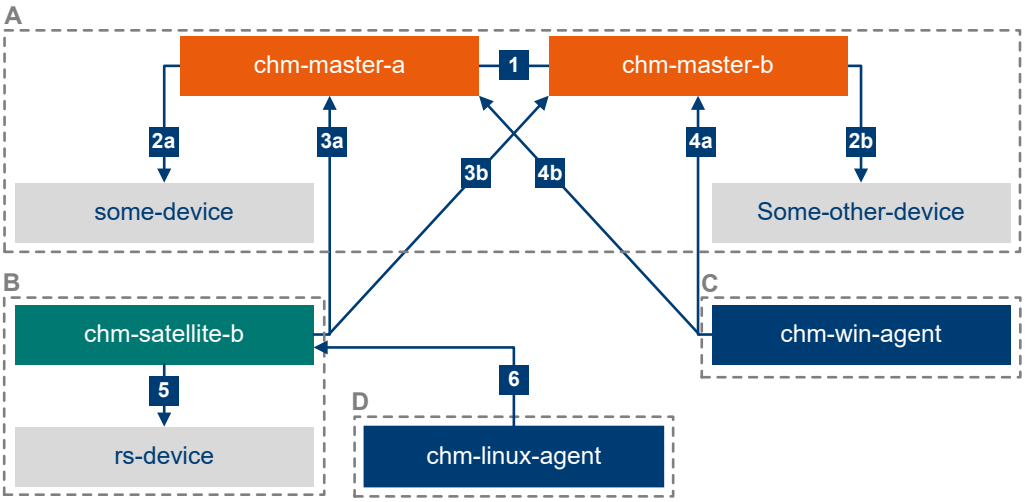


Figure 6-18: Multi-level, HA monitoring example

1 = Synchronization of monitoring results (HA configuration).
2a, 2b = Masters monitor devices.
3a, 3b = Satellite sends monitoring results to masters.
4a, 4b = Agent sends monitoring results to masters.
5 = Satellite monitors device.
6 = Agent sends monitoring results to satellite.

The top-level subsystem **A** comprises a primary master and a secondary master. Each of them directly monitors a device. The subsystems **C** and **D** comprise two agents.

The agent in **D** is indirectly connected to the masters by the satellite in subsystem **B**. This satellite forwards monitoring results from the agent and directly monitors another device. The other agent in subsystem **C** is directly connected to both masters.

To set up a multi-level HA monitoring system

1. On all masters and satellites, install the R&S CHM host software.
   How to: Section 4, "Installing R&S CHM", on page 18
2. On each other host that masters or satellites cannot monitor with external checks, install the agent software.
   How to: Section 4.2, "Installing R&S CHM agents", on page 20
3. Install certificates and keys.
   How to: Section 5.5, "Deploying certificates for distributed monitoring", on page 35
4. Edit the `chm.yaml` file to describe the multi-level HA monitoring architecture.
   How to: Example "YAML configuration: multi-level HA monitoring" on page 120.
5. All masters and satellites require an identical `chm.yaml` file. Save this file here:
   `/etc/opt/rohde-schwarz/chm/`
6. Restart the chm service on all masters and satellites to take the changes effect:
   `systemctl restart chm`
   We recommend starting the service in sequence on the masters and then on the satellite.
7. On each R&S CHM agent, run the node setup scripts with options that describe the multi-level HA system. See Section 6.12.4.2, "Configuring agents for multi-level HA monitoring", on page 121.

6.12.4.1 Editing the YAML configuration for multi-level HA monitoring

Multi-level HA monitoring configurations require that you configure subsystems for multi-level support and two masters for high-availability support.

► Specify the entries in the `chm.yaml` file for the multi-level HA monitoring configuration:

- The names of all subsystems
- The members of the subsystems, i.e. masters, satellites or agents, or monitored hosts or devices
- Exactly one parent subsystem except for the top-level subsystem
- Two R&S CHM host instances that serve as HA masters

Example: YAML configuration: multi-level HA monitoring

A satellite always requires a R&S CHM host installation on Linux. Thus, the satellite host requires the `connections: ["icinga2_linux"]` key.

This example:

- Uses the host names from Figure 6-18
- Omits any checks for clarity

```
subsystems:
  - name: "A"
    hosts:
      - "chm-master-a"
      - "chm-master-b"
      - "some-device"
      - "some-other-device"

  - name: "B"
    hosts:
      - "chm-satellite-b"
      - "rs-device"
    parent_subsystem: "A"

  - name: "C"
    hosts:
      - "chm-win-agent"
    parent_subsystem: "A"

  - name: "D"
    hosts:
      - "chm-linux-agent"
    parent_subsystem: "B"


hosts:
  # primary master in A
  - name: "chm-master-a"
    tags: ["chm"]

  # secondary master in A
  - name: "chm-master-b"
    tags: ["icinga2_ha"]

  # satellite in B
  - name: "chm-satellite-b"
    connections: ["icinga2_linux"]

  # windows agent in C
  - name: "chm-win-agent"
    connections: ["icinga2_win"]
```

```
# linux agent in D
- name: "chm-linux-agent"
  connections: ["icinga2_linux"]

# devices
- name: "some-device"
  checked_by: "chm-master-a"

- name: "some-other-device"
  checked_by: "chm-master-b"

- name: "rs-device"
```

The `checked_by` key for the host `some-other-device` ensures that this host is monitored by a specific R&S CHM instance, here the secondary master.

6.12.4.2 Configuring agents for multi-level HA monitoring

It is necessary that you inform the agents about these relations:

- The own subsystem.
- The parent subsystem.
- The connection to masters or satellites.
- The existence of both masters.

To configure the agents

► Run these scripts to complete agent configuration:

- On Linux agents, run the `chm_node_setup` shell script.
- On Windows agents, run the `chm-node-setup.bat` batch script.

For parameterization, see the following examples that use the FQDNs from Figure 6-18.

Example:

Script on the Linux agent **chm-linux-agent-d** (subsystem **D**):

```
chm_node_setup \
--subsys D \
--parent-subsys B \
--parent-chm chm-satellite-b
```

Example:

Script on the Windows agent **chm-win-agent-f** (subsystem **C**):

```
"C:\Program Files\chm\chm-node-setup.bat" \
--subsys C \
--parent-subsys A \
--parent-chm chm-master-a
--second-parent-chm chm-master-b
```

6.13 Configuring automatic system control

If you configure automatic system control, R&S CHM can automatically execute a predefined set of commands triggered by check result events from the logic network.

For example, R&S CHM can shut down host systems and switch off power outlets when temperature checks are critical.

Automatic system control allows for the following:

- Control hosts via SSH and SNMP set commands.
- Execute multiple sets of commands that can be individually triggered by configurable logic events.
  Such multiple sets of commands are named as `playbooks` in the `chm.yaml` configuration file.
- Execute commands in sequence in `taskgroups` or in parallel in a `task`.
- Delay the execution of `taskgroups` for a defined time to allow a graceful shutdown of hosts.

To configure automatic system control

In the `hosts` sections for the R&S CHM server and the host nodes, configure `automatic_system_control`.

1. In the R&S CHM server configuration, configure all playbooks and the trigger events from the logic network.
2. **For each controlled host node:**
   - Configure the individual SSH and SNMP connections.
   - Define the functions (commands with parameters) that are referred to in the playbooks for the specific host.

To filter for automatic system control information, see `logging` on page 57.

---

automatic_system_control (CHM server)

Configures automatic system control on the R&S CHM server.

Parameters:

playbooks	string Name of the playbook.
trigger	string Output from the logic network.
taskgroups	1 | 2 | <n> Individual `taskgroups` that you configure as pairs of `<target host>: <function label>` with an optional delay (`wait: <time in seconds>`).

Example: Automatic system control on R&S CHM server.

```
hosts:
  - name chm-server
    logic:
      reboot_all:
        function: worst
        ins: [ manual_reboot_all ]
    checks:
      - ping:
      # add manual check to trigger a playbook
      - manual:
          displayname: "manual_reboot_all"
          logic_id: "manual_reboot_all"
    automatic_system_control:
      playbooks:
        my_playbook_for_reboot:
          trigger: "reboot_all"
          taskgroups:
            - 1:
              - host_win_1: "reboot"
              - wait: 30
            - 2:
              - host_linux_1: "reboot"
              - wait: 30
            - 3:
              - host_raritan_pdu: "power_off_outlet_1"
              - host_raritan_pdu: "power_off_outlet_2"
```

---

automatic_system_control (CHM node)

Configures automatic system control on controlled hosts.

Related parameters

- snmp_connection on page 128 (SNMP v2/v3).

Parameters:

ssh_connection	If used, the SSH connection. See ssh_connection on page 131.
snmp_connection	If used, the SNMP v2/v3 connection. See snmp_connection on page 128.
functions	See functions on page 124.

Example:

```
hosts:
  - name: <Name of the chm node>
    automatic_system_control:
      functions:
       reboot_windows_via_ssh:
         cmdtype: "ssh"
         command: "shutdown"
         params: ["/r", "/t 5"]
       reboot_linux_via_ssh:
         cmdtype: “ssh_reboot_linux”
       power_off_raritan_pdu_outlet_1:
         cmdtype: "snmpset"
         oid: ".1.3.6.1.4.1.13742.6.4.1.2.1.1"
         value: 0
         value_type: "int"
       power_off_raritan_pdu_outlet_2:
         cmdtype: "switch_raritan_pdu"
         pdu_id: 1
         outlet: 2
         state: "off" # off|on|cycle
```

functions (Functions on CHM node)

Configures the `functions` on controlled hosts in the context of automatic system control. A function is a definition of the command that is executed for a specific host. See `automatic_system_control` on page 123.

Parameters:

<function label> string

Label of the specific function, without spaces in the name, e.g. `reboot_windows_via_ssh`.

cmdtype ssh | snmpset | ssh_shutdown_windows | ssh_reboot_windows | ssh_shutdown_linux | ssh_reboot_linux | switch_raritan_pdu

Type of function that is executed. The type can be one of the listed commands with corresponding parameters.

ssh
Executes an arbitrary SSH command. Specify the parameters as a list, e.g. for a shutdown, add `["/r", "/t 5"]`. See also Table 6-13.

snmpset
Executes an arbitrary SNMP command. See also Table 6-14.

ssh_shutdown_windows
Maps to `shutdown /s` SSH command.

ssh_reboot_windows
Maps to `shutdown /r` SSH command.

ssh_shutdown_linux
Maps to `sudo shutdown` command.

ssh_reboot_linux
Maps to `sudo reboot` command.

switch_raritan_pdu
Maps to Raritan PDU `OID .1.3.6.1.4.1.13742.6.4.1.2.1.2.<pdu_id>.<outlet>` command.

Example: For a comprehensive example, see `automatic_system_control` on page 123.

Table 6-13: Options for cmdtype "ssh"

Option	Description
`command`	Linux or Windows command executed via SSH
`params`	Optional list of parameters for the command. E.g., `["/A", "/B C"]`

Table 6-14: Options for cmdtype "snmpset"

Option	Description
`oid`	Numeric SNMP OID
`value`	Value to set
`value_type`	`"str"` for a string value or `"int"` for an integer value

6.14 Using common keys

You can use the following common keys with any status check that is listed in Section 7, "Configuring status checks", on page 134.

checkgroups 125
displayname 126
health_host 126
interval 126
logic_id 126
maps 126

checkgroups (Checkgroups)

Assigns a check to one or more specific groups that you can configure and display on the web GUI.

Example: `checkgroups: [Cluster, Buster]`

Example: If the check group contains a colon (:), enclose the whole check group string in quotation marks.
`checkgroups: ["Resources :- Disk space"]`

displayname (Display name)

Display a user-friendly name on the web GUI.

There is no explicit limit of the string length. R&S CHM supports at least 1024 characters.

R&S CHM supports at least the following special characters:

`!@#$%^&*()_+=-`~[]\\{}|;':\",./<>?äöüÄÖÜß €„"»«§°²³µ`

Example:

```
displayname: My special service name
```

health_host (Check redirection)

FQDN of the host that provides status information for the system component, e.g. a NAVICS or R&S RAMON device.

Use this key if you cannot obtain the status information from the system component itself but from a configured, central monitoring host.

Example:

```
health_host: navics_server.local
```

For an example in combination with the `navics` status check, see `navics` on page 164 and `gb2pp` on page 152.

interval (Configure execution interval)

Configure an individual execution interval for a status check (in s, default: 60 s).

Example:

```
- idrac:
    snmp_connection:
      community: public
    interval: 30
```

logic_id (Logic identifier)

Assign a unique identifier to a check. You can specify this identifier in `logic` on page 52.

Ensure that all `logic_id` values are unique in the `chm.yaml` file.

Example:

```
checks
  - icinga2_cluster:
      logic_id: component1
  - dhcp
      logic_id: component2
  - dns
      logic_id: component3
```

maps (Coordinates for status icons on maps)

Specifies the coordinates for status icons on the maps.

Related parameters

Graphical system view (maps) on page 86

Parameters:

<map_name>	Name of the map as specified in Graphical system view (maps) on page 86.
x	The x-value on the image (horizontal, left to right).
y	The y-value on the image (vertical, up and down).
label_<format>	Item-specific label background, border or style. For more information about these keys, see Graphical system view (maps) on page 86.

Example: In these host and service configurations, the names of the maps are Overview, Rack and Redundancy. Compare with the example in Graphical system view (maps) on page 86.

```
hosts:
  - name: chm2-staging-disa.rsint.net
    displayname: "CHM Master"
    connections: [icinga2_api]
    tags: [chm]
    maps:
      Overview:
        x: 235
        y: 270
      Rack:
        x: 60
        y: 170
      Redundancy:
        x: 80
        y: 215
        label_background: "#AAAAAA"
# [...] some other keys
    checks:
      - icinga2_cluster:
          displayname: Icinga2 connect. via JSON/RPC on 5665/tcp
      - dhcp:
          maps:
            Overview:
               x: 250
               y: 208
            Rack:
              x: 60
              y: 300
              label_border: "#1E90FF"
            Redundancy:
              x: 620
              y: 100
```

6.15 Using frequent keys

You can use the following keys in multiple status checks. For example, you need SNMP in all checks that are based on this protocol, e.g. `nport` on page 167.

snmp_connection 128
ssh_connection 131
thresholds 132

snmp_connection (SNMP connection)

Specifies the properties of the SNMP connection for communication between R&S CHM and the device. You can mix the SNMP versions in the configuration.

- SNMPv1/v2: unencrypted communication
- SNMPv3: encrypted communication

An individual `snmp_connection` check overrules the `snmp_connection` host configuration.

Parameters:

port — numeric

Communication port at the device, the SNMP agent (optional).

*RST: 161

version — 1 | 2 | 3

SNMP protocol version.

*RST: 2

retries — numeric

Number of retries to be used in the requests (optional).

*RST: 5

timeout — numeric

Timeout between retries (optional). Floating point numbers can be used to specify fractions of seconds, e.g. `1.25`.

*RST: 1
Default unit: s

snmp_engine_id — number

Sets a specific engine identifier that must be unique in the network. Otherwise, it is created automatically (optional). See also RFC 3411, section 5.

community — string

SNMP community string for SNMPv1/v2 transactions. The community string is a type of shared password between the SNMP management station and the device. The community string is used to authenticate the SNMP management station.

*RST: public

v3_users

One or more users that can retrieve data from the interface. The users are used for SNMPv3 authentication and replace the insecure SNMPv2 community string. Each SNMPv3 user uses the keys that are defined by the SNMPv3 user-based security model (USM): `secname`, `authproto`, `authpass`, `privproto`, `privpass`.
R&S CHM only supports the authPriv security model. It enforces the highest security level in SNMPv3 with authentication and encryption.

trapreceivers

Configures R&S CHM to inform a list of SNMP notification receivers about system status changes via SNMPv2/v3 traps (optional).
For SNMPv3, R&S CHM only supports the authPriv security model. You specify the keys that are defined by the SNMPv3 USM: `secname`, `authproto`, `authpass`, `privproto`, `privpass`.

host
The host name of the SNMP notification receivers.

inform
Sends inform messages instead of traps if set to `true` (optional).

port
The port of the host.

version
The used SNMP version (optional). Specify `version: 3` for SNMPv3 configurations. Otherwise, R&S CHM assumes an SNMPv2 connection.

community
The SNMP community that is expected by the SNMP notification receiver to accept the trap.

How to: Section 6.10.2, "Configuring SNMPv2 traps", on page 89

secname

string

Identifier (security name) used for authenticated SNMPv3 messages.
See also: Section 6.7, "Managing password identifiers", on page 76

authproto

MD5 | SHA | SHA-224 | SHA-256 | SHA-384 | SHA-512 | None

The authentication protocol that is used for authenticated SNMPv3 messages. If your operating system is hardened with FIPS mode, you cannot use `MD5`.

*RST: MD5

authpass

string

Password used for authenticated SNMPv3 messages (optional). The password requires a minimum length of 8 characters. If not specified, R&S CHM looks up the password in the password store using the `secname` value as the identifier.
Option 1: Clear text password as used in the example at the end of this key description.
Option 2: VAULT:<path_to_vault> as used in the example at the end of this key description (recommended).
Option 3: If not specified, R&S CHM looks up the password in the password store using the `secname` value as the identifier as used in the example at the end of this key description.

privproto

DES | 3DES | AES-128 | AES-192 | AES-256 | None

Privacy protocol used for encrypted SNMPv3 messages.

*RST: DES

privpass

string

Password used for encrypted SNMPv3 messages (optional). The password requires a minimum length of 8 characters.
Option 1: Clear text password as used in the example at the end of this key description.
Option 2: VAULT:<path_to_vault> as used in the example at the end of this key description (recommended).
Option 3: If not specified, R&S CHM looks up the password in the password store using the `secname` value as the identifier as used in the example at the end of this key description.

context

string

Context name used for SNMPv3 messages, e.g. `spectracom_time`

*RST: empty string ""

Example:

SNMPv1/2

```
snmp_connection:
  port: 161
  version: 2
  community: public
```

Example:

SNMPv3, option 1: Use a cleartext password for a Spectracom SecureSync time server and write the passwords in clear text to the `chm.yaml` configuration file:

```
- spectracom_time:
    checkgroups: [ ]
    snmp_connection:
      context: spectracom_time
      authpass: privatusprivatusprivatusprivatus # clear text password
      privpass: privatusprivatusprivatusprivatus # clear text password
    no_minor_alarm: true
    no_major_alarm: true
```

Example: **SNMPv3, option 2**: Use the password store with different passwords for `authpass` and `privpass`:

```
- nport
    checkgroups: [water, earth, fire, air]
    snmp_connection:
      version: 3
      context: nport
      secname: mydeviceaccount  # the snmp user
      authpass: VAULT:snmp_passwords/nport/device1/authpass
      # The path to the password in the password store
      privproto: AES-256
      privpass: VAULT:snmp_passwords/nport/device1/privpass
      authproto: SHA
...
```

Example: **SNMPv3, option 3** (deprecated): Use the password store with identical passwords:

```
- nport:
    checkgroups: [water, earth, fire, air]
    snmp_connection:
      version: 3
      context: nport
      secname: mydeviceaccount
         # lookup of passwords in password store
      authproto: SHA
      privproto: AES-256
```

---

ssh_connection (SSH connection on CHM node)

Configures SSH connection settings for, e.g. automatic system control. See `automatic_system_control` on page 123.

Parameters:

username	string Name of the SSH user.
password	string Password of the user. R&S CHM stores the SSH password in clear text. To use a password vault, see `snmp_connection` on page 128. **Option 1**: Clear text password. **Option 2**: Password vault. See also: `snmp_connection` on page 128
priv_key_path	string Key based authentication, specifying the key file path.
priv_key_data	string Key based authentication, providing the key data directly or referencing a password vault.

port numeric

Port of the remote SSH server (optional).

*RST: 22

Example: **Password authentication (option 1)**:

```
ssh_connection:
  username: <ssh user>
  password: <password>
```

Example: **Password authentication (option 2)**:

```
ssh_connection:
  username: <ssh user>
  password: VAULT:<ssh user>
```

Example: **Key based authentication (file path)**:

```
ssh_connection:
  username: <ssh user>
  priv_key_path: /etc/opt/rohde-schwarz/chm/chm_staging_id_rsa
```

Example: **Key based authentication (vault)**:

```
ssh_connection:
  username: <ssh user>
  priv_key_data: VAULT:<ssh user>
```

---

thresholds (Thresholds)

Specify thresholds for alert levels. Use `thresholds` together with suitable checks as mentioned in the description of the checks.

Thresholds are implemented according to the Monitoring Plugins Development Guidelines. The Table 6-15 is adopted from this guide.

Parameters:

warning Threshold for the `warning` alert level.

critical Threshold for the `critical` alert level.

Example:

```
thresholds:
  warning: ':0'   # E.g. alert if 1 or more exceed. occurred
  critical: ':0'  # E.g. alert if 1 or more exceed. occurred
thresholds:
  warning: '20:'  # E.g. alert if check cond. falls below 20
  critical: '10:' # E.g. alert if check cond. falls below 10
```

Generalized format of ranges:

`[@]start:end`

Table 6-15: Example ranges

Range definition	Generate an alert if x...
10	< 0 or > 10 (outside the range of {0 to10})
10:	< 10 (outside {10 to ∞})

Range definition	Generate an alert if x...
~:10	> 10 (outside the range of {-∞ to 10})
10:20	< 10 or > 20 (outside the range of {10 to 20})
@10:20	≥ 10 and ≤ 20 (inside the range of {10 to 20})

7 Configuring status checks

R&S CHM provides a specific set of status checks that you can configure. Here, you can obtain an overview of available status checks and necessary information on how to configure them.

For common keys that are supported by all status checks, see Section 6.14, "Using common keys", on page 125.

This user documentation uses the identifier ***RST** to indicate the default values in the descriptions of the YAML keys.

ar60 135
argus 135
bitdefender 136
chm_agent_conn 136
check_kerberos_auth 136
chm_remote, simcos3 137
chm_remote_grpc 137
cisco_hardware 142
cputemp 143
dhcp 143
dkn 144
dns 146
domain 147
dummy 148
eta_pdu 148
file_content 149
file_exists 150
fortinet 150
fortinet_wcs 151
gb2pp 152
generic_printer 154
gude 155
hums 156
icinga2_cluster 156
idrac 156
ilo 158
lancom_vpn_status 159
lancom_xs_gs_3000 160
load 160
mac_address 162
manual 162
meinberg 163
mikrotik 163
msr4 164
navics 164
nport 167
ntp_time 168

nw_interface....169
os_disk....171
os_memory....171
os_process....172
os_service....172
passive....172
ping....173
raritan_pdu....174
RCCM....177
snmp....178
snmp_diskspace....179
snmp_flex....179
snmp_hostalive....180
snmp_processes....181
snmp_time....181
spectracom_time....182
ssh....183
synology....184
system_state....184
tcp....185
tmr_radio....185
trustedfilter....185
ups....186
vmware....187
windowsupdateage....188
xcp_ng....189

ar60 (Check AR60 microwave modem)

Checks the global status of the AR60 satellite demodulator from Work Microwave.

Related parameters

- `snmp_connection`

Example:

```
- ar60:
    snmp_version: 2
    snmp_community: public
```

argus (Connected devices and users in R&S ARGUS)

Checks status information of connected RF equipment and the operating users in R&S ARGUS via SNMP.

Related parameters

- `snmp_connection`

Parameters:

users — string

User states (optional).

devices — string

Device states (optional).

Example:

```
- argus:
  snmp_version: 2
  snmp_community: public
```

---

bitdefender (Bitdefender virus definitions age; deprecated)

Monitors the age of the virus definitions of Bitdefender antivirus software.

Related parameters

- `thresholds`

Parameters:

thresholds — warning | critical

Alert levels for the age of the definition base (in days).
For more information about the `thresholds` syntax, see `thresholds` on page 132.

Example:

```
checks:
- bitdefender:
  thresholds:
    warning: '10'
    critical: '30'
```

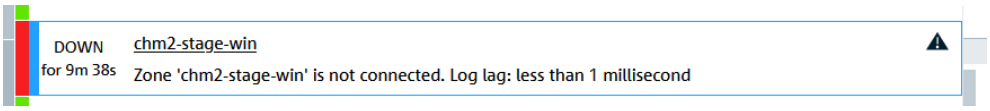
---

chm_agent_conn (CHM agent connection)

Checks the connection between the R&S CHM host and the R&S CHM service that runs on an agent. This check enhances the reliability of the returned status.

Return status values for checked agents:

- "UP" if the service is running and connection is possible.
- "DOWN" if the service is not running or connection is not possible.



You can configure this check for agents instead of `ping`. You can use the legacy name `chm_agent_connection` or the new name `chm_agent_conn`.

Example:

```
- chm_agent_connection:
```

Example:

```
- chm_agent_conn:
```

---

check_kerberos_auth (Check Kerberos authentication)

Monitor that Linux agents can authenticate against Active Directory.

Example:

```
- check_kerberos_auth:
```

chm_remote, simcos3 (RS-RAMON-CHM-REMOTE connection)

Monitors any device that implements RS-RAMON-CHM-REMOTE MIB, e.g. R&S RAMON and R&S SIMCOS.

Related parameters

- `health_host`
- `snmp_connection`

Parameters:

appid — string

The identifier of the software, see Table 7-2.

checkid — string

The identifier of the device, see Table 7-2.
With R&S SIMCOS, set the `checkid` that you have specified during device configuration.
With R&S SIMCOS, set the `checkid` that you have specified during device configuration.

Example: Alternative 1

```
- chm_remote:
    snmp_connection:
      port: 1234
      version: 2
      community: public
    appid: SIMCOSIII
    checkid: MODEM 1
```

Example: Alternative 2

```
- simcos3:
    snmp_connection:
      port: 1234
      version: 2
      community: public
    checkid: MODEM 1
```

chm_remote_grpc (gRPC-based RAMON monitoring)

Monitors health summary and status of R&S RAMON. For concepts a configuration instruction, see Section 6.8, "Configuring R&S RAMON for monitoring", on page 78.

Parameters:

appid — string

The identifier of the software, see Table 7-2.

checkid — string

The identifier of the device, see Table 7-2.
With R&S SIMCOS, set the `checkid` that you have specified during device configuration.

system_control
reboot | selftest | shutdown
Management functions for R&S RAMON components that you can show on the R&S CHM GUI. The ability of a device or driver to respond to these commands depends on the specific implementation of the device driver. Not all devices support all functions. Only configured functions are shown on the web GUI.

port
numeric
Remote TCP port.
*RST: 18005

server_root_cert
string
Path of the file that contains the PEM encoded root certificate of the target host. The certificate is used for authenticating the target host.
*RST: /var/lib/icinga2/certs/ca.crt

client_root_cert
string
Path of the file that contains the PEM encoded root certificate of the local host. The certificate is used by the server in combination with `client_cert` for authenticating the local host.
*RST: /var/lib/icinga2/certs/ca.crt

client_cert
string
Path of the file that contains the PEM encoded certificate of the local host. The certificate is used by the server in combination with `client_root_cert` for authenticating the local host.
*RST: /var/lib/icinga2/certs/<localhost_fqdn>.crt

client_privkey
string
Path of the file that contains the PEM encoded private key that corresponds to `client_cert` of the local host.
*RST: /var/lib/icinga2/certs/<localhost_fqdn>.crt

insecure
boolean
If set to true, try connecting without encryption and client/server authentication.
*RST: false

Example: Configuration of the paths to the certificates.

```
hosts:
  - name: applicationserver.some.net
    checks:
      - chm_remote_grpc:
          appid: RaCas
          checkid: 1
          server_root_cert: /var/certs/srv_ca.crt
          client_root_cert: /var/certs/cl_ca.crt
          client_cert: /var/certs/cl.crt
          client_privkey: /var/keys/cl.key
```

Example: "System Control":
With the following configuration, monitoring for R&S RAMON RACAS is activated and the "System Control" function "selftest" for R&S RAMON RACAS is displayed in the tab of the CHM GUI

```
hosts:
  - name: applicationserver.some.net
    checks:
      - chm_remote_grpc:
          appid: RaCas
          checkid: 1
          system_control:
            functions:
              selftest:
```

Example: The following configuration adds the `reboot` and `selftest` management functions to the web GUI > "System Control" view.

```
- chm_remote_grpc:
    appid: ESMEDRV1
    checkid: RxChmSnmpCheck1
    system_control:
      functions:
        reboot:
        selftest:
```

Table 7-1: Supported software and identifiers, only for - chm_remote_grpc

Software	appid	checkid
R&S EWCoM	EWCoMApplication	EWCoMHealthCheck1

Table 7-2: Supported software and identifiers for - chm_remote_grpc, - chm_remote, - simcos3

Software	appid (<x> is the number of the device)	checkid

R&S SIMCOS	SIMCOSIII
	Note: Only supported in these status checks:
	`- chm_remote`
	`- simcos3`
		<checkid>

R&S RAMON CA120	CA120Server	StorageUnits
R&S RAMON CA120	CA120Server	ProcessingUnits
R&S RAMON CA120	CA120Server	Tuners
R&S RAMON CA120	CA120Server	Server
R&S RAMON Antennamatrix	AntennaMatrixDRV1 to AntennaMatrixDRV<x>	ChmSnmpCheck1
R&S RAMON Amrec	AMRECServer1 to AMRECServer<x>	AMRECDevices
R&S RAMON Driver DDF007	DDF007DRV1 to DDF007DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDF1555	DDF1555DRV1 to DDF1555DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDF200M	DDF200MDRV1 to DDF200MDRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDF205	DDF205DRV1 to DDF205DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDF255	DDF255DRV1 to DDF255DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDF260	DDF260DRV1 to DDF260DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver DDFCTL	DDFCTLDRV1 to DDFCTLDRV<x>	RxChmSnmpCheck1
R&S RAMON Driver WPU500	WPUCTLDRV1 to WPUCTLDRV<x>	RxChmSnmpCheck1
R&S RAMON Driver EM100	EM100DRV1 to EM100DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver ESMD	ESMDDRV1 to ESMDDRV<x>	RxChmSnmpCheck1
R&S RAMON Driver ESME	ESMEDRV1 to ESMEDRV<x>	RxChmSnmpCheck1
R&S RAMON Driver ESMW	ESMWDRV[1] to ESMWDRV[x]	RxChmSnmpCheck1
R&S RAMON Driver EB200	EB200DRV1 to EB200DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver EB500	EB500DRV1 to EB500DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver EB510	EB510DRV1 to EB510DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver PR100	PR100DRV1 to PR100DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver PR200	PR200DRV1 to PR200DRV<x>	RxChmSnmpCheck1
R&S RAMON Driver EM200	EM200DRV1 to EM200DRV<x>	RxChmSnmpCheck1

Software	appid (<x> is the number of the device)	checkid
R&S RAMON RACAS	RaCas	1
R&S RAMON SIGDB	SIGDB	1
R&S BBI	BBI	GenChk
R&S BBI	BBI	MemChk
R&S BBI	BBI	ConChk
R&S BBI	BBI	SigChk
R&S BBI	BBI	KeyCalcChk
R&S BBO	BBO	GenChk
R&S BBO	BBO	MemChk
R&S BBO	BBO	ConChk
R&S BBO	BBO	DevoChk
R&S DCU	DCU	GenChk
R&S DCU	DCU	MemChk
R&S DCU	DCU	IfChk
R&S DCU	DCU	KeyCalcChk
R&S DCU	DCU	FPGAChk
R&S DCU	DCU	ProdChk
R&S GSA6Sensor	GSA6Sensor	GenChk
R&S GSA6Sensor	GSA6Sensor	MemChk
R&S GSA6Sensor	GSA6Sensor	HealthChk
R&S GSA6Sensor	GSA6Sensor	SigChk
R&S GSA6Sensor	GSA6Sensor	DbChk
R&S GSA6Sensor	GSA6Sensor	ProdChk
R&S Linkmanager	LnkMngr	GenChk
R&S Linkmanager	LnkMngr	MemChk
R&S Linkmanager	LnkMngr	HealthChk
R&S Linkmanager	LnkMngr	ConChk
R&S Linkmanager	LnkMngr	NtwrkChk
R&S Receiverserver	RcvSrv	GenChk
R&S Receiverserver	RcvSrv	MemChk
R&S Receiverserver	RcvSrv	HealthChk
R&S Receiverserver	RcvSrv	SigChk

Software	appid (<x> is the number of the device)	checkid
R&S Receiverserver	RcvSrv	ConChk
R&S Receiverserver	RcvSrv	SynchChk
R&S Receiverserver	RcvSrv	ProdChk
R&S SBU	SBU-T	GenChk
R&S SBU	SBU-T	MemChk
R&S SBU	SBU-T	SigChk
R&S SBU	SBU-T	ConChk
R&S SBU	SBU-T	SynchChk
R&S SBU	SBU-T	ProdChk
R&S SCG	SCG	GenChk
R&S SCG	SCG	MemChk
R&S SCG	SCG	HealthChk
R&S SCG	SCG	ConChk
R&S SCG	SCG	QualChk
R&S SCM	SCM	GenChk
R&S SCM	SCM	MemChk
R&S SCM	SCM	DbChk
R&S SCM	SCM	ConChk
R&S Sensorserver	SNS	GenChk
R&S Sensorserver	SNS	MemChk
R&S Sensorserver	SNS	ConChk
R&S Sensorserver	SNS	ShrdFldChk
R&S Sensorserver	SNS	ProdChk

cisco_hardware (Cisco hardware)

cisco_hardware.py

Monitors the hardware status of a Cisco switch via SNMP. The check monitors fans, temperature, power supplies and modules.

Supported devices

All devices that support the following MIBs, including Cisco Catalyst 9300:

- CISCO-ENVMON-MIB
- CISCO-STACKWISE-MIB
- CISCO-ENTITY-FRU-CONTROL-MIB

Related parameters

- snmp_connection

Parameters:

device_name — string

Name of the device. This name is shown in the status summary (optional).

return_status — CRITICAL | WARNING

Return status for failures (optional).

fans — numeric

Number of built-in fans (optional).

*RST: 2

powersupplies — numeric

Number of built-in power supplies (optional).

*RST: 2

Example:

```
- cisco_hardware:
    device_name: CISCO 9300 Center Switch
    fans: 3
    returnstatus: WARNING
```

---

cputemp (Monitor average CPU temperature)

Monitors the CPU package temperature for all CPUs on a Windows host. The CPU package temperature is a 256 millisecond average value of the hottest temperature sensor.

Related parameters

- thresholds

Parameters:

thresholds — warning | critical

Check-specific alert levels. For more information about the threshold syntax, see thresholds on page 132 (optional).

*RST: warning: 80, critical: 90
Default unit: °C

Example:

```
- cputemp:
    thresholds:
      warning: '70'
      critical: '90'
```

---

dhcp (DHCP server)

Tests the availability of DHCP servers on a network. By default, the check broadcasts a DHCPDISCOVER packet to port 67/UDP and checks whether a DHCPOFFER is received on 68/UDP within a given timeout.

Related parameters

- `thresholds`

Parameters:

servers — IP_address1 , IP_address2 , IP_address<n>

List of IP address of DHCP servers from which an answer is expected (optional). If multiple servers are specified, and some but not all respond, this situation results in a warning alert.

*RST: Any responding DHCP server is ok

offeredip — IP_address

Expected IP address in DHCPOFFER (optional). If specified, and a DHCPOFFER with unexpected IP is received, this situation results in a warning alert.

*RST: Any offered IP address is ok

timeout — time

Time to wait for DHCPOFFER (optional).

*RST: 2
Default unit: s

interface — string

Interface to be used for listening (optional).

*RST: eth0

mac — string

MAC address to use in the DHCP request (optional).

*RST: MAC address of the configured interface

unicast — true | false

If set to `true`, mimics a DHCP relay (optional). Requires to set also at least one server.

*RST: false

Example:

```
-dhcp
  servers: [192.168.178.0 , 192.168.178.1]
  unicast: true
```

---

dkn (Devices and nodes in a DKN)

R&S CHM lets you monitor the status of devices and nodes (BACs) by using the GEDIS KMS RLM SNMP MIB in a NEMAS DKN from the Thales Group.

Related parameters

- `snmp_connection`

For returned status values, see Table 7-3.

Parameters:

type — device_ready | device_status | node_link | node_status

Check type.

device_ready
Monitor if a DKN device is in ready state.

device_status
Monitor the status of a DKN device.

node_link
Monitor the link status of the node.

node_status
Monitor the node status.

id

numeric

Identifier of the DKN device or node.

Example:

```
- dkn:
  snmp_connection:
    version: 2
    community: public
  type: device_ready
  id: 1
- dkn:
  snmp_connection:
    version: 2
    community: public
  type: device_status
  id: 1
- dkn:
  snmp_connection:
    version: 2
    community: public
  type: node_link
  id: 2
- dkn:
  snmp_connection:
    version: 2
    community: public
  type: node_status
  id: 2
```

Table 7-3: Status mapping - DKN to web GUI

Check type	Status on web GUI	DKN status
node_link	"OK"	Connected
	"CRITICAL"	Disconnected
device_ready	"OK"	Ready
	"CRITICAL"	Not ready
device_status, node_status	"OK"	OK, Info

Check type	Status on web GUI	DKN status
	"WARNING"	Warning
	"CRITICAL"	Error, Fatal

dns (Domain name server)

Tests the availability of domain name servers on a network. The default servers from `/etc/resolv.conf` are used unless explicitly specified.

Related parameters

- `thresholds`

Parameters:

lookup — string

The host name or IP to query the DNS for (optional).

*RST: Name of host where the check is executed

server — IP_address

The DNS server to query.

*RST: The server configured in the OS.

query_type — A | AAAA | SRV | TXT | MX | ANY

The DNS record type (optional).

A
IPv4 address record.

AAAA
IPv6 address record.

SRV
Service location record.

TXT
Text record.

MX
Mail exchange record.

ANY
A special query (meta-query, deprecated).

*RST: A

answers — string

The answers to look for. A host name must end with a dot. Define multiple answers as array (optional).

*RST: Do not check for specific addresses in the answer

authoritative
true | false

Expect the server to send an authoritative answer. Non-authoritative answers are marked with "non-authoritative answer:" and mean that a name server looked up the entry from it is local cache (optional). If set to false, there is no check whether authoritative or not.

*RST: false

accept_cname
Accept CNAME (canonical name, aka alias) responses as a valid result to a query (optional).

timeout
numeric

Seconds before connection times out, i.e. forced interruption by SIGALRM, then SIGKILL (optional).

*RST: 10
Default unit: s

thresholds
Alert levels for used datastore space (optional).
For more information about the `thresholds` syntax, see `thresholds` on page 132.

Example:

```
- dns
  lookup: my_dnsserver
  accept_cname:
  timeout: 20
```

---

domain (Monitor a domain)

Monitors a domain using the given check type.

Parameters:

type
sec_channel | replication | membership

The type of domain-check to create.

sec_channel
Tests the secure channel between the local computer and its domain.

replication
Tests the replication between the domain controllers.

membership
Checks the Windows domain membership of a Windows host.

domainName
string

The domain to monitor (only optional for `replication`).

Example:

```
domain:
  type: sec_channel
  domainName: rsint.net
```

Example:

```
domain:
  type: replication
```

Example:

```
domain:
  type: membership
  domainName: rsint.net
```

---

dummy (Dummy)

The check always shows the status "UP" for the host. Use this check if you cannot use another host check, e.g. if ICMP is blocked in the network.

Example:

```
- name: host_prepare.net
  checks:
    - dummy:
```

---

eta_pdu (Monitor E-T-A PDUs)

Monitors PDUs that support the MIB `eta_RCI11_1.0.1_MIB.mib` from E-T-A® Engineering Technology.

Monitored aspects:

- Serial number of the fuse (info only)
- Status of fuse
- No system parameter available fault
- Parameter CRC fault
- Program memory CRC fault
- Internal memory fault
- Controller fault
- Watchdog reset fault
- Output status of the fuse
- Short is detected by this fuse
- Overload status
- Undervoltage status
- Overvoltage status
- Overtemperature status

Related parameters

- `thresholds`
- `snmp_connection`

Parameters:

fuse_number	numeric The fuse number to check (optional).
fuse_feed	A | B The fuse feed to check (`A` or `B`; optional).

temperature_sensor_id numeric

The ID of the temperature sensor to check (optional).

humidity_sensor_id numeric

The ID of the humidity sensor to check (optional).

Example: Check fuse no. `3` and fuse feed `B`.

```
- eta_pdu:
    fuse_number: 3
    fuse_feed: B
```

Example: Check fuse no. `1`, fuse feed `A`, temperature sensor ID and humidity sensor ID.

```
- eta_pdu:
    fuse_number: 1
    fuse_feed: "A"
- eta_pdu:
    temperature_sensor_id: 1
    thresholds:
      temperature:
        warning: ":30"
        critical: ":40"
- eta_pdu:
    humidity_sensor_id: 1
    thresholds:
      humidity:
        warning: ":80"
        critical: ":90"
```

---

file_content (Monitor file content)

Monitors the content of a file for a predefined string on Linux and Windows agents.

Parameters:

file string

Name of the monitored file (optional).

*RST: /tmp/import_service_result

string string

Search string. Mandatory on Linux agents and not applicable on Windows agents.

pattern string

Search string, expressed as a regular expression in .Net syntax. Mandatory on Windows agents and not applicable on Linux agents.

match_is_ok true | false

If `true`, a match is interpreted as ok (default). If `false`, a match is interpreted as failure.

returnstatus
WARNING | CRITICAL
Return value if the check fails, i.e. "WARNING" or "CRITICAL" (optional).

oksummary
string
Text that is shown if the string is found in the file.

badsummary
string
Text that is shown if the string is not found in the file.

showcontent
string
Content of the file in the long output.

Example:
Example of a Linux agent that uses a search string.

```
- file_content:
    file: /tmp/import_service_result
    string: specific_search_string
    returnstatus: CRITICAL
    oksummary: Import Service OK
    badsummary: Import Service FAILED
```

Example:
Example of a Windows agent that uses a search pattern.

```
- file_content:
    file: C:\Users\Operator\log.txt
    match_is_ok: false
    returnstatus: warning
    pattern: ^\s.*\d{10}.+abc.*\{\|\}~$
```

---

file_exists (Verify existence of a file or directory)

Verifies the existence of a file or directory under Linux.

Parameters:

file
string
The file or directory to monitor.

name
string
Name for the file or directory that is shown in the status summary (optional).

returnstatus
"WARNING" | "CRITICAL"
Defines the severity if the check fails (optional).

Example:

```
- file_exists:
    file: "/etc/hosts"
    name: "Important file"
    returnstatus: "WARNING"
```

---

fortinet (Fortinet controller)

Monitors the status of a controller from Fortinet Inc..

See also: fortinet_wcs on page 151

Related parameters

- snmp_connection

Parameters:

resources — true
Check the controller resources (optional).

controller — true
Check the controller status (optional).

accesspoints — true
Check the access points (optional).

Example: Monitor Fortinet controllers and access points.

```
- fortinet:
    snmp_connection:
      version: 2
      community: fortinet_ok
      port: 1234
    resources: true
    controller: true
    accesspoints: true
```

Example: Monitor Fortinet access points.

```
- fortinet:
    snmp_connection:
      version: 2
      community: fortinet_nok
      port: 1234
    accesspoints: true
```

---

fortinet_wcs (Fortinet WCS controller)

Monitors the status of a WCS controller of type WLC 500D from Fortinet Inc. in failover setups. The controllers are connected via SNMP.

Related parameters

- snmp_connection
- thresholds

Parameters:

hostname — IP_address
The IPv4 address of the main controller.

backupaddress — IP_address
The IPv4 address of the backup controller.

mainmac — MAC_address
The network device ID of the main controller.

backupmac
MAC_address
The network device ID backup controller.

check
main | backup
The controller to be checked: main controller or backup controller.

packets
number
Number of packets to send (optional).
*RST: 5

packet_interval
number
Interval between ping requests in milliseconds (optional).
*RST: 80
Default unit: ms

thresholds
rta | pl
Defines the thresholds for the round trip time (`rta`) and the packet loss (`pl`) (optional.)

rta
Defines the `warning` and `critical` thresholds for the round trip time (optional).

pl
Defines the `warning` and `critical` thresholds for the packet loss (optional).

Example:
Monitor Fortinet 500D controllers in failover setups.

```
- fortinet_wcs:
  hostname: '127.0.0.1'
  backupaddress: '128.0.0.2'
  mainmac: '11:22:33:$4:55:aa'
  backupmac: '11:22:33:44:55:bb'
  check: main
  snmp_version: 2
  snmp_community: public
  thresholds:
      rta:
          warning:    '5:'
          critical:   '500:'
      pl:
          warning:    '5:'
          critical:   '75:'
```

gb2pp (gb2pp server check over an R&S trusted filter)

Queries gb2pp servers for system or host group summary states to transfer these data via an R&S TF5900M trusted filter IP.

For details about the state aggregation logic, see Section 6.2, "Understanding aggregated states", on page 42.

The following figure shows how the status check works.

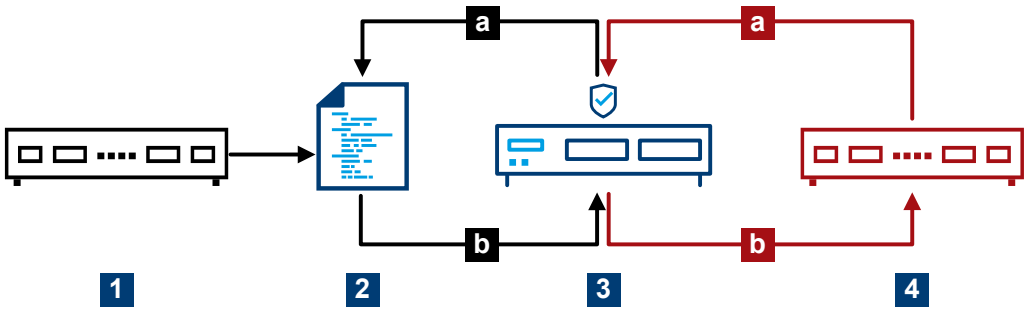


Figure 7-1: Conceptual representation of the gb2pp service check

1 = Host name: chmblack.example.net
2 = Monitoring data (gb2pp format)
3 = R&S TF5900M trusted filter IP
4 = Host name: chmred.example.net
a = Request monitoring data
b = Response

The following figure illustrates the relationship between the `health_host` key and the host name, i.e. the name of the gb2pp server.

```
hosts:
  - name: chmblack.example.net
    tags: [chm]
    connections: [gb2pp]
    # ...
    # some other attributes
    # ...

  - name: chmred.example.net
    tags: [chm]
    checks:
      # ...
      # some other checks
      # ...
      - gb2pp:
          health_host: "chmblack.example.net"
```

Figure 7-2: Relation between involved keys

The `gb2pp` check only works in combination with `hosts` on page 45 > `connections: ["gb2pp"]`.

Trusted filter devices between gb2pp server and client can possibly block TCP packets that contain TCP time stamps.

If so, disable TCP time stamps as follows:

- Run this command: `sysctl -w net.ipv4.tcp_timestamps=0`
- Add the line `net.ipv4.tcp_timestamps=0` to the default `sysctl.conf` file. You can find this file here: `/etc/sysctl.conf`.

Related parameters

- `health_host`

Parameters:

hostgroup — string

Checks the summary state of a host group (optional). Only in combination with `health_host`.

Example: **System state check**

```
hosts:
  - name: chmblack.example.net
    tags: [chm]
    connections: [gb2pp]
    # ...
    # some other attributes
    # ...

  - name: chmred.example.net
    tags: [chm]
    checks:
      # ...
      # some other checks
      # ...
      - gb2pp:
          health_host: "chmblack.example.net"
```

Example: **Host group state check**

```
hosts:
  - name: "chmblack.example.net"
    tags: ["chm"]
    connections: ["gb2pp"]
    hostgroups: ["saturn"]
    # ...
    # some other attributes
    # ...

  - name: "chmred.example.net"
    tags: ["chm"]
    checks:
      # ...
      # some other checks
      # ...
      - gb2pp:
          health_host: "chmblack.example.net"
          hostgroup: "saturn"
```

---

generic_printer (Monitor the status of network printers)

Monitors the status of network printers that support the HOST-RESOURCES-MIB.

Related parameters

- snmp_connection

Parameters:

name — string

The name for the device that is shown in the check results on the web GUI.

Example:

```
- generic_printer:
    snmp_version: 2
    snmp_community: public
    name: "My Printer"
```

---

gude (Monitor a Gude PDU)

Monitors temperature, humidity sensor and outlets of a Gude power distribution unit (PDU), e.g. Gude 8045.

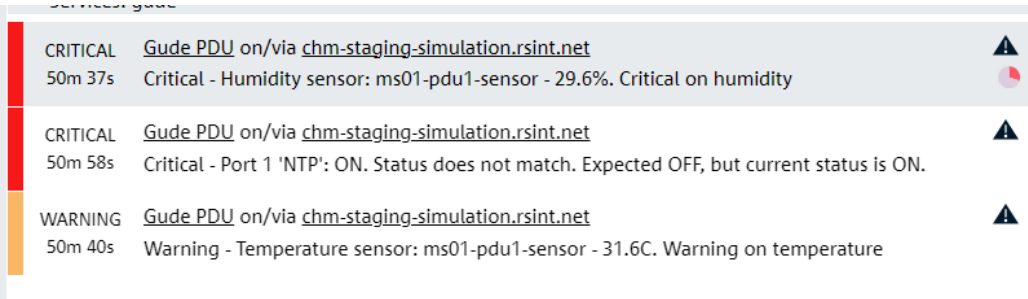


Figure 7-3: Example check results on the web GUI

Related parameters

- snmp_connection
- thresholds

Parameters:

check_temperature — true | false

Enables checking the temperature sensor (optional).

check_humidity — true | false

Enables checking the humidity sensor (optional).

port_number — numeric

Specifies the port number to check status for (optional).

expected_status — ON | OFF

The expected status for the port (optional).

Example: Check the temperature.

```
- gude:
    snmp_connection:
      version: 2
      community: gude
    check_temperature: true
    thresholds:
      temperature:
        warning: ':30'
        critical: ':40'
```

Example: Check the humidity.

```
- gude:
    snmp_connection:
      version: 2
      community: gude
    check_humidity: true
    thresholds:
      humidity:
        warning: ':80'
        critical: ':90'
```

Example: Check the port status.

```
- gude:
    snmp_connection:
      version: 2
      community: gude
    port_number: 1
    expected_status: "OFF"
```

---

hums (CHM instrument health & utilization)

Checks health and utilization data of R&S CHM instruments via LXI.

Example:

```
- hums:
```

---

icinga2_cluster (Icinga2 cluster)

Checks if all endpoints in the current Icinga2 zone and the directly connected zones are working properly.

Example:

```
- icinga2_cluster:
    logic_id: component1
```

---

idrac (Dell iDRAC hardware)

Monitors the hardware status of a server with a Dell iDRAC interface via SNMP.

Checked values

- Global system status

- Global LCD status
- System power
- Global storage status
- Power unit redundancy
- Power unit status
- Chassis intrusion sensor status
- Cooling unit status
- Status of all drives
- Predictive status of all drives
- All temperatures

If a component does not exist or if a sensor in the server version does not exist, set this check manually to `true`. For example, if there are no hard disks (diskless server), set the key `no_disks` to `true`.

Related parameters

- `snmp_connection`

Parameters:

no_storage	true Do not check global storage condition (optional).
no_system	true Do not check global system status (optional).
no_power	true Do not check global power status (optional).
no_temperature	true Do not check overall thermal environment condition (optional).
no_disks	true Do not check the disks (optional).
no_power_unit	true Do not check the power unit (optional).
no_intrusion	true Do not check the intrusion sensor (optional).
no_cooling	true Do not check the cooling unit (optional).
no_redundancy	true Do not check the power unit redundancy (optional).
no_predictive	true Do not check the predictive status of the disks (optional).
no_lcd	true Do not check the LCD status (optional).

Example:

```
- idrac:
    no_power_redundancy: true
```

---

ilo (HP iLo hardware)

Monitors the hardware status of a server with Hewlett-Packard iLO interface via SNMP.

Checked values

- Global storage status
- Global memory status
- Global system status
- Global power supply status
- Global power state (ON/OFF)
- Global thermal system
- Global temperature sensors
- Global fan status
- Disk controllers
- Power supply redundancy
- Fans
- Disk drive status
- Disk drives smart values
- Disk temperatures

If a component or a sensor does not exist, set this check manually to `true`.

Related parameters

- `snmp_connection`

Parameters:

drives	numeric Number of physical drives.
ps	numeric Number of connected power supplies.
fan	numeric Number of fans.
[no_storage]	true Do not check global storage condition (optional).
[no_system]	true Do not check global system state (optional).
no_powersupply	true Do not check global power supply condition (optional).
no_powerstate	true Do not check power state (optional).

no_temp
: true
Do not check overall thermal environment condition (optional).

no_temp_sensors
: true
Do not check temperature sensor condition (optional).

no_temp_drives
: true
Do not check the temperature sensor of the hard disk drives (optional).

no_fan
: true
Do not check global fan condition (optional).

no_memory
: true
Do not check memory condition (optional).

no_controller
: true
Do not check controller condition (optional).

no_logical_drives
: true
Do not check the logical drives (optional).

no_power_redund
: true
Do not check power supply redundancy (optional).

Example:

```
- ilo:
   drives: 2
   ps: 1
   fan: 3
   no_power_redund: true
```

---

lancom_vpn_status (Monitor VPN connection status)

Monitors the status of VPN connections on a LANCOM device via SNMP.

Related parameters

- `snmp_connection`

Parameters:

connection
: string
The name of the VPN connection to check.

Example:

```
checks:
  - snmp_hostalive:
  - lancom_vpn_status:
      connection: "PROJECT"
      snmp_connection:
        context: lancom_vpn_status
  - lancom_vpn_status:
      connection: "SECOND"
      snmp_connection:
        context: lancom_vpn_status
```

lancom_xs_gs_3000 (LANCOM device status)

Monitors the status of the hardware of a LANCOM device implementing the LCOS-SX-MIB via SNMP.

Related parameters

- `snmp_connection`

Parameters:

device_name — string

Name of the device that is shown in the status summary on the web GUI (optional).

returnstatus — WARNING | CRITICAL

Defines the severity return value if the check fails: "WARNING" or "CRITICAL" (optional).

powersupplies — string

The number of expected power supplies (optional).

*RST: 2

Example:

```
- lancom_xs_gs_3000:
    snmp_version: 2
    snmp_community: public
    device_name: "LANCOM GS-3000 Switch"
    returnstatus: "WARNING"
```

load (CPU load)

Monitors CPU load on Windows and Linux hosts.

Related parameters

- `thresholds`

Parameters:

thresholds — warning | critical

Check-specific alert levels. For more information about the threshold syntax, see `thresholds` on page 132. The following values only apply to the current load on Windows.
For Linux, see `load<minutes>`.

*RST: warning: 90, critical: 99
Default unit: %

load<minutes> warning | critical

On Linux, check load averages in the last 1 min, 5 min and 15 min (fixed). The threshold defines the utilization ratio of all processor cores.
The Linux load averages depend on the number of processor cores. For a single-core processor, a load of 1.0 means that the processor is exactly at capacity. Smaller values indicate that there is still capacity available. Higher values indicate problems, i.e. the system is slowing down or hanging.
On a multicore system, ensure that the load does not exceed the number of cores available. It does not matter how the cores are spread out over CPUs. **Two quad-cores** match **four dual-cores** match **eight single-cores**, i.e. in sum consider **eight cores** when configuring the alert levels.

Increment: 0.01
Default unit: numeric
For alert level defaults, see Table 7-4.

Example: **On Windows**

```
-load:
   thresholds:
     warning: '90'
     critical: '99'
```

Example: **On Linux**

```
- load:
    thresholds:
      load1:
        warning: '5.0'
        critical: '10.0'
      load5:
        warning: '4.0'
        critical: '6.0'
      load15:
        warning: '3.0'
        critical: '4.0'
```

Table 7-4: Load threshold defaults on Linux

Load averaging	Alert level and threshold defaults
load1	warning: 5.0
	critical: 10.0
load5	warning: 4.0
	critical: 6.0
load15	warning: 3.0
	critical: 4.0

mac_address (MAC address monitoring)

Defines a MAC address check for an SNMP-connected network device. Use it in environments that apply MAC address whitelisting so that only authorized devices communicate on the network. This check supports only SNMP-connected network devices. R&S CHM does not read MAC addresses from Windows or Linux agents without SNMP.

R&S CHM polls `ifPhysAddress` from the IF-MIB. Use `iface_no` to select an interface. Provide `iface_mac` together with `iface_no` to compare the device MAC address against an expected value. Set `debug` to list all interface names and MAC addresses in the long output.

Parameters:

iface_no — integer

Number of the network interface to monitor (optional).

iface_mac — string

Expected MAC address (optional). If you set `iface_mac` together with `iface_no`, R&S CHM compares the device MAC address with this value.

debug — string

You can set this key to anything, except an empty string (`""`). If you use `debug`, the web GUI displays all interface names and MAC addresses in the long output. You cannot combine `debug` with `iface_no`.

Example: Monitor interface `2` and compare its MAC address with a predefined value:

```
checks:
  - mac_address:
      iface_no: 2
      iface_mac: "00:11:22:33:44:55"
```

List all interface MAC addresses for debugging:

```
checks:
  - mac_address:
      debug: true
```

Usage: Setting only

manual (Add GUI button "manual" ("manual" check))

Adds a button for "passive" checks on the web GUI. If there is a device in the system that cannot be monitored, you can enter the manual status here, using the "Process manual check result" button.

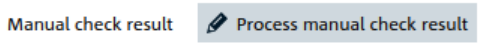


Figure 7-4: Web GUI example

To view the button on the web GUI, the user needs the permission `manual`. See `authorization` on page 68 > `permissions`.

Example:

```
checks:
  - ping:
  - manual:
      displayname: "<my_check_name>"
```

---

meinberg (Monitor Meinberg NTP)

Monitors the network time protocol (NTP) current state and GPS mode for devices that support the MBG-LANTIME-NG-MIB.

If the status is something else than "synchronized", R&S CHM returns a "WARNING" for the NTP current state and "CRITICAL" for the GPS mode.

Also, you can define a threshold for the good available satellites. E.g., if there are fewer than 5 satellites available, the status is "CRITICAL".

Related parameters

- `snmp_connection`
- `thresholds`

All keys are optional (`thresholds`, `satellites`, `warning` and `critical`).

Parameters:

satellites warning | critical

Defines the thresholds for the number of tracked satellites (optional).

*RST: warning: 5, critical: 3

Example: Monitoring configuration:

```
- meinberg:
    thresholds:
      satellites:
        warning: '10:'
        critical: '5:'
```

Example: Output on the R&S CHM web GUI:

```
"Critical - GPS Position: 48.1276 11.6124 619m.
Ntp Current State Int status: NOT_SYNCHRONIZED.
Gps Mode Int status: GPS_WARM_BOOT. Good satellitess: 7
```

---

mikrotik (MikroTik switches and router)

Monitors various aspects of a MikroTik device via SNMP.

Parameters:

check_power_supply1 true | false

Checks power supply 1 status (optional).

check_power_supply2 true | false

Checks power supply 2 status (optional).

check_fan1 true | false

Checks fan 1 speed (optional).

check_fan2 true | false

Checks fan 2 speed (optional).

check_hitemp true | false

Checks the HI temperature (optional). Not in combination with `check_devtemp`.

check_devtemp true | false

Checks the DEV temperature (optional). Not in combination with `check_hitemp`.

temperature warning | critical

If you configure a temperature check, specify the corresponding `thresholds` (optional).

Example:

```
- mikrotik:
   check_power_supply1: true
   check_power_supply2: true
   check_fan1: true
   check_fan2: true
   check_hitemp: true
   check_devtemp: false
   thresholds:
     temperature:
       warning: ":30"
       critical: ":40"
```

---

msr4 (Check R&S MSR4 via SNMP)

Monitors the the status and the temperature of the R&S MSR4 multipurpose satellite receiver via SNMP. The check also informs about the firmware version.

Related parameters

- `snmp_connection`

Example:

```
- msr4:
    snmp_version: 2
    snmp_community: public
```

---

navics (Monitor NAVICS)

Monitors the status of an IP based naval communications system from Rohde & Schwarz (NAVICS).

Related parameters

- `health_host`

Parameters:

type — server | groupserver | gw | cwp | sip | baa

Monitored NAVICS component. All components require an `equid` (equipment ID) or a `name`, except for the `groupserver`.

server
A session border control server.

groupserver
A radiotelephony control server.

gw
A media gateway.

cwp
A voice terminal.

sip
A SIP device.

baa
The NAVICS broadcast and alarm system (BAA).

master — string

If you configure `type: baa`, specify a `master` as the name of the first BAA media gateway.

agent — string

If you configure `type: baa`, specify an `agent` as the name of the secondary BAA media gateway.

equid — string

Equipment ID for type `cwp` and type `sip`.

name — string

Name for type gateway (`gw`) and type server (`server`).

Example:

```
- navics:
    type: server
    name: RADIO_SERVER
```

Example:

```
- navics:
    type: cwp
    eqid: EQID-VT-108
```

Example:

1) Typical NAVICS example

There is a host with a name `navicsbaseserver.example.net` and a service `navics`. The host is checked using `ping` and the service check is `navics`.

```
- name: navicsbaseserver.example.net
  connections: [snmp]
  snmp_connection:
    community: public
  checks:
    - ping:
    - navics:
        type: cwp
        eqid: VT1
```

This example has a disadvantage. You are monitoring voice terminals (VTs) but you cannot get the status information directly from the VTs. Instead, you ask the NAVICS server. To show the all the monitored VTs on the web GUI, you must specify them under the NAVICS server:

```
host: navicsbaseserver.example.net
   - service 1: Voice Terminal 1 status
   - service 2: Voice Terminal 2 status
   - service 3: Voice Terminal 3 status
   - service 4: Voice Terminal 4 status
   - ...
   - service 199: Voice Terminal 199 status
```

On the web GUI, all voice terminal status values are then also listed below the NAVICS server.

Example:

2) NAVICS example using the `health_host` key

To increase the overview of voice terminals on the web GUI, you can show every instance as a single host. If you can ping the voice terminals directly, you ask the NAVICS server for status information. Of course, you also configure the NAVICS server in the `chm.yaml` file.

```
- name: VT1.navics
    connections: [snmp]
    checks:
      - ping:
      - navics:
          health_host: navicsbaseserver.example.net
          type: cwp
          eqid: VT1

- name: navicsbaseserver.example.net
    connections: [snmp]
    snmp_connection:
      community: public
    checks:
      - ping:
```

If you cannot ping the voice terminals, you can configure a logic function as described in `logic` on page 52. Here, you also find a detailed NAVICS configuration example.
See also: `health_host` on page 126

Example:

3) Monitoring of the NAVICS broadcast and alarm system

```
- navics:
    type: baa
    master: ZYNQ_RGW3
    agent: ZYNQ_RGW4
    health_host: chm-staging-simulation.rsint.net
    snmp_connection:
      context: navics
```

---

nport (Moxa NPort 6000 series server)

Monitors a Moxa NPort 6000 series serial server via SNMP.

Supported MIBs

- RFC 1213-MIB
- MOXA-NP6000-MIB

Related parameters

- `snmp_connection`

Parameters:

serial_port — numeric

Monitored serial port.

name — string
Port name.

errormessage — string
Additional error message that indicates the status failure.

returnstatus — "CRITICAL" | "WARNING"
Returns status for failures.

dsr , cts , dtr — HIGH | LOW
Checks for the serial DSR, CTS or DTR flow control if the OK status is `HIGH` or `LOW`.

counter — Checks the port for frame, break, overrun and parity error counters (optional).

Example:

```
-nport:
   serial_port: 2
   dtr: LOW
   dsr: HIGH
   cts: LOW
   errormessage: "GENERATOR FAILED"
   name: "GENERATOR INPUT"
   returnstatus: "WARNING"
   counter:
-nport:
   serial_port: 3
   dtr: LOW
   errormessage: "AIRCONDITION FAILED"
   counter:
```

---

ntp_time (NTP server time synchronization)

Monitors time synchronization with an NTP server running on Windows or Linux. Only UTC time is used for calculating time offsets between client and server, even if your NTP client or server uses other timezones to display daytime.

Related parameters

- `thresholds`

Parameters:

server — FQDN | IP_address
FQDN, IPv4 or IPv6 address of the NTP server.

port — numeric
NTP port of the server (optional).
*RST: 123

timeout — time

Seconds before connection times out (optional).

*RST: 10
Default unit: s

offset — time

Expected time offset in seconds. Thresholds get adjusted automatically (optional).

*RST: 0
Default unit: s

thresholds — warning | critical

Alert levels for time offset to NTP server (optional).
For more information about the `thresholds` syntax, see `thresholds` on page 132.

*RST: warning: '-0.1:0.1', critical: '-0.5:0.5'
Default unit: s

Example:

```
- ntp_time:
    server: ntpserver.example.com
    port: 12345
    timeout: 5
    offset: 3600
    thresholds:
      warning: '-0.5:0.5'
      critical: '-1:1'
```

---

nw_interface (Network interface)

Monitors the status of the network interface of devices that implement the RFC 1213-MIB via SNMP.

Checked values

- Speed of the network interface
- Operational status
- Administrative status
- Port security MAC based
- Port security 702.1x based

Related parameters

- `snmp_connection`

Specify the interface properties and select one or more of the following checks and their defined "OK" status.

Parameters:

interface — numeric

Network interface to be monitored.

*RST: 1

name
: string
Name for the interface (optional).

errormessage
: string
An additional error message that is shown if the status fails (optional).

returnstatus
: WARNING | CRITICAL
Return value if the check fails (optional).

speed
: numeric
Speed of the network interface, e.g. 100, 1000 Mbit/s (optional).
*RST: 1000
Default unit: MBit/s

op_status
: UP | DOWN | TESTING | UNKNOWN | DORMANT | NOTPRESENT | LOWERLAYERDOWN
Check the operational status of the network interface (optional).

admin_status
: UP | DOWN | TESTING
Administration status of the network interface (optional).

port_sec_mac
: Checks if the MAC-based port security status of a device that is compatible with CISCO-PORT-SECURITY-MIB (optional).

port_sec_802
: Checks if the 802.1-based port security status of a device that is compatible with CISCO-PAE-MIB (optional).

port_sec_ieee802
: Checks if the PAE auth controlled port status of an interface is "AUTHORIZED" of a device that is compatible with the IEEE8021-PAE-MIB (optional).

Example:

```
- nw_interface:
    interface: 2
    speed: 1000
    op_status: UP
    admin_status: UP
    errormessage: "Failure on network interface for server"
    name: "server interface"
    returnstatus: "WARNING"
    port_sec_mac:

- nw_interface:
    interface: 3
    speed: 100
    port_sec_802:

- nw_interface:
    interface: 4
    port_sec_ieee802:
```

os_disk (Disk space)

Monitors available disk space.

Parameters:

include ['<drive or volume>' , 'drive or volume']

List of drives (on Windows) or volumes (on Linux) that are monitored (optional). If not set, R&S CHM monitors all disks or volumes.

*RST: none

thresholds warning | critical

Alert levels for available disk space (optional).
On Windows: **used** disk space.
On Linux: **free** disk space.
For more information about the `thresholds` syntax, see `thresholds` on page 132.

Range: 0 to 100
*RST: none (Windows) , 10 (Linux warning) , 20 (Linux critical)
Default unit: %

Example: For a Windows host

```
- os_disk:
    include: ['C', 'F']
    thresholds:
      warning: '80'
      critical: '90'
```

Example: For a Linux host

```
 - os_disk:
     include: ['/', '/boot']
     thresholds:
       warning: '10:'
       critical: '5:'
```

os_memory (Memory usage)

Monitors RAM usage and detects when your operating system is about to swap.

Related parameters

- `thresholds`

Example:

```
- os_memory:
    thresholds:
      warning: '10:'
      critical: '5:'
```

os_process (Operating system process)

Monitors if a defined process is running on the system.

Parameters:

name — Name of the process. If at least one instance is found, the check is OK.

commandline — The check is performed against the command line of the process (optional).
If at least one instance is found, the check is OK.
On Linux: Regex is supported. For escaping special characters, use a backslash (\).
On Windows: Wildcards are supported (see: https://docs.microsoft.com/en-us/windows/win32/wmisdk/like-operator)

Example: Checking for the process name:

```
- os_process:
    name: rsyslogd
```

Example: Checking for the command line on Windows:

```
- os_process:
    name: svchost
    commandline: "%svchost%Unistack%"
```

Example: Checking for the command line on Linux:

```
- os_process:
    name: icinga2
    commandline: icinga2.*daemon
```

os_service (Monitor Windows service status)

Monitors if a Windows service is in status "Running".

Parameters:

name — string

Specify the Windows "Service name". If the service is not in status "Running", the status is indicated as "CRITICAL" on the web GUI.

Example: Monitor the status of the "Icinga2" service:

```
- os_service:
    name: "Icinga2"
```

passive (Aggregated host status)

Adopts the aggregated status from a logic function instance and shows this status on the web GUI.

Depending on the position in the configuration, passive has two meanings:

- If you specify `passive` as the first host check, it results in a logic host check.

- If you specify `passive` after other checks, it results in a service check with a logic function instance.

Parameters:

src_logic_id <log_func_inst>

Specify here one of the configured logic function instances. You can select from an instance that is configured in `logic` on page 52 or `logic_id` on page 126.

Example:

```
checks:
  - passive:
      src_logic_id: aggregation1
```

See also: Example in `logic` on page 52

---

ping (Host availability ("ping" check))

Checks the availability of a host. To do so, R&S CHM sends ICMPv4 or ICMPv6 requests to the hosts.

This check cannot verify if the R&S CHM service runs on an agent. To check this property, use `chm_agent_connection`, see `chm_agent_conn` on page 136.

Related parameters

- `thresholds`

Parameters:

threshold

Thresholds for returned values of the `ping` command, i.e. `rta` and `pl`.

rta

warning | critical

Round-trip average time (optional).

*RST: 3000 | 5000
Default unit: ms

pl

warning | critical

Package loss (optional). Since five packages are sent, we recommend specifying one of the values 0, 20, 40, 60, 80, or 100.

*RST: 80 | 100
Default unit: %

packets

Number of packets to send (optional). For a fast detection and a reduced CPU load, we recommend sending only one ICMP package.

*RST: 5

packets_interval

Interval between ping requests in milliseconds (optional).

*RST: 80
Default unit: ms

timeout

Maximum time in seconds to wait for the ping operation (optional). For a fast detection, we recommend setting a small timeout. Consider the amount of packets, i.e. the `packet_interval` and the critical `rta` threshold. For example, 1 package with a rta of 300 ms results in a timeout of approximately 0.5 s.

*RST: 10
Default unit: s

Example:

```
checks:
  - ping:
      threshold:
        rta:
          warning: '500'
          critical: '1000'
        pl:
          warning: '60'
          critical: '80'
```

Example:

To increase speed and performance usage in FIPS enabled systems:

```
checks:
  - ping:
      thresholds:
          rta:
              warning: 5
              critical: 6
          pl:
              warning: 7
              critical: 8
      packets: 1
      packet_interval: 10
      timeout: 1
```

---

raritan_pdu (Checks Raritan PDU outlets)

Checks the outlet status of a Raritan power distribution unit (PDU) that is connected using SNMP.

Related parameters

- `snmp_connection`
- `thresholds`

Parameters:

outlet_number

numeric

The number of the power outlet to query.

sensor_number

numeric

The sensor number to check the status for.

sensor_type

absoluteHumidity | activeEnergy | activeInlet | activePower | airFlow | airPressure | apparentEnergy | apparentPower | binary | contact | crestFactor | displacementPowerFactor | doorContact | doorHandleLock | doorLockState | fanSpeed | fanStatus | frequency | humidity | i1smpsStatus | i2smpsStatus | illuminance | inletPhaseSync | inletPhaseSyncAngle | motionDetection | none | onOff | operatingState | other | overheatStatus | overloadStatus | peakCurrent | phaseAngle | powerFactor | powerQuality | rcmState | reactivePower | residualCurrent | residualDcCurrent | rmsCurrent | rmsVoltage | rmsVoltageLN | smokeDetection | surgeProtectorStatus | switchStatus | tamperDetection | temperature | trip | unbalancedCurrent | vibration | waterDetection

The type of sensor to query the status for.

expected_status

aboveUpperCritical | aboveUpperWarning | alarmed | belowLowerCritical | belowLowerWarning | closed | critical | detected | fail | fault | i1OpenFault | i1ShortFault | i2OpenFault | i2ShortFault | inSync | no | nonRedundant | normal | notDetected | off | ok | on | one | open | outOfSync | selfTest | standby | two | unavailable | warning | yes

The expected status for the outlet.

Example:

```
- raritan_pdu:
    snmp_version: 2
    snmp_community: public
    outlet_number: 2
    expected_status: "on"

- raritan_pdu:
    snmp_version: 2
    snmp_community: public
    sensor_number: 1
    sensor_type: onOff
    expected_status: "normmal"

- raritan_pdu:
    snmp_version: 2
    snmp_community: public
    sensor_number: 1
    sensor_type: humidity
    thresholds:
      humidity:
        warning: ":80"
        critical: ":90"
```

Table 7-5: Possible sensor states for each sensor type

rmsCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
peakCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
unbalancedCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical

rmsVoltage	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
activePower	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
apparentPower	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
powerFactor	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
activeEnergy	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
apparentEnergy	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
temperature	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
humidity	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
airFlow	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
airPressure	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
onOff	unavailable, on, off
trip	unavailable, open, closed
vibration	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
waterDetection	unavailable, normal, alarmed
smokeDetection	unavailable, normal, alarmed
binary	unavailable, normal, alarmed
contact	unavailable, normal, alarmed
fanSpeed	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
surgeProtectorStatus	unavailable, ok, fault
frequency	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
phaseAngle	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
rmsVoltageLN	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
residualCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
rcmState	unavailable, normal, warning, critical, selfTest, fail
absoluteHumidity	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
reactivePower	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
other	unavailable
none	unavailable
powerQuality	unavailable, normal, warning, critical
overloadStatus	unavailable, ok, fault
overheatStatus	unavailable, ok, fault
displacementPower-Factor	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
residualDcCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
fanStatus	unavailable, ok, fault

inletPhaseSyncAngle	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
inletPhaseSync	unavailable, inSync, outOfSync
operatingState	unavailable, normal, standby, nonRedundant, bypassActive, off
activeInlet	unavailable, one, two, off
illuminance	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
doorContact	unavailable, open, closed
tamperDetection	unavailable, normal, alarmed
motionDetection	unavailable, normal, alarmed
i1smpsStatus	unavailable, ok, fault
i2smpsStatus	unavailable, ok, fault
switchStatus	unavailable, ok, i1OpenFault, i1ShortFault, i2OpenFault, i2ShortFault
doorLockState	unavailable, open, closed
doorHandleLock	unavailable, open, closed
crestFactor	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
length	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
distance	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
activePowerDemand	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
residualAcCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
particleDensity	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
voltageThd	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
currentThd	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
inrushCurrent	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
unbalancedVoltage	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
unbalancedLineLine-Current	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
unbalancedLineLine-Voltage	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
dewPoint	unavailable, belowLowerCritical, belowLowerWarning, normal, aboveUpperWarning, aboveUpperCritical
transferSwitchBy-passState	unavailable, inactive, i1Selected, i1SelectedAndActive, i2Selected, i2SelectedAndActive, fault
installFaultStatus	unavailable, ok, fault
transferSwitchOutput-Status	unavailable, ok, fault

RCCM (Monitor RCCM device)

Monitors a device via RCCM device access layer API.

Related parameters

- `health_host`

Parameters:

device_id — String
Unique device identifier.

scheme — String
Device access layer API URL scheme (optional).

port — Number
Device access layer API port (optional).

Example: Minimum configuration example.

```
- rccm:
    device_id: s5200-sim-transmitter
```

Example: Configuration with health host definition.

```
- rccm:
    device_id: s5200-sim-transmitter
    health_host: device-access.foo.net
```

---

snmp (SNMP OID check)

Checks individual SNMP OIDs of a host for their return value. R&S CHM shows the status of the host with optional status message on the web GUI.

Status indication on the web GUI:

- "OK" if the returned value matches the expected value.
- "CRITICAL" if the returned value does not match the expected value.

Related parameters

- `snmp_connection`

Parameters:

oid — string
The SNMP OID to be checked.

expected — string
The expected return value.

okmessage — string
Show this message if the returned value matches the expected value.

criticalmessage — string
Show this message if the returned value does not match the expected value.

hwinfo — true

If you specify `hwinfo: true`, R&S CHM queries the System-Descr OID and shows it on the web GUI > "Host" > "Result" (optional). The OID contains some basic information like the firmware version (if applicable). The check always returns as "OK".

Example:

```
checks:
  - snmp:
      snmp_connection:
        version: 2
        community: public
      oid: ".1.3.6.1.4.1.9.9.500.1.2.1.1.6"
      expected: "4"
      okmessage: "Cisco Switch State is READY"
      criticalmessage: "Cisco Switch State NOT READY"
```

The following is output on the web GUI if the check was successful:
"OK - Cisco Switch State is READY"

---

snmp_diskspace (Check server diskspace via SNMP)

Monitors disk usage of a server via SNMP and reports partition names that are in critical state.

Related parameters

- `snmp_connection`

Parameters:

max-disk-usage — Threshold for the disk usage check (optional).

*RST: 90
Default unit: %

Example:

```
- snmp_diskspace:
    snmp_version: 2
    snmp_community: public
    max-disk-usage: 80
```

---

snmp_flex (Generic SNMP check)

Runs the generic YAML-based SNMP check. The SNMP transport and credentials are taken from the host's SNMP connection.

For details about writing custom YAML configurations, see Section 6.11, "Configuring generic monitoring of SNMP third-party devices", on page 90. For additional information about snmp_flex configuration in the `chm.yaml`, see Section 6.11.10, "Configuring snmp_flex in the chm.yaml", on page 106.

You can include the configuration as follows. The keys `config` and `config_content` are mutually exclusive. One of them is required.

- `config`
  Path to a YAML configuration file that already exists on the monitoring node (recommended).
- `config_content`
  An inline YAML mapping. R&S CHM serializes it verbatim, deploys it to the node and points the plugin at the deployed file.

Parameters:

config	File path Path to an existing `snmp_flex`-based YAML configuration on the node.
config_content	Inline `snmp_flex` YAML configuration (mapping) deployed by R&S CHM.
vars	String List of `KEY=VALUE` pairs that are passed as repeated `--var` to R&S CHM.

Example: Path mode with variables.

```
- snmp_flex:
    config: /etc/rohde-schwarz/snmp_flex/mydevice.yaml
    vars:
      - threshold_warn=60
      - threshold_crit=90
```

Example: Path mode with distributed file name.

```
- snmp_flex:
    config: mydevice.yaml
```

Example: Inline mode.

```
- snmp_flex:
    config_content:
      name: My device
      vars:
        threshold_warn: 60
        threshold_crit: 90
      checks: []
```

---

snmp_hostalive (Host availability ("snmp_hostalive" check))

Checks the availability of a host. To do so, the check sends an SNMP `GetNext` request targeting some OID close to the MIB root to the target host. If the host sends a response without SNMP error indication or status, the host is considered to be up and running.

You can use the check to determine if a host is "UP" or "DOWN" if ICMP is blocked in a system by a firewall.

Related parameters

- snmp_connection

Example:

```
checks:
  - snmp_hostalive:
      snmp_connection:
        port: 1234
        community: public
```

---

snmp_processes (Check Linux operating system processes via SNMP)

Monitors the Linux processes running on the system via SNMP.

Prerequisites

Configure the SNMP daemon on a Linux system (net-snmpd) to watch specific services with `proc <service>` statement.

Related parameters

- snmp_connection

Example:

```
- snmp_processes:
    snmp_version: 2
    snmp_community: public
```

---

snmp_time (Check time offset to R&S CHM host)

Compares the time of a device with the time of the R&S CHM host using SNMP. The check supports all SNMP versions and can use all SNMP arguments.

Related parameters

- snmp_connection
- thresholds

Parameters:

tzoffset — numeric

Offset between the remote device and the R&S CHM host (in min).

localtime — boolean

Comparison method.

true
Compares remote time with the local time of the R&S CHM host.

false
Compares remote time with UTC.

thresholds — Thresholds for this status check.

offset — warning | critical

Thresholds for the time offset between device and server (in s).

*RST: 5 | 10
Default unit: s

Example:

```
checks:
  - snmp_time:
      tzoffset: 60
      localtime: false
      thresholds:
        offset:
          warning: '-20:20'
          critical: '-60:60'
```

---

spectracom_time (Spectracom time)

Monitors a Spectracom SecureSync time server via SNMP.

Checked values

- Status of AC and DC power supply
- Major and minor alarms
- GPS reference antenna status
- GPS reference time validity
- System synchronization status
- System holdover status
- Number of satellites

Supported MIBs

- SPECTRACOM-SECURESYNC-MIB

Tested devices

- Spectracom SecureSync GT4030

Related parameters

- `snmp_connection`

Parameters:

name	string The name for the device that is shown in the check results. *RST: Spectracom SecureSync
no_acpower	Do not check the AC power status (optional).
no_dcpower	true Do not check the DC power status (optional). This key requires that you specify `no_dcpower: true` in the configuration file.
no_minor_alarm	Do not check for minor system alarms (optional).
no_major_alarm	Do not check for major system alarms (optional).
no_ref_time_validity	Do not check the GPS ref time validity (optional).
no_sync_state	Do not check the system sync status.
no_holdover_state	Do not check the system holdover status (optional).

no_ref_antenna_state Do not check the GPS ref antenna status (optional).

no_tracked_satellites Do not check the number of tracked satellites (optional).

thresholds Check-specific alert levels (optional). For more information about the threshold syntax, see `thresholds` on page 132.

tracked_satellites warning | critical

Defines the `thresholds` for the number of tracked satellites (optional).

boolean

*RST: warning: '5:' , critical: '3:'

Example:

```
- spectracom_time:
    name: Spectracom GT4030
    no_dcpower:
    thresholds:
      tracked_satellites:
        - warning: '5:'
        - critical: '3:'
```

---

ssh (Establish SSH connection)

This check attempts to establish an SSH connection to the specified host and port.

The check results:

- If the connection is successful:
  - The check returns "OK".
  - The check returns "CRITICAL" in combination with `ssh_negate: True`.
- If the connection fails:
  - The check returns "CRITICAL"
  - The check returns "OK" in combination with `ssh_negate: True`.

Parameters:

ssh_port integer

The port number on which the SSH server is listening (optional). Default port is 22.

ssh_timeout integer

The timeout (in s) for the SSH connection (optional). Default is 1 s.

ssh_negate True | False

If set to `True`, the check returns "OK" if the SSH server is unreachable or the connection is refused. Default is `False`.

Example:

```
checks:
  - ssh:
      ssh_port: 22
      ssh_timeout: 10
      ssh_negate: False
```

Example:

```
checks:
  - ssh:
      ssh_timeout: 5
```

Example:

```
checks:
   - ssh:
       ssh_negate: True
```

---

synology (Synology NAS)

Monitors various aspects of a Synology NAS via SNMP.

Checked values

- Global temperature
- Power unit status
- Fan status
- Status of all drives
- RAID status

Related parameters

- `snmp_connection`

Parameters:

no_temperature — true
Do not check overall thermal environment condition (optional).

no_power_status — true
Do not check global power status (optional).

no_fan_status — true
Do not check the fan status (optional).

no_disks — true
Do not check the disk status (optional).

no_raid — true
Do not check the RAID status (optional).

Example:

```
- synology:
    no_fan_status: true
```

---

system_state (Enable client interface and check logic)

Enables the Windows client interface and the check logic. The check shows the aggregated state of the entire system on the web GUI. The clients connect to this check and replicate the status to the notification icon. If there is a change in any check, the `system_state` check mirrors that state.

Example:

```
hosts:
  - name: host1.de
    tags: [chm]
    checks:
      - ping:
      - system_state:
```

How to: Section 4.3, "Installing R&S CHM clients", on page 23

---

tcp (TCP port connectivity check)

Checks if a TCP port is open and reachable from the R&S CHM host.

Parameters:

port_number The port to be checked.

Example:

```
- tcp:
    tcp_port: 22
```

---

tmr_radio (TMR-MIB compatible radio)

Shows the mode of TMR-MIB compatible radios. Such devices can be in normal mode or control mode. The check returns the state of the radio by using the plugin output. For *normal*, the web GUI shows "OK - Normal Mode", for *control*, it shows "OK - Control Mode". If anything else is returned, the web GUI shows "UNKNOWN - Unknown Mode (n)".

Example:

```
checks:
  - tmr_radio:
```

---

trustedfilter (Monitor R&S TF5900M trusted filter IP)

Monitors the status of the R&S TF5900M trusted filter IP firewall, which secures the boundaries of networks with different classified material domains:

- Status power supply unit 1/2
- Status power fan unit 1/2
- Status internal voltage
- Status internal temperature
- Error status
- Activity state
- Status log fill level

Related parameters

- `snmp_connection`

Example:

```
checks:
  - trustedfilter:
      snmp_connection:
        version: 2
        community: public
```

---

ups (Uninterruptible power supply - RFC1628-compatible)

Monitors a UPS that is compatible to RFC 1628 via SNMP.

Related parameters

- `snmp_connection`
- `thresholds`

Select one of the following checks. Each check returns a single metric.

Parameters:

alarms	Checks the present number of active alarm conditions. In combination with `thresholds`, R&S CHM generates an alert.
secondsonbattery	Checks if the unit is running on battery power? If not, the UPS returns zero. If the unit is not running on battery power the following is checked, whichever is less: The elapsed time since the UPS last switched to battery power. – or – The time since the network management subsystem was last restarted. In combination with `thresholds`, R&S CHM generates an alert. Default unit: s
minutesremaining	Checks estimated time to battery charge depletion under the present load states in the following cases: The utility power is off and remains off. – or – The utility power is going to be lost and remains off. In combination with `thresholds`, R&S CHM generates an alert. Default unit: min
thresholds	Check-specific alert levels. For more information about the threshold syntax, see `thresholds` on page 132.

Example:

```
- ups:
   alarms:
   thresholds:
      critical: '0'
- ups:
   secondsonbattery:
   thresholds:
      warning: '0'
      critical: ':10'
- ups:
   minutesremaining:
   thresholds:
      warning: '20:'
      critical: '10:'
```

---

vmware (VMware ESXi/vcenter server inventory)

Monitors a VMware ESXi/vcenter server, e.g. datastores. You can specify up to four checks for a host.

Available checks

- Alarms
- Datastore usage
- CPU usage
- Memory usage

Related parameters

- `thresholds`

Parameters:

user	string The user name that is used to log in at the server.
insecure	false | true Checks the server certificate (optional). The server certificate is checked (`'false'`) or *not* checked (`'true'`). *RST: false
type	alarm | datastore | hostsystem The entity type of the monitored object.
alarm	Currently not acknowledged alarms on the alarm list result in an alert with the severest alarm state, i.e. warning or critical.
datastore	Gets used disk space on datastore objects.
hostsystem	Gets CPU and memory usage on all HostSystem objects, i.e. ESX(i) hosts. See also the thresholds parameter.

id — string

The unique identifier for the monitored object (optional). If no `id` is given, all objects of the specified type are checked. E.g., for datastores, `id` is the name of the datastore. The parameter is not supported for `alarm` and `hostsystem`.

port — numeric

Port of the VMware vSphere API (optional).

*RST: 443

thresholds — cpu | memory

Check-specific alert levels (optional). For more information about the `thresholds` syntax, see `thresholds` on page 132. The thresholds for the datastore usage define the used datastore space (in %).

cpu
Usage of CPU (in %).

memory
Usage of RAM (in %).

Example:

```
- vmware:
    user: axolotl
    type: datastore
    id: mydatastore
    thresholds:
      warning: '90'
      critical: '95'
- vmware:
    user: axolotl
    type: alarm
- vmware:
    user: axolotl
    type: hostsystem
    thresholds:
      cpu:
        warning: '90'
        critical: '95'
      memory:
        warning: '98'
        critical: '99'
```

---

windowsupdateage (Windows security update)

Checks if at least one Windows security update was installed within the last given number of days.

Related parameters

- `thresholds`

Parameters:

thresholds — warning | critical

Alert levels for the age of the definition files (optional).

*RST: critical: '20'
Default unit: d

Example:

```
- windowsupdateage:
    thresholds:
      critical: '100'
```

---

xcp_ng (Check XCP-ng system)

Checks the disk usage and status of the virtual machine of an XCP-ng system. XCP-ng is a high-performance enterprise-level virtualization platform. An XCP-ng server is dedicated entirely to running XCP-ng and hosting virtual machines.

Do not use deprecated keys any longer. Use `ssh_connection` instead.

Related parameters

- `ssh_connection`

Parameters:

user — string

User name used for the SSH connection (optional, deprecated).

*RST: root

password — string

Password for the SSH user (optional, deprecated).
See also: Section 6.7, "Managing password identifiers", on page 76

port — numeric

Port of the remote SSH server (optional, deprecated).

*RST: 22

ssh-timeout — numeric

Maximum time to wait for the SSH handshake to succeed (optional, deprecated).

*RST: 2
Default unit: s

identity-file — string

Path to a private SSH key (optional, deprecated).

check-disk-usage — string

Check disk usage on the VM host (optional).

max-disk-usage — Threshold for the disk usage check (optional).

*RST: 90
Default unit: %

check-vm Check if the specified VM is running (optional).

Example:

```
- xcp_ng:
    ssh_connection:
      username: peter
      password: VAULT:peter
    check-disk-usage: true
    max-disk-usage: 75
```

8 YAML configuration examples

This section provides some examples for configuration of hosts and services in the YAML configuration file.

- R&S CHM host configuration 191
- Linux host configurations 192
- Example configuration for R&S CHM Windows agents 194
- Example configuration for R&S CHM Linux agents 195

8.1 R&S CHM host configuration

The following YAML code snippet shows the top part of the configuration file with the definition of the R&S CHM host. For configuration details, see Section 6.4, "Configuring hosts", on page 44.

```
hosts:
  - name: host1.de
    tags: [chm]
    authentication:
      monitoring:
        - ldap:
            server: ldapserv.ourlocal.net
            port: 35636
            encryption: ldaps
            base_dn: ou=ldap_users,dc=ldapserv,dc=ourlocal,dc=net
            user_class: user
            user_name_attr: sAMAccountName
            bind_dn: service_user
            bind_pwd_path: ldap/service_user
    authorization:
      monitoring:
        roles:
          admin:
            permissions:
              - check
              - acknowledge
              - comment
              - downtime
            users:
              - admin
              - armin
            groups:
              - G_Admins
              - G_Armins
          superoperator:
            permissions:
              - acknowledge
```

```
        users:
          - supop
      special:
connections: [local]
hostgroups: [monitoring, control]
checks:
  - icinga2_cluster:
      checkgroups: [cluster, buster]
  - dhcp:
      displayname: Check our awesome DHCP servers
      servers: 192.168.1.253, 192.168.1.254
      interface: eth0
  - dns:
      displayname: Check our insane DNS servers
      lookup: somehosttolookup.ourlocal.net
      server: 192.168.1.254
      answers: 192.168.1.10, 192.168.1.11
      authoritative: true
      accept_cname: true
      timeout: 15
      thresholds:
        warning: '5'
        critical: '10'
```

8.2 Linux host configurations

Here, you can find some examples for Linux host configurations.

Example: host3.de

```
- name: host3.de
    connections: [icinga2_linux]
    checks:
      - os_process:
          name: test
      - load:
          thresholds:
            load1:
              warning: '9'
              critical: '10'
            load5:
              warning: '8'
              critical: '9'
      - os_disk:
          include: ['/', '/boot']
          thresholds:
            warning: '10:'
            critical: '5:'
      - ntp_time:
          server: ntpserver.example.com
          thresholds:
            warning: '1'
            critical: '2'
```

Example: chm2-test-linux-node.rsint.net

```
- name: chm2-test-linux-node.rsint.net
    connections: [icinga2_linux]
    hostgroups: [oumuamua]
    checks:
      - ping:
      - os_memory:
      - os_disk:
          include: ['/', '/boot']
          thresholds:
            warning: '10:'
            critical: '5:'
      - nport:
          checkgroups: [water, earth, fire, air]
          snmp_connection:
            version: 3
            context: nport
            secname: rsadmin # lookup of passwords in password store
            authproto: MD5
            privproto: DES
            port: 1234
          serial_port: 1
          cts: LOW
          errormessage: "GENERATOR FAILED"
          name: "GENERATOR INPUT"
          returnstatus: "WARNING"
```

8.3 Example configuration for R&S CHM Windows agents

Example for an R&S CHM Windows agent configuration.

Example:

```
- name: chm2-win
  hostgroups: [ "Computers - Windows" ]
  connections: [ icinga2_win ]
  checks:
    - ping:
    - os_memory:
    - os_disk:
        include: [ C, D ]
        thresholds:
          warning: '85'
          critical: '90'
```

8.4 Example configuration for R&S CHM Linux agents

Example for an R&S CHM Linux agent configuration.

Example:

```
- name: chm2-linux-node.rsint.net
  connections: [ icinga2_linux ]
  hostgroups: [ "Computers - Linux" ]
  checks:
    - ping:
    - os_memory:
    - os_disk:
        include: [ '/', '/boot', '/var' ]
        thresholds:
          warning: '10:'
          critical: '5:'
```

9 Troubleshooting

This section informs about problems that can occur and provides basic troubleshooting procedures. Problems that apply to the web GUI probably cannot be resolved by operators or administrators due to missing privileges. Then, contact the system administrator to resolve these problems.

9.1 Web GUI is unavailable

If the web GUI is unavailable, possibly the services are not up and running on the R&S CHM system status monitoring host.

Resolution

► Check if this service is running on the R&S CHM host:
`systemctl status chm`

► Restart these services on the R&S CHM host:
`systemctl restart chm`

See also: "To edit the configuration file" on page 44.

9.2 Web GUI shows message Wrong SNMP PDU digest

Or you can see the SNMP error "No SNMP response received before timeout".

Resolution

► Check the SNMP settings on the device, i.e. `snmp_connection` keys `context`, `authpass`, `privpass`, `authproto`, etc. The configuration in the `chm.yaml` file does not match the monitored device.

See also: `snmp_connection` on page 128

9.3 Web GUI shows 404 error

This error is a standard HTTP error message code. It means that the website that you were trying to reach could not be found on the server. One of the possible causes is that the LDAP server is not reachable.

Resolution

1. Ensure that the LDAP server is up and running.

2. If you cannot fix the problem, consider disabling LDAP in the YAML configuration to access the web GUI using a local user account.
   To disable LDAP, see Section 6.5, "Configuring web GUI users", on page 61.

9.4 Troubleshooting installation problems on Windows agents

9.4.1 Accessing the event log

If you experience problems during installation of Windows agents using the CHM_Windows_Agent_<version>.exe installer, you can find troubleshooting information in the Windows Event Viewer.

1. Select ⊞.
2. Type *event viewer*.

   The Event Viewer opens.
3. In the left navigation area, select "Custom Views" > "CHM Agent".

   The events from the R&S CHM Windows agent installation are listed.

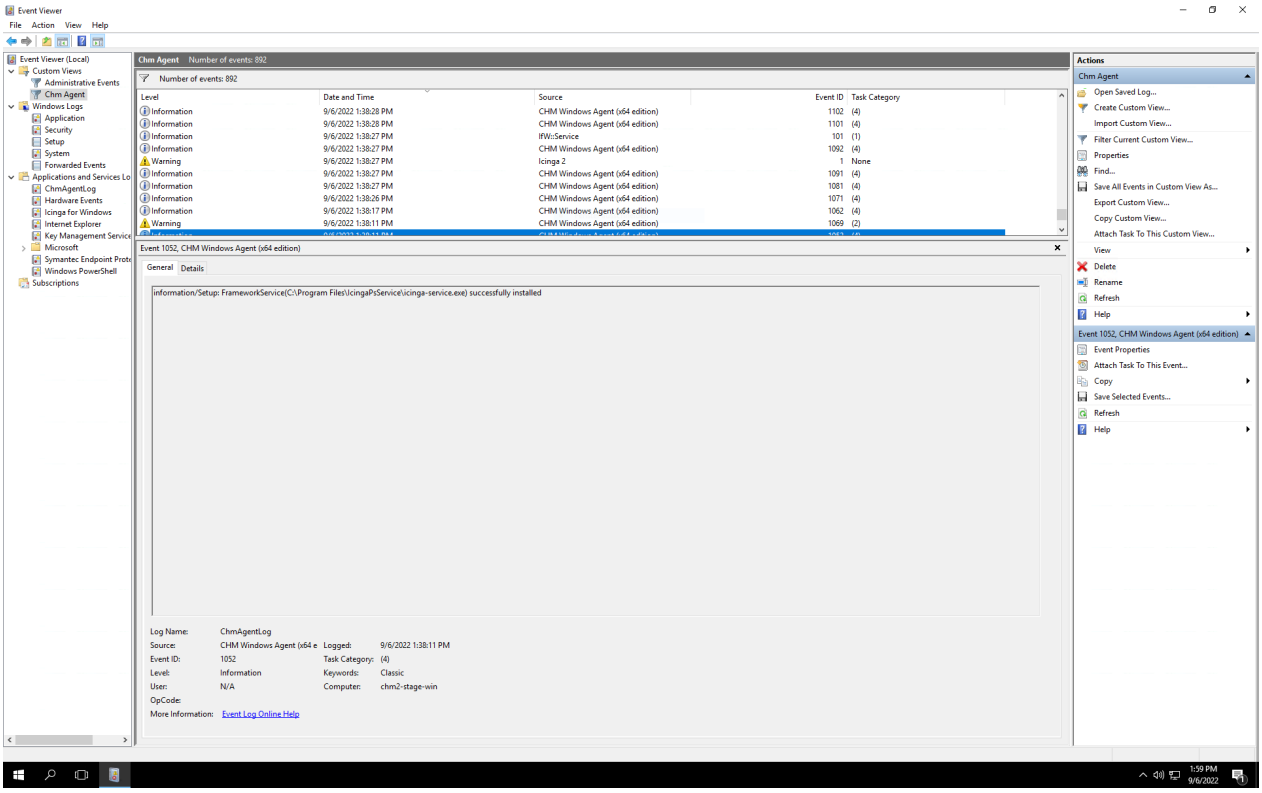


Figure 9-1: Event Viewer - logs from R&S CHM (example)

9.4.2 Accessing the MSI log files

When troubleshooting issues with a microsoft installer (MSI) package, it is often necessary to debug the installer and read the logs generated during the installation process. This guide helps you locate and interpret these logs, specifically under this directory:

`%localappdata%\Temp`.

1. Locate the log file.
   a) In a file explorer, open this directory:
      `%localappdata%\Temp`
   b) Look for this log file:
      `CHM_Windows_Agent_(x64_edition)_*.msi` (or the name you specified)
2. Use a text editor to open the log file, e.g. Notepad++

   The log file contains detailed information about each step of the installation process.
3. Check the log file for problems. The following list helps identify possible issues.
   - Start and end of installation:
     Look for entries that indicate the start and end of the installation process. These entries help you understand the sequence of events.

- Action start and action end:
  The installer logs each action with "Action start" and "Action ended" entries. These entries help you identify which actions were successful and which ones failed.
- Error messages:
  Search for the keyword "Error" to locate any error messages. Error messages typically include an error code and a brief description of the issue.
- Return values:
  Each action ends with a return value. A return value of 1 indicates success, while other values indicate different types of failures.

9.5 Contacting customer support

Technical support – where and when you need it

For quick, expert help with any Rohde & Schwarz product, contact our customer support center. A team of highly qualified engineers provides support and works with you to find a solution to your query on any aspect of the operation, programming or applications of Rohde & Schwarz products.

Contact information

Contact our customer support center at www.rohde-schwarz.com/support, or follow this QR code:

Figure 9-2: QR code to the Rohde & Schwarz support page

9.6 Reporting cybersecurity issues

Contact us immediately if you discover a cybersecurity issue in a Rohde & Schwarz product. We keep reports confidential and never reveal your identity without consent.

1. Prepare a report with the following information, if available:
   - Affected product with product name, serial number and software version
   - Detailed description of the issue including the steps to reproduce it

- Contact information, including reporter name, organization and preferred communication channel

2. Contact our customer support or send an email to:
   productsecurity@rohde-schwarz.com

   Our product security team will analyze your report and coordinate any necessary vulnerability disclosure. We will give you an estimated resolution timeframe and keep you informed of progress.

Glossary: Abbreviations and terms

A

AES: Advanced encryption standard

agent: An R&S CHM agent instance on Windows or Linux hosts that sends its monitoring results to an R&S CHM host. Read complete definition: Section 6.12, "Configuring distributed monitoring", on page 108

API: Application programming interface

B

bind user: The user that is necessary to access the user and group information at the LDAP server.

C

CA: Certificate authority

client: A client is a host that runs the R&S CHM client application. It is intended for running the web GUI with additional features.
See Section 4.3, "Installing R&S CHM clients", on page 23.

CPU: Central processing unit

CSR: Certificate signing request

CSS: Cascading style sheets

CTS: Clear to send

D

DES: Data encryption standard

DISA: Defense Information Systems Agency

DKN: Digitales Kommunikationsnetz (digital communications network)

DN: Distinguished name

DNS: Domain name server

DSR: Data set ready. A DSR signal change indicates that the power of the data communication equipment is off.

DTR: Data terminal ready

F

FIPS: Federal information processing standard. FIPS standards establish requirements, e.g. for ensuring computer security and interoperability.

FQDN: Fully qualified domain name

G

gb2pp: A Rohde & Schwarz proprietary network protocol

GPG: GNU privacy guard

gRPC: General-purpose remote procedure calls.

GUI: Graphical user interface

H

HA: High availability

HDD: Hard disk drive

HMAC: Hash-based message authentication code

host: A host is an independent device in the system, which is addressed and monitored by R&S CHM. A host is, e.g. a Windows PC or a Linux virtual machine, or a device that you monitor using SNMP.

HP iLO: Integrated Lights-Out interface from Hewlett-Packard for configuration, update and remote server operation

HTML5: Hypertext mark-up language, version 5

HTTP: Hypertext transfer protocol

HTTPS: Hypertext transfer protocol secure

HUMS: Rohde & Schwarz health and utilization monitoring system

I

ICMP: Internet control message protocol

iDRAC: Integrated Dell remote access controller

ISO image: A disc image that contains everything that would be written to an optical disc. The ISO image contains the binary image of the optical media file system.

J

JSON: JavaScript Object Notation

K

KDC: Key distribution center, it handles authentication, ticket granting and holds a database with all the principals. See also principal.

Kerberos: A computer network authentication protocol.

Kerberos ticket: A certificate that is issued by an authentication server and encrypted using the server key. There are two types of tickets, TGT and ST.

keytab: Short for "key table". A file that stores long-term keys for one or more principals. See also principal. Can be extracted from principal database on KDC server.

L

LAN: Local area network

LCD: Liquid crystal display

LCSM: Lifecycle software manager

LDAP: Lightweight directory access protocol

LXI: LAN extensions for instrumentation

M

MAC: Media access control

master: R&S CHM host instance that is located in the top-level subsystem. Read complete definition: Section 6.12, "Configuring distributed monitoring", on page 108

MD5: Message digest algorithm 5

MIB: Management information base. Collection of objects in a virtual database that allows network managers using Cisco IOS software to manage devices such as routers and switches in a network.

MSI: Microsoft Software Installer

N

NAS: Network attached storage

NAVICS: Navy integrated communication system

NSS: Name service switch. Provides a central configuration store where services can look up sources for various configuration and name resolution mechanisms.

NTP: Network time protocol

O

OID: Object identifier. An address that uniquely identifies managed devices and their statuses. The SNMP protocol uses OIDs to identify resources that can be queried, among other things.

P

package cache: The package cache folder is a system folder. By default, it is located on the drive where your operating system is installed. The folder is used by applications to store settings, caches, installers and packages.

PAE: Port access entity

PDF: Portable document format. Frequently used file format for saving and exchanging documents.

PDU: Power distribution unit

PEM: Privacy-enhanced mail; a container format that can include only a public certificate or an entire certificate chain, including public key, private key, and root certificates.

PFX: Personal information exchange

PKI: Public key infrastructure

principal: A kerberos principal is a unique identity to which kerberos can assign tickets.

R

RAID: Redundant array of independent disks.

RAM: Random access memory

RCCM: Remote control, configuration and monitoring

RPM: Red Hat Package Manager

S

satellite: R&S CHM host instance that is not placed in the top-level subsystem. Read complete definition: Section 6.12, "Configuring distributed monitoring", on page 108

SHA: Secure hash algorithm

SIP: Session initiation protocol

SNMP: Simple network management protocol. It allows devices to exchange monitoring and managing information between network devices.

SSD: Solid state drive

SSH: Secure shell

SSO: Single sign-on

SSSD: The system security services daemon is a system daemon.

ST: Service ticket. Obtained from the TGS.

subsystem: A subsystem is at least one R&S CHM node that is grouped with any number of non-R&S CHM hosts or devices. Each R&S CHM host instance in a subsystem provides its own web GUI.

swap partition: A dedicated section of the hard drive that acts as an extension of the physical RAM.

T

TCP: Transmission control protocol

TGS: Ticket granting server. A logical KDC component that is used by the Kerberos protocol as a trusted third party.

TGT: Ticket granting ticket. A user authentication token issued by the KDC that is used to request access tokens from the TGS for specific resources or systems that are joined to the domain.

TLS: Transport layer security

U

UPS: Uninterruptible power supply

URL: Uniform resource locator, i.e. a web address

USM: SNMPv3 user-based security model

UTC: Universal time coordinated

V

VM: Virtual machine

W

WAN: Wide area network

WCS: Wireless communications system

web GUI: Short for R&S CHM web GUI. The web GUI runs in a browser. It shows all information collected by R&S CHM. If you run the web GUI on a Windows client, you can take advantage of additional features.
See Section 4.3, "Installing R&S CHM clients", on page 23.

X

XML: Extensible markup language

Y

YAML: YAML™ ain't markup language

Glossary: Specifications

I

ISO 8601: Date and time

R

RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II

RFC 1628: UPS Management Information Base

RFC 3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks

RFC 5424: The Syslog Protocol

List of YAML keys

ar60....135
argus....135
authentication....64
authorization....68
automatic_system_control....122
automatic_system_control....123
bitdefender....136
builtin....65
check_kerberos_auth....136
checkgroups....125
chm_agent_conn....136
chm_remote_grpc....137
chm_remote, simcos3....137
cisco_hardware....142
cputemp....143
dashboards....48
dhcp....143
displayname....126
dkn....144
dns....146
domain....147
dummy....148
eta_pdu....148
exports....50
file_content....149
file_exists....150
forget_states_on_restart....73
fortinet....150
fortinet_wcs....151
functions....124
gb2pp....152
generic_printer....154
Graphical system view (maps)....86
graphs....72
gssapi....65
gude....155
health_host....126
hosts....45
hums....156
icinga2_cluster....156
idrac....156
ilo....158
interval....126
lancom_vpn_status....159
lancom_xs_gs_3000....160
ldap....66
load....160

logging.....57
logic.....52
logic_id.....126
mac_address.....162
manual.....162
maps.....126
meinberg.....163
mikrotik.....163
monitoring.....64
msr4.....164
navics.....164
nport.....167
ntp_time.....168
nw_interface.....169
os_disk.....171
os_memory.....171
os_process.....172
os_service.....172
passive.....172
ping.....173
raritan_pdu.....174
RCCM.....177
reporting.....74
reports.....75
snmp.....178
snmp_connection.....128
snmp_diskspace.....179
snmp_flex.....179
snmp_hostalive.....180
snmp_processes.....181
snmp_time.....181
spectracom_time.....182
ssh.....183
ssh_connection.....131
subsystems.....112
synology.....184
system_state.....184
tcp.....185
thresholds.....132
tmr_radio.....185
trustedfilter.....185
ups.....186
vmware.....187
webinterface_url.....60
webserver.....61
widgets.....49
windowsupdateage.....188
xcp_ng.....189