

**XINNOR**

**xiRAID Classic 4.3.0  
Administrator's Guide**

# Contents

<b>xiRAID Classic 4.3.0 Administrator's Guide.....</b>	<b>4</b>
Introduction.....	4
About xiRAID Classic 4.3.0.....	5
Using xicli.....	6
Command-Line Interface (CLI) Overview.....	7
Accepting EULA.....	8
License Management.....	10
RAIDs explained.....	13
RAID components.....	13
RAID levels.....	14
RAID lifecycle.....	17
RAID Management.....	18
Creating a RAID.....	18
RAID Initialization.....	25
Showing RAID State.....	26
Managing RAID configuration.....	33
Scanning for RAID Silent Data Corruption.....	47
Deleting a RAID.....	48
Drive Management.....	50
RAID Reconstruction.....	62
CPU Management.....	63
System Administration.....	64
Scanning RAIDs and Drives, LED Indication.....	64
Configuration Files and Metadata.....	67
Update Management.....	80
Notifications.....	81
Using RAID in HA cluster.....	88
General Configuration Recommendations.....	88
RAID Creation.....	88

Using NVMe-oF devices to create a RAID.....	89
RAID and System Setup Recommendations.....	90
File System Mounting Examples.....	98
DKMS Specifics when Updating Linux Kernel.....	100
Authenticating gRPC Client.....	102
Updating GPG Keys.....	104
Troubleshooting.....	105
Command Reference.....	106
Overview.....	106
Command Line Interface (CLI) Description.....	106
config.....	107
drive.....	109
license.....	112
log.....	113
mail.....	115
pool.....	117
raid.....	121
settings.....	143
sdc.....	155
update.....	157

# xiRAID Classic 4.3.0 Administrator's Guide

Instructions, practical guides, and tips for administering xiRAID Classic 4.3.0

## Introduction

### Intended Audience

This guide is intended for administrators and users of RAIDs based on the xiRAID Classic 4.3.0 software.


The guide contains instructions on how to configure and manage RAIDs in xiRAID Classic 4.3.0.


### Guide Conventions

The Guide uses the typefaces and formatting to specify different names and terms:

Convention	Uses
<b>Bold</b>	Documentation titles, section titles, GUI controls, option value, minor titles.
<i>Italic</i>	Emphasis, term references, file paths.
<b>Text color</b>	Instructions for specific situations and configurations, links.
Monospace	Commands, command utilities, and console-driven text.

Text paragraphs that need your special attention are marked with the following frame:

 Tip - a note, which provides valuable information.

 Warning - binding instructions to guarantee the proper work of the software.

## About xiRAID Classic 4.3.0

xiRAID Classic 4.3.0 is high-performance software RAID developed specifically for NVMe storage devices and new types of SAN networks. xiRAID Classic 4.3.0 technologies use high potential of Flash devices (NVMe, SAS, SATA) to create a fast fault-tolerant RAID available as a local block device with opportunity of export via network by using auxiliary software.

xiRAID Classic 4.3.0 is a Linux kernel module and a management utility, which is built and configured for the most popular distributions (see the xiRAID Classic 4.3.0 System Requirements document). The software is installed on servers with slots for Flash memory devices or with connected JBOFs. xiRAID Classic 4.3.0 enables you to combine drives into high-performance fault-tolerant RAID.

## xiRAID Classic 4.3.0 Specification

Supported RAID levels	<ul style="list-style-type: none"><li>• RAID 0</li><li>• RAID 1</li><li>• RAID 10</li><li>• RAID 5</li><li>• RAID 6</li><li>• RAID 7.3</li><li>• RAID 50</li><li>• RAID 60</li><li>• RAID 70</li><li>• RAID N+M</li></ul>
Maximum number of drives in a RAID	64.
Maximum number of drives in the system	Depends on hardware configuration.
Maximum number of RAIDs	128.
Maximum RAID size	Defined by drive sizes.
Space for RAID metadata storage	The system reserves the first 96 MiB and the last 96 MiB of each drive in a RAID.
Support for NVMe Multipathing	Native multipathing functionality with the round-robin policy is supported.

## Using xicli

Manage your software xiRAID Classic in Linux by using the `xicli` program.

Most of the commands listed in this document require superuser privileges. Please log in as an administrator or root to run these. However, the following commands can be run without superuser privileges: all commands with the `show` subcommand (raid show, config show, drive faulty-count show, settings eula show, license show etc) and any command with the `--help` parameter.

## Command-Line Interface (CLI) Overview

### Conventions on CLI command syntax

Item format	Description
item	A required item (command, subcommand, argument, option).
<item>	A placeholder variable.
[item]	An optional item.

In the CLI, enter commands in the following format:

```
# xicli <command> <subcommand> <required_args> [optional_args]
```

To show the full list of commands, run

```
# xicli -h
```

To show the `xicli` version and the RAID driver version, run

```
# xicli -v
```

CLI syntax specifics:

1. Type the arguments of the subcommands in one line.
2. Subcommand arguments are separated by spaces.
3. Use short or long forms of subcommand argument options.
4. To get the list of all subcommands and arguments, add the `-h` option:

```
# xicli <command> <subcommand> -h
```

## The list of available commands <command>

config	Operations with the configuration file.
drive	Operations with the drives.
license	Operations with the license.
log	Operations with the event log.
mail	Operations with the mail notifications.
pool	Operations with the spare pools.
raid	Operations with the RAIDs.
settings	Operations with the additional settings of the <code>xicli</code> program.
update	Operations with the Update Check service.

A detailed description of the commands and subcommands is presented in the corresponding sections of the document.

## Accepting EULA

The first time you run any `xicli` command after installation (except for 'settings eula modify', 'settings eula show', and any command with the '--help' parameter), you will be prompted to accept the EULA.

After accepting the EULA, the ran command executes, and you can use xiRAID Classic 4.3.0.



# License Management

You can manage the license with the command

```
# xicli license <subcommand>
```

Subcommands for the `license` command:

<code>delete</code>	Delete the current license.
---------------------	-----------------------------

---

<code>show</code>	Show info on the current license.
-------------------	-----------------------------------

---

<code>update</code>	Update the current license.
---------------------	-----------------------------

To start working with the system, add the valid license file on each node. To do so, you need the hardware key (`hwkey`) which can be found by running the command:

```
# xicli license show
```



To request a trial license, please do not hesitate to contact us at [support@xinnor.io](mailto:support@xinnor.io).

Command output example when no license was added:

```
Kernel version: 5.15.0-131-generic  
hwkey: 96F84E31B68254EB  
license_key: null  
version: 0  
crypto_version: 0  
created: 0-0-0  
expired: 0-0-0  
disks: 0  
levels: 0  
type: nvme  
disks_in_use: 0  
status: expired
```

Command output example when a license was added:

```
Kernel version: 5.15.0-131-generic  
hwkey: 96F84E31B68254EB  
license_key: 4C24F88BBD70A127121A3442884FF33D  
5147A1BE70E97C4755C586A162AAA6DFD78F27CD7382C  
7F5155090B7BCB96C3C0986DD25FBC78DE9ABE2ED96E9  
E1E5FF12BE359A2159BB133DDE43BC5E9D13F06031B9C  
836E168951D7DFE48DC21ECEE6917CD5E15F0B941F9C7  
86E7BD2B0FA174631932DA27DB39D615B15B406A8EEF  
version: 1  
crypto_version: 1  
created: 2025-2-19  
expired: 2025-3-19  
disks: 6  
levels: 5  
type: nvme  
disks_in_use: 4  
status: valid
```

### Description of the `license` command output

Kernel version	Kernel version.
hwkey	Hardware key.

## Description of the `license` command output (continued)

<code>license_key</code>	License key.
<code>version</code>	License version.
<code>crypto_version</code>	Version of crypto-API for the license generator.
<code>created</code>	The date when the license was created.
<code>expired</code>	License expiration date.
<code>disks</code>	Maximum number of drives.
<code>levels</code>	Maximum RAID level. RAID levels from minimal to maximal: 0, 1, 10, 5, 6, 7 (stands for 7.3), 50, 60, 70 (includes N+M).
<code>type</code>	Drive type.
<code>disks in use</code>	Number of used drives in the system.
<code>status</code>	License state.

You can save the command output as a text file by running the command:

```
# xicli license show > license_request.txt
```

To get your license key, send your hardware key to the Xinnor support team at [support@xinnor.io](mailto:support@xinnor.io).

After you get your license file, copy it to the system, and apply the license key by running the command below.

```
# xicli license update -p license.txt
```

To check the applied license, run:

```
# xicli license show
```

## RAIDs explained

RAID (Redundant Array of Independent Disks) is a way of combining storage devices to ensure data integrity and high performance. There are several methods of combining hard drives called RAID levels. Each level has its pros and cons and offers a different balance of performance, data protection, and storage efficiency.

In this chapter, you will learn about the components, levels and operation details of the xiRAID Classic RAIDs.

## RAID components

RAID is a combination of multiple disks, with *striping*, *mirroring*, and *parity* forming the three basics of RAID levels:

- **Mirroring** - replication of data across disks (RAID 1).
- **Striping** - the process of dividing data into blocks and distributing these blocks across multiple storage devices in a RAID (RAID 0, 5, 6, 7, N+M).
- **RAID parity** is the additional data calculated based on the user data stored on the drives in RAID 5, 6, 7, or within one RAID group in a nested RAID configuration. This parity data allows for the restoration of user data from one or more failed drives in a RAID array in the event of a failure.

**RAID strip size.** In RAID striping, data is divided into strips. When data is written to a RAID level with striping, it is broken down into pieces (strips), and each piece is written to a single disk in the array (*the set of strips spanning across all the drives in that RAID set is called a stripe*). The size of the strip determines the size of the data piece. In xiRAID, the **strip size** can be selected from 16, 32, 64, 128, and 256 Kbytes. By default, the strip size is determined as follows:

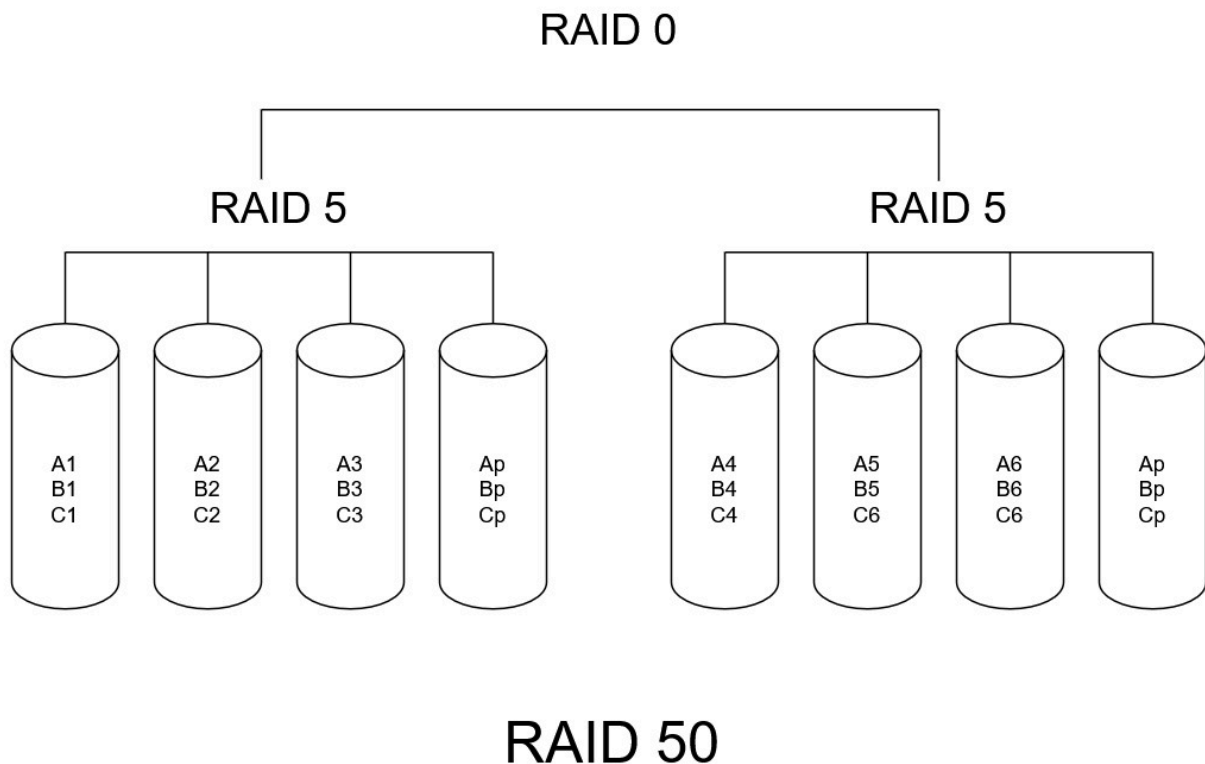
- For RAID levels 0, 1, 5, 6, 7.3, and N+M, use the largest strip size (not exceeding 128K) to ensure the total RAID stripe size does not exceed 1MB.
- For RAID levels 10, 50, 60, and 70, use the largest strip size (not exceeding 128K) to ensure the sub-RAID (group) stripe size does not exceed 1MB.

**RAID block size** - the size of the stripes written to each disk in a RAID array. The RAID block size can be either 512 or 4096 bytes, with the default being 4096.

**Nested RAID** – RAID 10, 50, 60 and 70 (RAID 0 combination of several RAID 1, 5, 6 or 7).

**RAID group size** – the number of disks in one RAID group. A RAID group is the segment of the RAID 0 striping.

The picture illustrates a *nested* RAID 50 configuration where data is *striped* across 6 drives organized into 2 *groups* of 3. Slots A1, A2, Ap, etc. represent *strips*, and data is written to and read from the RAID in *stripes (blocks)* [A1, A2, Ap, A3, A4, Ap]. Strips Ap, Bp, etc. represent *parity*.



## RAID levels

In xiRAID, you can create RAID 0, 1, 5, 6, 7.3, 10, 50, 60, 70 and N+M.

<b>RAID level</b>	<b>Description</b>	<b>Features</b>	<b>Redundancy</b>	<b>Requirements</b>
0	Disk striping without mirroring or parity. The data blocks are distributed across several drives.	Data is in parallel access mode that provides high performance.	Due to the lack of redundancy, RAID 0 doesn't provide data reliability – the failure of one drive in RAID leads to the whole RAID degradation.	RAID 0 requires at least 1 drives.
1	Mirroring without parity or striping. The data is mirrored on all drives of the RAID.	Random read performance can match the combined speed of individual drives, but write performance is limited by the slowest drive.	This level offers the highest redundancy by creating a 1-for-1 copy of all data.	RAID 1 requires at least 2 drives.
5	Disk striping with distributed parity.		Sustains the complete failure of one drive.	RAID 5 requires at least 4 drives.
6	Disk striping with double parity distribution.	Two checksums are calculated, the capacity of two drives is allocated for checksums.	Sustains the complete failure of two drives.	RAID 6 requires at least 4 drives.
7.3	Disk striping with triple parity distribution.	Three checksums are calculated using different algorithms,	Sustains the complete failure of three drives.	RAID 7 requires at least 6 drives.

(continued)

RAID level	Description	Features	Redundancy	Requirements
		the capacity of three drives is allocated for checksums.		
10	RAID 0, which components are RAID 1 instead of separate drives.		Data integrity is preserved when one drive fails, but if both drives in a mirrored pair fail, irreversible RAID destruction can happen.	RAID 10 requires at least 2 drives (the number of drives must be even).
50	RAID 0 striping combination across multiple RAID 5.		Recoverable from 1 drive failure in each group.	At least 8 drives, the drive number must be a multiple of the the group size (at least 2 groups are required). The group size is at least 4 drives.
60	RAID 0 striping combination across multiple RAID 6.	RAID 60 is the equivalent of RAID 50 with a higher level of fault tolerance.	Recoverable from 2 failures in each group.	At least 8 drives, the drive number must be a multiple of the the group size (at least 2 groups are required). The group size is at least 4 drives.

(continued)

RAID level	Description	Features	Redundancy	Requirements
70	RAID 0 striping combination across multiple RAID5 level 7.3.	RAID 70 is the equivalent of RAID 60 with a higher level of fault tolerance.	Recoverable from 3 failures in each group.	At least 12 drives, the drive number must be a multiple of the the group size (at least 2 groups are required). The group size is at least 6 drives.
N+M	The level of interleaving blocks with N drives and M checksums.	RAID N+M allows users to select the number of drives allocated for checksums.  The number of checksums can range from 4 to 32.	Recoverable from up to 32 disk failures in each group.	At least 8 drives. At least 4 drives must be allocated for checksums.  Additional conditions: $N+M \leq 64$ and $M \leq N$ .

## RAID lifecycle

After creating the RAID, it starts initializing - setting up the drives, calculating parity, and writing metadata to the drives. Once this process concludes, the RAID transitions to a fully operational 'initialized' state. However, RAID 0 doesn't require initialization. After creation, it's immediately ready for work, transitioning to the 'online' state.

If one or more drives fail, the RAID automatically switches to a "*degraded*" state. To restore the RAID, the failed drive needs to be replaced.

After replacing the drive, the RAID system starts the process of reconstructing data on the new drive, and the RAID goes into a "*reconstructing*" state. During this process, performance and redundancy remain degraded until it is completed.

A RAID can be destroyed without deleting metadata and configuration files. If this happens, it displays a 'None' state and can be restored.

## RAID Management

In this chapter you will learn how to create and delete xiRAID Classic RAID objects.

### Creating a RAID

You can create the RAID with the command

```
# xicli raid create <args> [optional_args]
```

For the argument descriptions, see the table below. For recommendations on configuring RAID settings, see the [General Configuration Recommendations](#) chapter.



Creating xiRAID Classic RAID over xiRAID Classic RAID devices is not allowed. To pool a large number of drives into a single address area, use RAIDs 10, 50, 60, 70.

Minimum number of drives required to create a RAID:


- of levels 5, 6 – at least 4 drives;
- of level 7.3 - at least 6 drives;
- of level 10 – at least 4 drives (the number of drives must be even);
- of level 0 – at least 1 drive;
- of level 1 – at least 2 drives;
- of levels 50, 60 – at least 8 drives (make sure the total drives number is multiple of the --group-size parameter value);
- of level 70 - at least 12 drives (make sure the total drives number is multiple of the --group-size parameter value);
- of level N+M – at least 8 drives.

Creating xiRAID RAIDs requires at least 1024 MiB of RAM.

## Arguments for the `create` subcommand

### Required arguments

---

<code>-n</code>	<code>--name</code>	<p>The name of the RAID.</p> <p>The maximum RAID name length is <b>28</b> characters. The RAID name may contain Latin letters, numbers, and underscores (<code>_</code>). Additionally, the following names are not permitted: <b>power</b> and <b>uevent</b>.</p> <p>To avoid naming conflicts between RAIDs and their partitions, avoid names that could overlap with partition identifiers. For example, do not create a RAID named <code>test1</code> if a RAID named <code>test</code> already exists, as partitions of <code>/dev/xi_test</code> may generate identifiers such as <code>/dev/xi_test1</code>, leading to conflicts.</p>
<code>-l</code>	<code>--level</code>	<p>The level of the RAID: <b>0</b>, <b>1</b>, <b>5</b>, <b>6</b>, <b>7</b>, <b>10</b>, <b>50</b>, <b>60</b>, <b>70</b>, or <b>nm</b>.</p> <div data-bbox="722 1167 1433 1301" style="background-color: #e6f2e6; padding: 10px;"><p> Use the value <b>7</b> to create RAID 7.3.</p></div>
<code>-d</code>	<code>--drives</code>	<p>The list of block devices (<code>/dev/nvme*</code>) separated by spaces.</p> <p>The order in which drives are specified during RAID creation directly influences how the drives are distributed across RAID groups. Drives are assigned to RAID groups in the exact sequence they are listed in the RAID creation command. For instance, in the case of a RAID 10 configuration, if drives are listed as <code>-d /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1</code>, the first two drives, <code>/dev/nvme0n1</code> and <code>/dev/nvme1n1</code>, will be placed in the first RAID group, while the next two drives, <code>/dev/nvme2n1</code></p>

## Arguments for the `create` subcommand (continued)

and `/dev/nvme3n1`, will be allocated to the second RAID group.

---

`-gs`      `--group_size`

**Only for RAIDs 10, 50, 60, or 70.**

The number of drives for one RAID group of level 5, 6, or 7.3 of the appropriate RAID 50, 60, or 70.

Possible values are integers from **4** to **32**.

---

`-sc`      `--synd_cnt`

**Only for RAIDs N+M.**

The number of syndromes M.

Possible values are integers from **4** to **32**.

Additional conditions:  $N+M \leq 64$  and  $M \leq N$ .

---

### Optional arguments

---

`-am`      `--adaptive_merge`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Adaptive Merge write function.

---

`--single_run`

**Except RAIDs 0, 1, 10.**

Use this parameter to adjust **the Adaptive Merge values** once at startup. After that, the values are set and do not change at system reboot. The Adaptive Merge write function is then turned off.

Does not take any value.

**Can only be used with the `--adaptive_merge` parameter.**

---

`-bs`      `--block_size`

RAID block size: **512** or **4096** bytes.

## Arguments for the `create` subcommand (continued)

The default: **4096**.

---

`-ca`      `--cpu_allowed`

Specify the CPUs on which the RAID will be allowed to run.

Possible values: a comma-separated list of CPUs, a range of CPUs indicated by a hyphen, or the value 'all' (the RAID will run on all available CPUs).

The default: `all`.

---

`-inp`      `--init_prio`

**Except RAID 0.**

Initialization priority in %.

Possible values are from **1** to **100** (maximum rate of initialization).

The default: **50**.

---

`-mwe`      `--merge_write_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for write operations.

The default: **0**.

---

`-mre`      `--merge_read_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for read operations.

The default: **0**.

---

`-mrm`      `--merge_read_max`

**Except RAIDs 0, 1, 10.**

Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for read requests.

## Arguments for the `create` subcommand (continued)

Possible values: integers from **1** to **100000**.

The default: **1000**.

---

`-mrw`      `--merge_read_wait`

**Except RAIDs 0, 1, 10.**

Wait time (in microseconds) between read requests with the Merge function enabled.

Possible values: integers from **1** to **100000**.

The default: **300**.

---

`-mwm`      `--merge_write_max`

**Except RAIDs 0, 1, 10.**

Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for write requests.

Possible values: integers from **1** to **100000**.

The default: **1000**.

---

`-mww`      `--merge_write_wait`

**Except RAIDs 0, 1, 10.**

Wait time (in microseconds) between write requests with the Merge function enabled.

Possible values: integers from **1** to **100000**.

The default: **300**.

---

`-ml`        `--memory_limit`

RAM usage limit in MiB.

Possible values: **0** and integers from **1024** to **1048576**.

The **0** value sets unlimited RAM usage.

The default: **0**.

---

## Arguments for the `create` subcommand (continued)

`-mp`      `--memory_prealloc`      The amount of reserved memory to allocate in MiB. This amount cannot exceed the RAM usage limit (`--memory_limit`). Additionally, it must be ensured that the specified amount does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free.

Possible values: **0** and integers from **1024** to **65536**.

A value of **0** indicates that reserved memory will not be allocated.

The default: **2048** (as long as it does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free).

---

`-rcp`      `--recon_prio`      **Except RAID 0.**

Reconstruction priority in %.

Possible values are from **1** to **100** (maximum rate of reconstruction).

The default: **50**.

---

`-rl`      `--request_limit`      Number of simultaneous I/O requests on RAID.

Possible values: from **0** (unlimited) to **4294967295**.

The **0** value disables the restriction.

The default: **0**.

---

`-rsp`      `--restripe_prio`      Restriping priority in %.

Possible values are from **1** to **100** (maximum rate of restriping).

## Arguments for the `create` subcommand (continued)

		The default: <b>100</b> .
<code>-sdcp</code>	<code>--sdc_prio</code>	SDC priority in %. The SDC priority cannot be set for RAIDs 0, 1, 10.  Possible values are from <b>1</b> to <b>100</b> (maximum rate of scanning).  The default: <b>50</b> .
<code>-se</code>	<code>--sched_enabled</code>	Enable ( <b>1</b> ) or disable ( <b>0</b> ) the scheduling function.  The default: <b>0</b> .
<code>-sp</code>	<code>--sparepool</code>	Name of the spare pool to assign to the RAID.
<code>-ss</code>	<code>--strip_size</code>	Strip size in KiB.  Possible values: <b>16</b> , <b>32</b> , <b>64</b> , <b>128</b> , or <b>256</b> .  The default: <b>16</b> .
<code>--force_metadata</code>		Allow overwriting of metadata on disks during the RAID creation process. Use this option only if you no longer need the data on the disks, as recovering data from a RAID without metadata can be extremely challenging.
<code>--trim</code>		This option runs the TRIM procedure on all disks in the RAID. However, the TRIM procedure will automatically run on RAID drives even without this option if the following conditions are met: <ul style="list-style-type: none"><li>• All disks support TRIM.</li><li>• None of the disks contain any metadata. This check ensures that no data is accidentally deleted. If a disk contains metadata, performing the TRIM procedure will</li></ul>

## Arguments for the `create` subcommand (continued)

make it impossible to restore the data from that disk. This condition helps prevent unintentional data loss.

- The `--no_trim` option is not specified.

---

`--no_trim`

Use this option to prevent the TRIM procedure from being performed on disks in the RAID.

Example: Create the RAID 5 named "media5" over 4 NVMe drives — "nvme0n1", "nvme1n1", "nvme2n1", "nvme3n1", with strip size equal to 64 KiB and enabled Merge function for write operations.

```
# xicli raid create -n media5 -l 5 -d /dev/nvme0n1 /dev/nvme1n1 /  
dev/nvme2n1 /dev/nvme3n1 -ss 64 -mwe 1
```

## RAID Initialization

Initialization will automatically begin after a RAID, excluding RAID 0, is created. Initialization is crucial to prevent data loss.



To improve the system performance under the load, try **decreasing initialization priority** by changing the corresponding RAID parameter.

After a RAID is created and initialized, its performance may differ from the sustained performance for a specific workload pattern (it could be higher or lower). In real-world conditions, when receiving the workload, the RAID performance will eventually reach the sustained level on its own, but this may take some time. If you need to run performance tests, we recommend preconditioning the RAID for the specific pattern before testing in order to measure the sustained performance. For example, to achieve sustained performance for a sequential pattern on large block sequential patterns (both read and write), we recommend sequentially rewriting the RAID with a block size equal to the RAID stripe width twice.

## raid init start

To start or continue the RAID initialization, run

```
# xicli raid init start -n <raid_name>
```

Example: Start initialization of the RAID "media5":

```
# xicli raid init start -n media5
```

## raid init stop

To stop the RAID initialization, run

```
# xicli raid init stop -n <raid_name>
```

Example: Stop the RAID initialization process for RAID "media5":

```
# xicli raid init stop -n media5
```

## raid init reset

To start RAID reinitialization after an unsafe system shutdown, thereby protecting syndromic RAIDs (RAID levels 5, 50, 6, 60, 7, 70) from data loss caused by a write hole, run the following command:

```
# xicli raid init reset -n <raid_name>
```

Example: Start reinitialization of the RAID "media5":

```
# xicli raid init reset -n media5
```

## Showing RAID State

You can view info about the RAID with the command

```
# xicli raid show [optional_args]
```

### Arguments for the `show` subcommand

Optional arguments

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## Arguments for the `show` subcommand (continued)

Without the argument, show info on all xiRAID Classic RAIDs.

---

`-o`      `--online`

Only show RAIDs that are in the “online” state (RAIDs that were not unloaded by the `raid unload` command and are not offline).

The argument takes no value.

---

`-u`      `--units`

Dimension:

- **s** – sectors (1 sector=512 bytes);
- **k** – kilobytes;
- **m** – megabytes;
- **g** – gigabytes.

The default: **g**.

---

`-f`      `--format`

Output format:

- **table**;
- **json**;
- **prettyjson** – human-readable json.

The default: **table**.

---

`-e`      `--extended`

Show extended output.

The argument takes no value.

Example: Show information on the RAIDs:

```
# xicli raid show -e
```

RAID name	static	state	devices	health	wear	serials	params	info
test5	size: 29 GiB level: 5 strip_size: 128 block_size: 4096 sparepool: - active: True config: True	online initing	0 /dev/sdd online 1 /dev/sde online 2 /dev/sdf online 3 /dev/sdg online	100% 100% 100% 100%	N/A N/A N/A N/A	drive-scsi3 drive-scsi4 drive-scsi5 drive-scsi6	adaptive_merge : 0 cpu_allowed : 0-3 init_prio : 50 memory_limit_mb : 0 memory_prealloc_mb : 2048 merge_read_enabled : 0 merge_read_max_usecs : 1000 merge_read_wait_usecs : 300 merge_write_enabled : 0 merge_write_max_usecs : 1000 merge_write_wait_usecs : 300 recon_prio : 50 request_limit : 0 restripe_prio : 100 sched_enabled : 0 sdc_prio : 50	init_progress : 4 memory_usage_mb : 2048

### Description of the show subcommand output

Row	Description
name	RAID name.
static	Static RAID parameters: <ul style="list-style-type: none"> <li>• size.</li> <li>• level.</li> <li>• synd_cnt – only for RAIDs N+M – number of syndromes.</li> <li>• block_size – RAID block size.</li> <li>• group_size – only for RAIDs 10, 50, 60, and 70 – size of the corresponding RAID group.</li> <li>• strip_size.</li> <li>• sparepool – name of the assigned spare pool.</li> <li>• active:                             <ul style="list-style-type: none"> <li>◦ True, if the RAID’s block device is in the system.</li> <li>◦ False, if the RAID was not loaded after reboot or if the RAID is unloaded.</li> </ul> </li> <li>• config:                             <ul style="list-style-type: none"> <li>◦ True, if the RAID is in the configuration file.</li> <li>◦ False, if the RAID is missing.</li> </ul> </li> </ul>
state	RAID state:

## Description of the `show` subcommand output (continued)

Row	Description
	<ul style="list-style-type: none"> <li>• online – the RAID is available and ready to work.</li> <li>• initialized – initialization is finished.</li> <li>• initing – the RAID is initializing.</li> <li>• inconsistent – an integrity error was detected during an SDC scan.</li> <li>• degraded – the RAID is available and ready for work but some drives are missing or failed.</li> <li>• reconstructing – the RAID is reconstructing.</li> <li>• offline – the RAID is unavailable (after the RAID was unloaded or its configuration file restored), or if the number of available drives in the RAID is insufficient for its operation.</li> <li>• need_recon – the RAID needs reconstruction.</li> <li>• need_init – the RAID needs initialization.</li> <li>• read_only – the license has expired. The RAID is read-only.</li> <li>• unrecovered – RAID can't complete reconstruction because of unrecoverable sections.</li> <li>• none – RAID was unloaded via the <code>unload</code> command or was not restored after reboot.</li> <li>• restriping – RAID is restriping.</li> <li>• sdc_scanning – an SDC scan is in progress.</li> <li>• need_resize – restriping was finished, the RAID size increase is available.</li> <li>• need_restripe – restriping was stopped and not finished.</li> </ul>
devices	The list of devices included in the RAID, and their current states:

## Description of the `show` subcommand output (continued)

Row	Description
	<ul style="list-style-type: none"> <li>• online – the drive is active.</li> <li>• offline – the drive is missing or unavailable.</li> <li>• reconstructing – the drive is in process of reconstruction.</li> <li>• need_recon – the drive needs reconstruction.</li> </ul> <p>For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.</p> <p>If the RAID is in 'restripping' or 'need_resize' state, the devices involved in the resize operation are highlighted in yellow.</p>
health	<p>To show, use the command with the <code>-e</code> parameter.</p> <p>Percent of valid drive data.</p> <p>When health is 100% – no reconstruction required.</p> <p>For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.</p>
wear	<p>To show, use the command with the <code>-e</code> parameter.</p> <p>The wear percentage of the SSD or NVMe drive.</p> <p>When the drive reaches the 90% threshold, the system sends an error message to the mail.</p> <p>The S.M.A.R.T. values "Percentage used endurance indicator" and "Percentage Used" are used to check SSD and NVMe drives respectively.</p> <p>For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.</p> <p>By default, the wear value of drives is checked by the system every 24 hours. The output of the <code>xicli raid show -e</code> command returns the value from the last check, which</p>

## Description of the `show` subcommand output (continued)

Row	Description
	<p>occurred sometime within the past 24 hours. The 24-hour interval is controlled by the <code>smart_polling_interval</code> parameter.</p> <p>If drives in a RAID are under consistently high load (e.g., the RAID is completely rewritten multiple times a day), the wear on the drives might become significant. In such cases, reducing the interval between wear checks may be necessary to get more accurate information. This depends on how important the accuracy of wear values is for your setup.</p> <p>To view the value of this parameter, run the <code>xicli settings scanner show</code> command.</p> <p>To update the default 24-hour interval, use the <code>xicli settings scanner modify</code> command.</p>
serials	<p>To show, use the command with the <code>-e</code> parameter.</p> <p>Serial numbers of drives in RAID.</p> <p>For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.</p>
params	<p>To show, use the command with the <code>-e</code> parameter.</p> <p>Editable RAID parameters:</p> <ul style="list-style-type: none"> <li>• <code>init_prio</code> – (except RAID 0) initialization priority: from 1% to 100%.</li> <li>• <code>memory_limit_mb</code> – the RAM usage limit in MiB.</li> <li>• <code>memory_prealloc_mb</code> – the amount of allocated reserved memory in MiB.</li> <li>• <code>memory_prealloc_conf</code> - if the actual size of allocated reserved memory (<code>memory_prealloc_mb</code>) differs from the value specified in the configuration file, the value specified in the configuration file is</li> </ul>

## Description of the `show` subcommand output (continued)

Row	Description
	<p>displayed here (in yellow). This discrepancy may occur when a RAID with allocated reserved memory is restored, but the reserved memory was not restored due to the lack of available system memory.</p> <ul style="list-style-type: none"><li>• <code>merge_read_enabled</code> – is the function Merge enabled (1) or disabled (0) for read operations.</li><li>• <code>merge_write_enabled</code> – is the function Merge enabled (1) or disabled (0) for write operations.</li><li>• <code>merge_read_wait_usecs</code> – waiting time between read requests when Merge is enabled.</li><li>• <code>merge_read_max_usecs</code> – maximum time to wait for read requests accumulation with Merge enabled.</li><li>• <code>merge_write_wait_usecs</code> – waiting time between write requests when Merge is enabled.</li><li>• <code>merge_write_max_usecs</code> – maximum time to wait for write requests accumulation with Merge enabled.</li><li>• <code>recon_prio</code> – (except RAID 0) reconstruction priority: from 1% to 100%.</li><li>• <code>sched_enabled</code> – is the function Scheduling enabled (1) or disabled (0).</li><li>• <code>sdc_prio</code> - priority for SDC scans: from 1% to 100%.</li><li>• <code>request_limit</code> – number of simultaneous I/O requests on RAID (0 for no limit).</li><li>• <code>restripe_prio</code> – priority for restriping: from 1% to 100%.</li><li>• <code>cpu_allowed</code> - the CPUs on which the RAID is allowed to run.</li><li>• <code>cpu_allowed_configured</code> - the CPUs on which the RAID is allowed to run as specified in the configuration file. This parameter appears when the <code>cpu_allowed</code> parameter exceeds the number of available</li></ul>

## Description of the `show` subcommand output (continued)

Row	Description
	<p data-bbox="643 271 1434 353">CPUs in the system. Only available in High Availability configuration.</p> <ul data-bbox="624 376 1377 459" style="list-style-type: none"> <li data-bbox="624 376 1377 459">• <code>adaptive_merge</code> - is the function Adaptive Merge enabled (True) or disabled (False).</li> </ul>
info	<p data-bbox="560 577 895 609">Dynamic RAID values:</p> <ul data-bbox="624 667 1434 1408" style="list-style-type: none"> <li data-bbox="624 667 1434 750">• <code>init_progress</code> - initialization progress: from 0% to 100%.</li> <li data-bbox="624 772 1434 855">• <code>recon_progress</code> - reconstruction progress: from 0% to 100%.</li> <li data-bbox="624 878 1434 960">• <code>restripe_progress</code> - restriping progress: from 0% to 100%.</li> <li data-bbox="624 983 1434 1066">• <code>sdsc_progress</code> - SDC scan progress: from 0% to 100%.</li> <li data-bbox="624 1088 1434 1408">• <code>memory_usage_mb</code> - Displays the actual memory consumed by the specific RAID on the system. This includes both the amount of reserved memory (allocated but not configured) and the volume of memory buffers allocated outside of memory pools. This value is shown only when the <code>--extended</code> flag is used with the <code>show</code> command.</li> </ul>

## Managing RAID configuration

In this section you will learn how to configure and manage xiRAID Classic RAID objects.

## Increasing Size and Changing Level of RAID

In this chapter, you will learn about the following RAID operations:

- Changing the RAID level with the addition of new disks.
- Increasing the size of a RAID by adding new disks.
- ~~Increasing the size of a RAID by replacing its disks with larger disks (*vertical scaling* RAID operation).~~



The operation of increasing the RAID level by replacing the disks with larger ones is not supported in xiRAID Classic 4.3.0. Please refer to the "xiRAID Classic 4.3.0 Known Issues" for more information.

RAID operations that add new disks to a RAID consist of two steps: *restriping* (`raid restripe` command), which selects the disks to be added and the RAID level, and *resizing* (`raid resize` subcommand), which applies changes to the RAID (which is in *need\_resize* status).

## Requirements and Specifics of Restripping and Resizing

### Requirements and specifics:

- Except RAID N+M.
- Only one RAID can be restriped at a time.
- To improve the performance of your system under workload, try to **change the priority of restripping** by changing the corresponding RAID parameter.
- RAID state must not be one of the following:
  - offline;
  - need\_resize;
  - need\_restripe;
  - restripping;
  - degraded.



The RAID level can only be changed if the number of added drives is sufficient to fit all the data stored prior to the level change. The number of data drives in a group must not decrease. Note that for RAID levels 50, 60, and 70, the total number of drives must be a multiple of the group size. Please keep in mind that RAID 50 includes one syndrome drive per group, while RAID 60 has two and RAID 70 has three. These drives do not store information but are necessary for data recovery in case of loss.

Example: Adding 1 drive to a RAID 60 configuration with 14 disks and 2 groups is not feasible. The only possible configuration for 15 drives is 3 groups of 5 drives, with only 3 of them containing data. Performing this operation would result in a new RAID with fewer data drives, hence it cannot be done. However, it is possible to add 2 drives to create a new RAID configuration consisting of 2 groups, each containing 8 drives (2 syndrome and 6 data drives). This will increase the data storage space.

The available options for RAID level changes and the minimum required number of drives

Current level	New level	Requirements	Minimal number of drives you should add
RAID 0	RAID 0		1

**The available options for RAID level changes and the minimum required number of drives (continued)**

<b>Current level</b>	<b>New level</b>	<b>Requirements</b>	<b>Minimal number of drives you should add</b>
	RAID 1		1
	RAID 10	RAID 0 contains only 1 drive.	3
		RAID 0 contains more than 1 drive	The number of drives to be added must be equal to the number of drives in the RAID 0.
RAID 1	RAID 1		1
	RAID 10		2
	RAID 5		1
RAID 10	RAID 10		2
	RAID 5		1
	RAID 50	The number of RAID 50 disks must be a multiple of its group size.	1
RAID 5	RAID 5		1
	RAID 6		1
	RAID 10	The number of RAID 10 disks must be an even number.	1

**The available options for RAID level changes and the minimum required number of drives (continued)**

<b>Current level</b>	<b>New level</b>	<b>Requirements</b>	<b>Minimal number of drives you should add</b>
		The number of information disks in a group must not decrease.	
	RAID 50	<p>The number of disks in RAID 50 must be at least 8.</p> <p>The number of disks in RAID 50 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1
RAID 6	RAID 6		1
	RAID 7.3		1
	RAID 60	<p>The number of disks in RAID 60 must be at least 8.</p> <p>The number of disks in RAID 60 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1
RAID 7.3	RAID 7.3		1
	RAID 70	The number of disks in RAID 70 must be at least 12.	1

**The available options for RAID level changes and the minimum required number of drives (continued)**

<b>Current level</b>	<b>New level</b>	<b>Requirements</b>	<b>Minimal number of drives you should add</b>
		<p>The number of disks in RAID 70 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	
RAID 50	RAID 50	<p>The number of disks in RAID 50 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1
	RAID 60	<p>The number of disks in RAID 60 must be at least 8.</p> <p>The number of disks in RAID 60 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1
RAID 60	RAID 60	<p>The number of disks in RAID 60 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1

## The available options for RAID level changes and the minimum required number of drives (continued)

Current level	New level	Requirements	Minimal number of drives you should add
	RAID 70	<p>The number of disks in RAID 70 must be at least 12.</p> <p>The number of disks in RAID 70 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1
RAID 70	RAID 70	<p>The number of disks in RAID 70 must be a multiple of its group size.</p> <p>The number of information disks in a group must not decrease.</p>	1

## raid restripe

The chapter describes the available subcommands for the restripe operation.

Restripping refers to any change in RAID configuration, such as the position of checksums or data drives, with the aim of changing the RAID level or size.

To start RAID restripping, run

```
# xicli raid restripe start <args>
```

### Arguments for the `restripe start` subcommand

Required arguments

---

**Arguments for the `restripe start` subcommand (continued)**

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-l</code>	<code>--level</code>	The new level for the RAID.  If you are only increasing the RAID size, enter the current RAID level for this argument.
<code>-gs</code>	<code>--group_size</code>	<b>Only for RAIDs 50, 60, and 70.</b>  The new group size for the RAID.  Possible values: integers from <b>4</b> to <b>32</b> .
<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to add to the RAID.

To pause RAID restriping, run

```
# xicli raid restripe stop <arg>
```

**Argument for the `restripe stop` subcommand**

Required argument

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

To continue RAID restriping, run

```
# xicli raid restripe continue <arg>
```

**Argument for the `restripe continue` subcommand**

Required argument

## Argument for the `restripe continue` subcommand (continued)

`-n`                    `--name`                    The name of the RAID.

## raid resize

The chapter describes the command for the resizing operation.

Following the restriping process, the RAID goes into the 'need\_resize' state. We recommend starting the resizing operation in order to restore optimal system speed.



The `raid resize` command should be executed when there is no workload on the RAID.

```
# xicli raid resize <arg>
```

## Argument for the `resize` subcommand

Required argument

---

`-n`                    `--name`                    The name of the RAID.

## Examples of Restriping and Resizing



The operation of increasing the RAID level by replacing the disks with larger ones is not supported in xiRAID Classic 4.3.0. Please refer to the "xiRAID Classic 4.3.0 Known Issues" for more information.

Example: Restriping of the "media5" RAID with adding new drives `/dev/sdf` `/dev/sdg` `/dev/sdh` and RAID level changing from 5 to 6:

```
# xicli raid restripe start -n media5 -l 6 -d /dev/sdf /dev/sdg /dev/sdh
```

Example: Restripe the RAID “media5” by adding a new drive `/dev/sdi` without changing the RAID level (increasing the RAID size):

```
# xicli raid restripe start -n media5 -l 5 -d /dev/sdi
```

After restriping is finished, the RAID state is `need_resize` until you run

```
# xicli raid resize -n <raid_name>
```

## Changing RAID Parameters

To improve the system performance under the workload, try decreasing initialization, reconstruction, or restriping priorities.

See recommendations on configuring RAID parameters in the chapter [RAID and System Setup Recommendations](#).

To change the RAID dynamic parameters, run:

```
# xicli raid modify <arg> [optional_args]
```

### Arguments for the `modify` subcommand

#### Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.  The maximum RAID name length is <b>28</b> characters.
-----------------	---------------------	--

---

#### Optional arguments

---

<code>-am</code>	<code>--adaptive_merge</code>	<b>Except RAIDs 0, 1, 10.</b>  Enable ( <b>1</b> ) or disable ( <b>0</b> ) the Adaptive Merge write function.
------------------	-------------------------------	---

---

	<code>--single_run</code>	<b>Except RAIDs 0, 1, 10.</b>  Use this parameter to adjust <b>the Adaptive Merge values</b> once at start-up. After that, the values are set and
--	---------------------------	---

**Arguments for the `modify` subcommand (continued)**

do not change at system reboot. The Adaptive Merge write function is then turned off.

Does not take any value.

Can only be used with the `adaptive_merge` parameter.

`-ca`                    `--cpu_allowed`

Change the CPUs on which the RAID will be allowed to run.

Possible values: a comma-separated list of CPUs, a range of CPUs indicated by a hyphen, or the value 'all' (the RAID will run on all available CPUs).

The default: `all`.

`-inp`                    `--init_prio`

**Except RAID 0.**

Initialization priority in %.

Possible values are from **0** to **100** (maximum rate of initialization).

The default: **50**.

`-mwe`                    `--merge_write_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for write operations.

The default: **0**.

`-mre`                    `--merge_read_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for read operations.

**Arguments for the `modify` subcommand (continued)**

The default: **0**.

---

<code>-mrm</code>	<code>--merge_read_max</code>	<b>Except RAIDs 0, 1, 10.</b>  Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for read requests.  Possible values: integers from <b>1</b> to <b>100000</b> .  The default: <b>1000</b> .
<code>-mrw</code>	<code>--merge_read_wait</code>	<b>Except RAIDs 0, 1, 10.</b>  Wait time (in microseconds) between read requests with the Merge function enabled.  Possible values: integers from <b>1</b> to <b>100000</b> .  The default: <b>300</b> .
<code>-mwm</code>	<code>--merge_write_max</code>	<b>Except RAIDs 0, 1, 10.</b>  Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for write requests.  Possible values: integers from <b>1</b> to <b>100000</b> .  The default: <b>1000</b> .
<code>-mww</code>	<code>--merge_write_wait</code>	<b>Except RAIDs 0, 1, 10.</b>

---

## Arguments for the `modify` subcommand (continued)

Wait time (in microseconds) between write requests with the Merge function enabled.

Possible values: integers from **1** to **100000**.

The default: **300**.

---

`-ml`

`--memory_limit`

RAM usage limit in MiB.

Possible values: **0** and integers from **1024** to **1048576**.

The **0** value sets unlimited RAM usage.

The default: **0** (unlimited).

---

`-mp`

`--memory_prealloc`

The amount of reserved memory to allocate in MiB. This amount cannot exceed the RAM usage limit (`--memory_limit`). Additionally, it must be ensured that the specified amount does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free. If there is not enough available memory on the system, xiRAID Classic will attempt to change the allocated amount to its previous size. If this operation fails, an error message will be displayed, and the allocated amount will be set to **0**.

Possible values: **0** and integers from **1024** to **65536**.

Setting the value to **0** removes reserved memory for the specified RAID.

**Arguments for the `modify` subcommand (continued)**

The default: **2048** (as long as it does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free).

---

<code>-rcp</code>	<code>--recon_prio</code>	<b>Except RAID 0.</b>  Reconstruction priority in %.  Possible values: from <b>1</b> to <b>100</b> (maximum rate of reconstruction).  The default: <b>50</b> .
<code>-rl</code>	<code>--request_limit</code>	Number of simultaneous I/O requests on RAID.  Possible values: integers from <b>0</b> to <b>4294967295</b> .  The <b>0</b> value disables the restriction.  The default: <b>0</b> .
<code>-rsp</code>	<code>--restripe_prio</code>	Restriping priority in %.  Possible values are from <b>0</b> to <b>100</b> (maximum rate of restriping).  The default: <b>100</b> .
<code>-sdcp</code>	<code>--sdc_prio</code>	SDC priority in %.  Possible values are from <b>1</b> to <b>100</b> (maximum rate of scanning).  The default: <b>50</b> .
<code>-se</code>	<code>--sched_enabled</code>	Enable ( <b>1</b> ) or disable ( <b>0</b> ) the scheduling function.

---

**Arguments for the `modify` subcommand (continued)**The default: `0`.

---

<code>-sp</code>	<code>--sparepool</code>	<p>Name of the spare pool to assign to the RAID.</p> <p>The <code>null</code> value removes the spare pool from the RAID.</p> <p>Spare pool can not be assigned to RAID 0.</p>
	<code>--force_online</code>	<p>Change RAID state to online if the RAID has unrecoverable sections.</p> <p>I/O operations on unrecoverable sections may lead to data corruption.</p> <p>The argument takes no value.</p>
	<code>--force_resync</code>	<p>This parameter is deprecated and will be removed in a future release. Use the <code>raid init reset</code> command instead.</p> <p><b>Except RAIDs 0, 1, 10.</b></p> <p>Force RAID re-initialization.</p> <p>The argument takes no value.</p>

---

Example: Setting reconstruction priority for the RAID "media5" equal to 50%:

```
# xicli raid modify -n media5 -rcp 50
```

**Scanning for RAID Silent Data Corruption**

Scanning a RAID for Silent Data Corruption (SDC) is recommended when you suspect or want to prevent data integrity issues that might be caused by undetected corruption in the storage array. Silent data corruption can occur without any obvious

signs or warnings, meaning the data is corrupted but the system or xiRAID Classic itself doesn't show errors, potentially leading to serious issues when the data is accessed later.

To start a RAID SDC scan, run the `sdc start` command:

```
xicli sdc start -n <RAID_name>
```

where `<RAID_name>` is the name of the RAID for which to start the scan. By default, the scanner identifies inconsistencies but does not resolve them. To fix inconsistencies during the scan, enable the `fixup` mode by using the `-m fixup` option:

```
xicli sdc start -n <RAID_name> -m fixup
```



Silent Data Corruption scanning is supported only for the following RAID levels: 5, 50, 6, 60, 7, and 70. Note that the `fixup` mode is not supported for RAID levels 5 and 50.

After the SDC scan process begins, the RAID switches to the `sdc_scanning` state. The progress of the scan is displayed as a percentage in the `sdc_progress` option. You can monitor the scan progress in the output of the `xicli raid show` command. To pause the scan, run the `sdc stop` command. Alternatively, if the scan appears to be stuck, you may stop the scan and reset the scan's progress by using the `sdc reset` command.


If an irreparable integrity error is detected during the scan, the RAID will enter the `offline` state, and the `inconsistent` flag will be applied. In this case, you may attempt to read available data on the RAID by manually switching the RAID to the `online` state. For more information, refer to `--force_online`.

## Deleting a RAID

The existing RAID can be either destroyed (removed from the system configuration without the possibility of restoration) or unloaded (removed from the system configuration while retaining its metadata and the possibility of restoration).

In this section you will learn how to destroy, unload and restore xiRAID RAIDs.

## Destroying the RAID

 Warning! The result of the command is irreversible. Read the description carefully.

You can delete the RAID without possibility to restore the RAID and data on it with the command

```
# xicli raid destroy <arg> [optional_args]
```

### Arguments for the `destroy` subcommand

Mutually exclusive required arguments

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-a</code>	<code>--all</code>	Delete all the xiRAID Classic RAIDs.  The argument takes no value.

Optional arguments (only applied with the `--all` argument)

	<code>--force</code>	Force the command execution.
	<code>--config_only</code>	Remove RAID only from config.

Example: Deleting the RAID "media5":

```
# xicli raid destroy -n media5
```



If you destroy an unloaded RAID, the metadata on the drives used in the RAID will not be erased. To erase the metadata, use the `xicli drive clean` command as described in [Removing Drive Metadata and Resetting Current Error Count](#).

## Unloading the RAID

You can remove (unload) the RAID with possibility to restore the RAID and save data on it with the command

```
# xicli raid unload <arg>
```

### Arguments for the `unload` subcommand

Mutually exclusive required arguments

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-a</code>	<code>--all</code>	Unload all available xiRAID Classic RAIDs.  The argument takes no value.

Example: Unloading the RAID "media5":

```
# xicli raid unload -n media5
```

After unloading the RAID, it goes into the 'offline' state.

To restore unloaded RAIDs, run:

```
# xicli raid restore {-n <raid_name>|-a}
```



To learn more about the RAID restoration process and its limitations, see [RAID restore](#).

## Drive Management

## Manual Drive Replacement or Excluding

You can exclude a drive from a RAID configuration, or replace it with another one, whether it has failed or is still functioning in the RAID. To do so, use the command:

```
# xicli raid replace <args>
```



If you manually replace a drive that is a part of a spare pool, the drive excludes from the spare pool.

### Arguments for the `replace` subcommand

#### Required arguments

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-no</code>	<code>--number</code>	The number of the drive.  To find out the number of the drive, use <pre># xicli raid show</pre>
<code>-d</code>	<code>--drive</code>	The new block device.  To remove the drive (to mark it as missing) set the <b>null</b> value.

Example: In the RAID "media5", replacing the drive "0" with the drive "nvme4n1":

```
# xicli raid replace -n media5 -no 0 -d /dev/nvme4n1
```

## Automatic Drive Replacement

A drive can be automatically replaced after it

- failed (was physically removed from a RAID);
- exceeded the threshold value of I/O errors (the default is 3. To change the threshold, use the command `xicli settings faulty-count modify --threshold`).

To automatically replace drives in a RAID, create a spare pool, then assign the created spare pool to the RAID. You can only assign one spare pool to each RAID array at a time. If a drive in the RAID fails, it can be replaced by a drive from the spare pool, provided the replacement drive is the same size or larger than the failed one.

If the system has a spare pool, you can assign it to an existing RAID or when creating a new RAID using the commands `xicli raid create/modify --sparepool`.

### Commands for managing spare pools

To create a spare pool, run

```
# xicli pool create <args>
```

#### Arguments for the `create` subcommand

##### Required arguments

---

<code>-n</code>	<code>--name</code>	The name for the spare pool.
<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.

To add drive(s) to a spare pool, run

```
# xicli pool add <args>
```



All drives in the spare pool must be of the same type; for example, you cannot mix HDDs and NVMe drives in the same spare pool.

## Arguments for the `add` subcommand

### Required arguments

<code>-n</code>	<code>--name</code>	The name of the spare pool.
<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.

To delete the spare pool, run

```
# xicli pool delete <arg>
```

## Argument for the `delete` subcommand

### Required argument

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

To remove drive(s) from the spare pool, run

```
# xicli pool remove <args>
```

## Arguments for the `remove` subcommand

### Required arguments

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

## Arguments for the `remove` subcommand (continued)

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.
-----------------	-----------------------	--

To show info on the spare pool, run

```
# xicli pool show [optional_args]
```

## Arguments for the `show` subcommand

### Optional arguments

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.  Without the argument, show info on all spare pools.
<code>-f</code>	<code>--format</code>	Output format: <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul> <p>The default: <b>table</b>.</p>
<code>-u</code>	<code>--units</code>	Size units:

---

### Arguments for the `show` subcommand (continued)

- **s** – sectors (1 sector=512 bytes);
- **k** – kilobytes;
- **m** – megabytes;
- **g** – gigabytes.

The default: **g**.

Spare Pools				
name	devices	serials	sizes	state
pool1	0 /dev/sdb	drive-scsi1	10 GiB	active
	1 /dev/sdc	drive-scsi2	10 GiB	

Possible drive states:

- ready – the drive is able for replacement;
- absent – drive is missing in the system;
- failed – attempt to replace with this drive from the spare pool failed, the drive will not be used for replacement.

Possible spare pool states:

- active - the spare pool is active;
- inactive - the spare pool is inactive, and drives from the pool cannot be used to replace failed drives in a RAID.

To manage delay timer for the drive replacement from the spare pools, run

```
# xicli settings pool modify <arg>
```



When you change any parameter of the `xicli settings pool modify` command, the `xiraid-scanner.service` restarts.



Drives that have exceeded the specified threshold value for I/O errors (`faulty-count`) are replaced immediately and without delay.

## Argument for the `pool modify` subcommand

### Required argument

---

<code>-rd</code>	<code>--replace_delay</code>	Delay time (in seconds) for the drive replacement from the spare pools.
		Only one delay time is used for all the spare pools.
		Possible values: integers from <b>1</b> to <b>3600</b> .
		The default: <b>180</b> .

To show delay time used for the drive replacement from the spare pools, run

```
# xicli settings pool show
```

## Argument for the `pool show` subcommand

### Optional argument

---

<code>-f</code>	<code>--format</code>	Output format:
		<ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul>
		The default: <b>table</b> .

Example: Creating a sparepool “pool1” and assigning it to the RAID “media5”:

## 1. Create a sparepool:

```
# xicli pool create -n pool1 -d /dev/sda /dev/sdb
```

## 2. Assign the created sparepool to the RAID:

```
# xicli raid modify -n media5 -sp pool1
```

Example: Setting the replacement timer for the sparepools to 60 seconds:

```
# xicli settings pool modify -rd 60
```

## Drive I/O Error Counter

You can keep track of drives where I/O errors (faults) have started to appear so that you can replace such drives with healthy ones in a timely manner.



We recommend setting up email notifications (to learn more, see the [Setting up Email Notifications](#)) chapter to trace drives with I/O errors.

*Fault threshold* is the common number of faults for each drive, above which the drive will be removed from the RAID (marked as 'missing') or replaced with a suitable drive from the spare pool. You can set the fault threshold value in the range from 1 to 1000 using the command `xicli settings faulty-count modify -t`. If you change the fault threshold value, the current number of faults on the drives is reset.

When a drive is removed from a RAID *because* the fault threshold is exceeded:

- if the RAID has a SparePool with the suitable drive, *the removed drive* will be replaced and then the RAID reconstruction will start;
- if *the removed drive* has not been replaced in the RAID (automatically or manually), the drive will return in the RAID after resetting the current number of faults on that drive;
- the `drive clean` command applied to *the removed drive* resets the current number of faults and does not remove metadata from the drive.

To manage the threshold value of I/O errors for all drives, run

```
# xicli settings faulty-count modify <arg>
```



When you change any parameter of the `xicli settings faulty-count modify` command, the `xiraid-scanner.service` restarts.

## Argument for the `faulty-count modify` subcommand

### Required argument

---

<code>-t</code>	<code>--threshold</code>	The threshold value for all drives.  If you set a new fault threshold value, the current numbers of faults are reset for all the drives.  Possible values: integers from <b>1</b> to <b>1000</b> .  The default: <b>3</b> .
-----------------	--------------------------	---

Example: Set the drive fault threshold value to 10:

```
# xicli settings faulty-count modify -t 10
```

To show the threshold value of I/O errors, run

```
# xicli settings faulty-count show
```

## Argument for the `faulty-count show` subcommand

### Optional argument

---

<code>-f</code>	<code>--format</code>	Output format:
-----------------	-----------------------	----------------

## Argument for the `faulty-count show` subcommand (continued)

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

To reset the current numbers of faults for drives, run

```
# xicli drive faulty-count reset <arg>
```



The RAID that contains the drive must be loaded.

When you change any parameter of the `xicli drive faulty-count reset` command, the `xiraid-scanner.service` restarts.

## Arguments for the `faulty-count reset` subcommand

### Required argument

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to reset their current numbers of faults.
-----------------	-----------------------	---

Example: reset current values of fault count for drives `/dev/sda`, `/dev/sdb`, `/dev/sdd`:

```
# xicli drive faulty-count reset -d /dev/sd[a-b] /dev/sdd
```

To show the current numbers of faults for drives, run

```
# xicli drive faulty-count show [optional_args]
```

## Arguments for the `faulty-count show` subcommand

### Mutually exclusive optional arguments

---

**Arguments for the `faulty-count show` subcommand (continued)**

<code>-n</code>	<code>--name</code>	<p>The RAID name for which drives the current number of faults will be shown.</p> <p>If neither of the two arguments is specified, show the values for all drives.</p>
<code>-d</code>	<code>--drives</code>	<p>The list of block devices (<code>/dev/sd*</code>, <code>/dev/mapper/mpath*</code>, <code>/dev/nvme*</code>, <code>/dev/dm-*</code>) separated by a space to show their current numbers of faults.</p> <p>If neither of the two arguments is specified, show the values for all drives.</p>

**Optional argument**


<code>-f</code>	<code>--format</code>	<p>Output format:</p> <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul>
-----------------	-----------------------	---

The default: **table**.

Example: show current values of fault count for drives `/dev/sda`, `/dev/sdb`, `/dev/sdd`:

```
# xicli drive faulty-count show -d /dev/sd[a-b] /dev/sdd
```

## Removing Drive Metadata and Resetting Current Error Count

 Warning! The result of the command is irreversible. Read the description carefully.

*Metadata* is xiRAID Classic device configuration information (to learn more, see [Configuration Files and Metadata](#)).

The `drive clean` command resets the current error counter value and/or removes metadata from selected disks depending on the status and state of those disks.

The `drive clean` command resets the current error counter and doesn't delete the metadata:

- on a disk included in a RAID that is present in the common configuration file.
- on a disk that was removed from a RAID due to exceeding the I/O error threshold.

In other cases, the `drive clean` command resets the current error counter and deletes the metadata.

To remove metadata from the drives and/or reset their current error count, run:

```
# xicli drive clean <arg>
```

### Argument for the `clean` subcommand

Required argument

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to reset the current fault counter and/or delete the metadata.
-----------------	-----------------------	--

Example: Deleting metadata from drives `"/dev/nvme5n1"` and `"/dev/nvme1n1"`:

```
# xicli drive clean -d /dev/nvme1n1 /dev/nvme5n1
```

## RAID Reconstruction

Reconstruction of a RAID (except RAID 0) starts automatically after a drive has been replaced in the RAID. While a RAID is being reconstructed, the functions initialization and restriping are paused.



To improve the system performance under the workload, try **decreasing reconstruction priority** by changing the corresponding RAID parameter.

To stop the RAID reconstruction, run

```
# xicli raid recon stop <arg>
```

### Argument for the `recon stop` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

To start the RAID reconstruction, run

```
# xicli raid recon start <arg>
```

### Argument for the `recon start` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

Example: Start RAID “media5” reconstruction:

```
# xicli raid recon start -n media5
```

## CPU Management

By default, xiraid has access to all CPUs in the system, but their usage can be limited. To limit the number of CPUs on which a RAID is allowed to run without restarting the module and stopping the services, use the commands "raid create" or "raid modify".

The limit for the CPU usage for specific RAIDs can be set during RAID creation and changed later using the commands "raid create" and "raid modify".

Specifics:

- possible to select specific CPUs for each RAID;
- do not require restarting the module.

To limit the number of CPUs during RAID creation, use the command:

```
# xicli raid create -n <name> -d <drives> -l <level> --cpu_allowed
<CPUs>
```

*Example:*

```
# xicli raid create -n media5 -l 5 -d /dev/nvme0n1 /dev/nvme1n1 /
dev/nvme2n1 /dev/nvme3n1 --cpu_allowed 0-21
```

To change the list of CPUs on which the RAID is allowed to run, use the command:

```
# xicli raid modify -n <name> --cpu_allowed <CPUs>
```

*Example:*

```
# xicli raid modify -n media 5 --cpu_allowed 1,3-4
```

-ca	--cpu_allowed	Specify the CPUs on which the RAID will be allowed to run.  Possible values: a list of CPUs separated by spaces, a range of CPUs indicated by a hyphen, or the value
-----	---------------	--

(continued)

		'all' (the RAID will run on all available CPUs).  The default: <code>a11</code> .
--	--	---

## System Administration

In this chapter you will learn how to configure the xiRAID Classic system.

### Scanning RAIDs and Drives, LED Indication

The *scanner* mechanism is used to monitor the statuses of RAIDs and drives:

- RAID statuses;
- RAID drives;
- automatic drive LED indication.

To manually control the LED indication, use the `drive locate` command (see description and examples below in this chapter).

#### The scanner command

The service, which runs in the system, automatically manages the indication of failed and working drives and creates corresponding messages in the log file.

To manage RAIDs monitoring, the LED indication and drive SMART settings, run

```
# xicli settings scanner modify <args>
```



When you change any parameter of the `xicli settings scanner modify` command, the `xiraid-scanner.service` restarts.

## Arguments for the `scanner modify` subcommand

At least one argument is required

---

<code>-spi</code>	<code>--scanner_polling_interval</code>	The polling interval for xiRAID Classic RAID's and drives in seconds.  The parameter affects the auto-start delay for the RAID initialization, reconstruction, and restriping.  Possible values: integers from <b>1</b> to <b>3600</b> (1 hour).  The default: <b>1</b> .
-------------------	---	---

---

<code>-spi</code>	<code>--smart_polling_interval</code>	S.M.A.R.T. drive health polling interval, in seconds.  Possible values: integers from <b>60</b> to <b>86400</b> (24 hours).  The default: <b>86400</b> .
-------------------	---------------------------------------	--

---

<code>-le</code>	<code>--led_enabled</code>	Enable ( <b>1</b> ) or disable ( <b>0</b> ) the automatic LED indication of drives in the system.  The default: <b>1</b> .  The argument doesn't affect manual LED indication.
------------------	----------------------------	--

Example: Disable automatic LED indication of drives:

```
# xicli settings scanner modify --led_enabled 0
```

To show the settings of the LED indication and drive scanner, run

```
# xicli settings scanner show
```

## Argument for the `scanner show` subcommand

### Optional argument

---

`-f`

`--format`

Output format:

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

## The `locate` command

You can manually control the LED indication of the drives by using the command

```
# xicli drive locate <arg>
```

## Argument for the `locate` subcommand

### Required argument

---

`-d`

`--drives`

The list of block devices (`/dev/sd*`, `/dev/mapper/mpath*`, `/dev/nvme*`, `/dev/dm-*`) separated by a space to switch the indication on, or switch the indication off (with the `null` value).

The argument doesn't affect the automatic indication.

Example: Turn location indication on for drives `"/dev/nvme0n1"` and `"/dev/nvme1n1"`:

```
# xicli drive locate -d /dev/nvme0n1 /dev/nvme1n1
```

## Configuration Files and Metadata

In this chapter, you can learn:

- how the information about the system is stored;
- how the information about xiRAID Classic RAID is stored;
- how to use this information to import a RAID to another system;
- how to manage the system and RAID configuration files;
- how to restore a RAID.

## xiRAID Classic System and RAID Configuration

Information about the xiRAID system and xiRAID RAID configurations is automatically stored in the following locations in the system:

- the common configuration file `/etc/xiraid/raid.conf` (xiRAID Classic system configuration);
- the configuration files of individual RAID in the `/etc/xiraid/raids` directory (xiRAID Classic RAID configurations);
- the metadata on the disks included in the created xiRAID Classic RAID device (xiRAID Classic RAID configurations);
- spare pool configuration files in `/etc/xiraid/pools`.

Additionally, before changing the common configurations file, the system automatically creates a backup file of the common configuration `/etc/xiraid/raid.conf.bak`. Similar backups are created for configurations in the `/etc/xiraid/raids/*` folder.

Thus, if necessary, you can restore the previous version of the device configuration.

### #ommon configuration file

The common configurations file stores the most recent changes to the xiRAID Classic system.

The example of the configuration file:

```
{
  "raid_autostart": 1,
  "pool_autoactivate": 1,
  "drives": {
    "drive-scsi8": "1",
    "drive-scsi7": "2"
  },
  "faulty_count_threshold": 3,
  "scanner_polling_interval": 1,
  "smart_polling_interval": 86400,
  "led_enabled": 1,
  "timestamp": 1648628802.796359
}
```

- `raid_autostart` indicates whether RAID autostart is enabled (1) or disabled (0) at the module loading. RAID autostart must be disabled in High Availability configurations.
- `pool_autoactivate` indicates whether spare pool automatic activation is enabled (1) or disabled (0). Spare pool automatic activation must be disabled in High Availability configurations.
- `drives` contains the serial numbers of the drives with an error count greater than 0 and their corresponding numbers of errors.
- `faulty_count_threshold` contains the value of the error threshold value for the drives.
- `scanner_polling_interval` contains the value of polling interval for xiRAID RAID's and drives in seconds.
- `smart_polling_interval` object contains the value of S.M.A.R.T. drive health polling interval in seconds. By default, the value is set to 24 hours.
- `led_enabled` indicates whether the automatic LED indication of drives in the system is enabled.
- `timestamp` contains the creation date of this configuration file in timestamp format.

## RAID configuration file

The RAID configuration files store the most recent changes to the xiRAID Classic RAID devices and are used when working with created xiRAID Classic RAID devices.

The example of a RAID configuration file:

```
{
  "name": "rvraid_1",
  "uuid": "CFF21DA4-A6A6-45DB-A27D-0E58CCF3BDAD",
  "level": "5",
  "synd_cnt": 1,
  "strip_size": 16,
  "block_size": 4096,
  "drives": [
    "drive-scsi2",
    "drive-scsi3",
    "drive-scsi4",
    "drive-scsi5"
  ],
  "size": 30302208,
  "group_size": 4,
  "auto_init": true,
  "auto_recon": true,
  "auto_restripe": true,
  "cpu_allowed": [],
  "timestamp": 1719439498.5411358,
  "memory_prealloc_mb": 1024,
  "memory_limit_mb": 0,
}
```

## Configuration metadata on disks

Since the common configurations file is stored on the system disk, to protect against system disk failure, the configuration information is also stored on the disks that belong to the xiRAID Classic RAID devices.

Each disk in a RAID contains data that enables a complete copy of this RAID configuration file to be restored. The system reserves the first 96 MiB and the last 96 MiB of each drive in a RAID.

## Spare pool configuration

Configuration files for spare pools are located in the */etc/xiraid/pools* directory. Below is an example of a spare pool configuration file:

```
{
  "name": "rvpool",
  "drive_type": "sd",
  "drives": {
    "drive-scsi11": 10493952
  },
  "timestamp": 1740050079.5879934,
  "host": "172.16.133.176:6066"
}
```

- `name` indicates the name of the spare pool.
- `drive_type` specifies the type of drive used in this spare pool:
  - `sd` represents SATA/SCSI drives,
  - `nvme` represents NVMe (Non-Volatile Memory Express) drives,
- `drives` contains the names of the individual drives in the pool and the number of sectors on the disks (a sector is typically 512 bytes).
- `timestamp` - The timestamp when this configuration file was created.
- `host` - In High Availability configurations, this field specifies the network address and port of the node where the disks in the spare pool are accessible.

## Importing a RAID

RAID Import is the transfer of RAID-containing disks from one system to another. Once imported, the xiRAID Classic RAID device becomes available for management and its configuration information is added to the RAID configuration file of the new system. Usually, configuration information from the metadata on the drives is used for the import.

However, when importing a RAID, there may be situations where there is already a RAID with the same name on the system where the RAID is migrated, or block devices with the same serial number are used.

Below in this chapter is a list of commands that enable you to import RAID and resolve possible conflicts, and an example of the import process.

The commands of the RAID import:

- `raid import apply`
- `raid import show`

The import commands work with RAIDs that are present in the disk metadata, but are not present in the common configuration file.

See examples of the import process in the chapter [Example of Importing a RAID](#).

## raid import show

To show info about the RAIDs that can be imported (restored) from the drives, run

```
# xicli raid import show [optional_args]
```

### Arguments for the `import show` subcommand

#### Optional arguments

-d	--drives	<p>The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) separated by a space to show the info.</p> <p>Without the argument, shows the info from all drives.</p>
-f	--format	<p>Output format:</p> <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul> <p>The default: <b>table</b>.</p>
	--offline	<p>Show non-recoverable RAIDs in the import list.</p>

## Arguments for the `import show` subcommand (continued)

The argument takes no value.

### Possible conflicts:

- `name`: Conflict with `in-system RAID(s)`;
- `drives`: Conflict with `in-system RAID(s)`;
- `name`: Conflict with `import RAID(s)`;
- `drives`: Conflict with `import RAID(s)`;
- `name`: Conflict with `in-system and import RAID(s)`;
- `drives`: Conflict with `in-system and import RAID(s)`.

### Drives statuses (messages in the devices row):

- `no_metadata` – drive has no xiRAID Classic RAID metadata. After drive import, run drive reconstruction.
- `in_use` – drive is in in-system RAID. After import, the drive will go to the `offline` state.
- `normal` – drive works properly (the state may be changed after import).

### Example:

To show all RAIDds that are available for import, run:

```
# xicli raid import show
```

The command will find and display information about founded RAIDds on drives that can be imported. Three RAIDds available for import shown in the figure below:

RAIDd uuid	info	devices	serials	import status
073DE372-94D0-43B3-9982-BFC67F3824CC	size: 9 GiB level: 0 synd_cnt: 0 strip_size: 16 block_size: 4096	0 /dev/sdl normal 1 /dev/sdm normal	drive-sca11 drive-sca12	name: OK drives: OK
61BF2210-E1B0-4F76-9F01-02D29E1C715	size: 9 GiB level: 6 synd_cnt: 2 strip_size: 16 block_size: 4096	0 /dev/sdh in_use,no_metadata 1 /dev/sdi normal 2 /dev/sdj normal 3 /dev/sdk normal	drive-sca17 drive-sca18 drive-sca19 drive-sca10	name: Conflict with in-system RAID(s) drives: Conflict with in-system RAID(s) medial
D698D77D-1B10-48D6-99FE-B037E7B0C966	size: 14 GiB level: 7 synd_cnt: 3 strip_size: 16 block_size: 4096	0 /dev/sdb normal 1 /dev/sdc normal 2 null no_metadata 3 null no_metadata 4 /dev/sdf normal 5 /dev/sdg normal	drive-sca11 drive-sca12 null null drive-sca15 drive-sca16	name: Conflict with in-system and import RAID(s) drives: OK

Conflicts are highlighted in red color. The first RAID does not conflict with any RAIDds. The second RAID conflicts with in-system and import RAIDds by the name and disks. The third RAID has a conflict with in-system RAID by the name.

## raid import apply

To import the RAID from disk metadata, run

```
# xicli raid import apply <arg> [optional_arg]
```

### Arguments for the `import apply` subcommand

Required argument

---

<code>-id</code>	<code>--uuid</code>	UUID of the RAID.
------------------	---------------------	-------------------

---

Optional argument

---

<code>-nn</code>	<code>--new_name</code>	The new name for the RAID.
------------------	-------------------------	----------------------------

## Example of Importing a RAID

Example: move RAID "media5" to another system that already has a RAID with the same name:

1. Stop the workflow on the RAID.

2. Unload the RAID:

```
# xicli raid unload -n media5
```

3. Remove the RAID drives from this system and insert them into another system.

4. Check for import conflicts:


```
# xicli raid import show
```

5. Import the RAID with changing its name to "media5\_2":

```
# xicli raid import apply -id  
52538D69-CF99-4471-8C85-DD42C9026A22 -nn media5_2
```

## Managing CPUs when importing the RAID

The number of CPUs specified for a RAID is stored in the RAID configuration file. However, when importing a RAID to a system or migrating it to another node in an HA cluster, it is important to note that the number of available CPUs may differ.

 When allocating all available CPUs on a system to a RAID by enumerating them (e. g., 0-3) instead of using the `-all` flag, please note that only the specified CPUs will be used for this RAID on the system to which it is imported.

If the specified number of CPUs is invalid for the destination node system, the following error will occur:

```
"Error: The specified CPU number exceeds the maximum allowed:  
<allowed_cpu_number>."
```

In this scenario, after migration, the RAID will operate on all available CPUs in the destination node system and will obtain an additional parameter - `cpu_allowed_configured` - which indicates the number of CPUs specified in this RAID's the configuration file. This information is available in the output of the `xicli raid show -e` command.

RAID's name	static	state	devices	health	wear	serials	params	info
test	size: 9 GiB level: 1 strip_size: 128 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/sdb online 1 /dev/sdc online	100% 100%	N/A N/A	drive-scsi1 drive-scsi2	cpu_allowed : 0-2 cpu_allowed_configured: 0-7 init_prio : 50 memory_limit_mb : 0 memory_prealloc_mb : 2048 recon_prio : 50 request_limit : 0 restripe_prio : 100 sched_enabled : 0	memory_usage_mb : 2048

'cpu\_allowed\_configured' parameter

## Configuration File Recovery


You can recover xiRAID Classic RAID objects and system configuration using the command:

```
# xicli config <subcommand> <args> [optional_args]
```

Subcommands for the `config` command:

apply	Apply the configuration files for all restoring RAIDs.
backup	Save the common configuration file (create the backup file <i>backup_raid.conf</i> at the current directory).
restore	Restore the configuration file from a file or from the drives.
show	Show configuration files stored on the drives.


### config apply

 Warning! The result of the command is irreversible. Read the description carefully.

To apply the configuration file for all restoring RAIDs, run

```
# xicli config apply
```

The command unloads and destroys all RAIDs in the system whose configuration files are NOT present in the `/etc/xiraid/raids/*` directory, and then restores all RAIDs that can be restored (that depends on whether drives in a RAID configuration are available - see [Restoring the RAID](#)).

 Ensure that the drives specified in a RAID configuration are not part of another existing RAID on the system; otherwise, it will lead to conflicts. We recommend unmounting all devices before running this command.

## config backup

To save the current configuration file from `/etc/xiraid/raid.conf` (to create the backup file `backup_raid.conf` in the current directory), run

```
# xicli config backup
```

## config restore



Warning! The result of the command is irreversible when using the `--common_file` argument.

To restore the configuration file from a file or from the drives, run

```
# xicli config restore <arg>
```

The command *restores* (if missing) or *replaces* the common configuration file from a specified location (from a file or disk metadata), but *does not apply* it.

### Arguments for the `restore` subcommand

#### Mutually excluded required arguments

-c	<code>--common_file</code>	<p>A file to restore or replace the common configuration file <code>/etc/xiraid/raid.conf</code>.</p> <p>If no file is specified, restore or replace from <code>/etc/xiraid/raid.conf.bak</code>.</p>
-d	<code>--drives</code>	<p>The list of block devices (<code>/dev/sd*</code>, <code>/dev/mapper/mpath*</code>, <code>/dev/nvme*</code>, <code>/dev/dm-*</code>) separated by a space to restore the most recent RAID configuration files to the <code>/etc/xiraid/raids.drive/</code> directory.</p> <p>If no disks are specified, restore the most recent RAID configuration files from all disks.</p>

## Arguments for the `restore` subcommand (continued)

<code>-r</code>	<code>--raid_file</code>	A file to restore or replace the RAID configuration file in <code>/etc/xi-raid/raids/</code> .
-----------------	--------------------------	--

## config show

To show configuration files stored on the RAID drives, run

```
# xicli config show [optional_arg]
```

## Argument for the `show` subcommand

Optional argument

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.  Without the argument, show from all disks.  The command also shows the newest configuration file from the drives.
-----------------	-----------------------	---

## Examples of Restoring the Common Configuration File

**Example:** restore the common configuration file from disk metadata.

### RAID Configuration Update Behavior

If a RAID configuration with the corresponding UUID exists in the `/etc/xiraid/raids` directory, the system will handle it accordingly: if the RAID is unloaded, an attempt will be made to restore it; if the RAID is online, the configuration will simply be updated on all of its disks without unloading it.

However, if no RAID configuration with the corresponding UUID is found, the RAID will be destroyed. The RAID will be unloaded and its metadata will be wiped. It is important to ensure that all RAIDs are unmounted before executing the `xicli config apply` command to prevent any unintended data loss or RAID destruction.

To restore the common configuratino file from disk metadata, follow these steps:

1. Restore the most recent configuration among all drives. At this stage, the `/etc/xiraid/raids.drive/` folder appears, containing the configuration files for different RAID's named `<raidname>.conf`. The RAID configuration file paths look like: `/etc/xiraid/raids.drive/raidname.conf`

```
# xicli config restore -d
```

2. Now, to restore the file for a specific RAID from `/etc/xiraid/raid.drive/`, you need to select it and use the command:

```
# xicli config restore -r /etc/xiraid/raids.drive/  
<raidname>.conf
```

3. Apply the restored configuration:

```
# xicli config apply
```

**Example:** save a copy of the common configuration file to a flash drive.

1. Change the current directory to flash drive:

```
$ cd /mnt/<device>
```

2. Create a copy of the common configuration file:

```
# xicli config backup
```

## Restoring a RAID

RAIDs can be restored, either automatically or manually, using the configuration specified in the configuration file. After unloading a RAID, you can restore it manually. Following a system failure or a system and services reboot, RAID's are restored automatically. In these cases, the initialization timeout for all drives is set to 10 seconds.

However, if the RAID configuration includes drives that are not available in the host system for any reason, one of the following will occur:

- If the number of remaining drives in a RAID is sufficient for its operation, it will be restored to the 'degraded' state. If the missing drives later become available in the host system, they will be added to the RAID automatically and the reconstruction of this RAID will begin.



If an unexpected system reboot occurs, the initialization will automatically start to prevent write holes after RAID restoration, provided that the resync function is enabled (which is the default setting).

- If the number of remaining drives in a RAID is insufficient for its proper operation, the RAID will not be restored. In such case, contact the xiRAID Classic support team at [support@xinnor.io](mailto:support@xinnor.io).

The log message may contain the following warning if not all drives included in the RAID configuration have been initialized:

```
ERROR :: For RAID 'raidname' not all devices were initialized.
Uninitialized devices are 'drive-serial1, drive-serial2'.
```



If the system does not have enough available memory to restore a RAID with its reserved memory, the RAID will be restored without the reserved memory. To restore the RAID with its reserved memory, you must ensure that the amount of reserved memory does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free.

You can restore RAID's manually from the configuration files using

```
# xicli raid restore <arg>
```

## Arguments for the `restore` subcommand

Mutually exclusive required arguments

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

---

## Arguments for the `restore` subcommand (continued)

<code>-a</code>	<code>--all</code>	Restore all available xiRAID Classic RAIDs.
		Argument takes no value.

## Update Management

Update Check Service allows you to control the process of updating xiRAID Classic packages to newer versions. After installing xiRAID Classic 4.3.0 or updating xiRAID Classic products, this service locks the current version of the packages, preventing it from being automatically updated on general system update commands (`apt/yum/dnf update`).

To check for an available update, run:

```
# xicli update check
```



For Ubuntu and Proxmox, it is necessary to update the package index before checking for xiRAID Classic updates. To update the package index, run the following command:

```
# apt update
```

Disable the Update Check Service to update xiRAID Classic 4.3.0 to a new available version.



Please, follow the instructions provided at [xinnor.io](https://xinnor.io) to safely update your xiRAID Classic. Ignoring these steps may result in filesystem panick and even data loss.

To disable the Update Check Service, run:

```
# xicli update prepare
```



Please, do not run this command unless there is a new available xiRAID Classic version. Otherwise, the proper functioning of xiRAID Classic cannot be guaranteed.



The Update Check Service will inform you of any detected mounted xiRAID Classic devices. Please, unmount the devices before continuing the update process.

We recommend setting up email notifications using [xiRAID Classic Administrator's Guide](#), so we could inform you about the latest xiRAID Classic releases. The notifications will be sent to you once every three days. The corresponding messages will be created in the log file regardless of your notifications settings.

## Notifications

In xiRAID Classic, you can configure email notifications to stay informed about the system's functioning.

## System Log

Logs contain information about the system status and system operations at specific points in time. The logs are recorded in the system log (journalctl).



Collect system logs and attach them when contacting the [support@xinnor.io](mailto:support@xinnor.io) team in the event of system or RAID malfunction.

You can:

- manage and view the type of system messages that will be logged;
- collect logs into a file;
- view the latest error and warning messages on the system.

To configure the type of system messages that will be added to the system log, run

```
# xicli settings log modify <arg>
```

## Argument for the `log modify` subcommand

Required argument

---

<code>-l</code>	<code>--level</code>	The type of system messages that will be added to the system log.  Possible values: <b>error</b> , <b>warning</b> , <b>info</b> , <b>debug</b> .  Each next type includes the previous one.  The default: <b>info</b> .
-----------------	----------------------	---

To collect all logs into a file, run

```
# xicli log collect
```

The log file will be available in the `/var/log/xiraid/` directory.

After collecting is complete, the message "*xiRAID Classic logs have been collected and saved in: /var/log/xiraid/xiraid\_xinnor\_2024.04.22\_06-58-47.tar.gz*" shows.

To see the selected type of system messages for the system log, run

```
# xicli settings log show [optional_arg]
```

To see the latest error and warning messages on the system, run

```
# xicli log show [optional_arg]
```

## Argument for the `show` subcommand

Optional argument

---

<code>-l</code>	<code>--lines</code>	The number of error and warning messages in the event log to show, starting from the last entry.  Possible values: integers from <b>1</b> to <b>1000</b> .  The default: <b>10</b> .
-----------------	----------------------	--

## Setting up Email Notifications



Make sure the system has configured MTA (Mail Transfer Agent) (for example, Postfix, Sendmail or Exim).

The MTA must meet the following requirements:

- It should be able to accept standard MIMEText messages.
- It must be accessed at `/usr/sbin/sendmail` — either by hard or symbolic link.
- It must be enabled to start at system startup:

```
# systemctl enable <postfix/sendmail/exim/...>
```

To add an email notification recipient and configure their notification level, run the following command:

```
# xicli mail add <args>
```



When you change any parameter of the `xicli mail add` command, the `xiraid-mail.service` automatically restarts.

## Arguments for the `add` subcommand

### Required arguments

<code>-a</code>	<code>--address</code>	Receiver's email.
<code>-l</code>	<code>--level</code>	The notification level.  Possible values: <ul style="list-style-type: none"> <li>• <b>info</b> – All notifications;</li> <li>• <b>warning</b> – Error and Warning notifications;</li> <li>• <b>error</b> – Error notifications.</li> </ul> <p>Refer to <a href="#">Email Notification Categories</a> for notification examples.</p>

Example: Add the receiver with the “`user2@email.com`” email for all notification types:

```
# xicli mail add -a user2@email.com -l info
```

To send a test email notification to all configured recipients in the system, run the following command:

```
xicli mail send
```

To remove an email from the list of notification recipients, run the following command:

```
# xicli mail remove <arg>
```



When you change any parameter of the `xicli mail remove` command, the `xiraid-mail.service` restarts.

## Argument for the `remove` subcommand

### Required argument

---

<code>-a</code>	<code>--address</code>	The email address to remove from the notifications.
-----------------	------------------------	---

To show the list of email notification recipients, run the following command:

```
# xicli mail show
```

## Argument for the `mail show` subcommand

### Optional argument

---

<code>-f</code>	<code>--format</code>	Output format: <ul style="list-style-type: none"><li>• <b>table</b>;</li><li>• <b>json</b>;</li><li>• <b>prettyjson</b> – human-readable json.</li></ul>
-----------------	-----------------------	--

The default: **table**.

To manage email notification settings, run the following command:

```
# xicli settings mail modify <args>
```



When you change any parameter of the `xicli settings mail modify` command, the `xiraid-mail.service` restarts.

## Arguments for the `mail modify` subcommand

At least one argument is required

---

<code>-pi</code>	<code>--polling_interval</code>	<p>The polling interval for xiRAID Classic RAIDs and the drives in seconds.</p> <p>Possible values: integers from <b>0</b> to <b>86400</b> (24 hours).</p> <p>The default: <b>10</b>.</p>
------------------	---------------------------------	---

---

<code>-ppi</code>	<code>--progress_polling_interval</code>	<p>Polling interval for the progress of initialization and reconstruction, in minutes.</p> <p>Possible values: integers from <b>0</b> to <b>1440</b> (24 hours).</p> <p>The default: <b>10</b>.</p>
-------------------	--	---

To show email notification settings, run the following command:

```
# xicli settings mail show
```

## Argument for the `mail show` subcommand

Optional argument

---

<code>-f</code>	<code>--format</code>	<p>Output format:</p> <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul> <p>The default: <b>table</b>.</p>
-----------------	-----------------------	---

## Email Notification Categories

<b>Info</b>	<b>Warning</b>	<b>Error</b>
Initialization completed on RAID (...)	Initialization not completed on RAID (...)	RAID (...) is offline now
Initialization progress on RAID (...) is (...) percent	RAID (...) is read-only now	RAID (...) is unrecovered now
Initialization started on RAID (...)	System is up after re-boot/crash	After reboot/crash, RAID (...) not restored
RAID (...) is healthy now	Reconstruction not completed on RAID (...)	After reboot/crash, RAID (...) has restored in read only mode
RAID (...) is online now	SparePool (...) ran out of drives	RAID (...) is degraded now
Reconstruction completed on RAID (...)	The number of errors on the bdev (...) is increased. The current number is (...)	After reboot/crash, RAID (...) has restored in offline state
Reconstruction progress on RAID (...) is (...) percent		xiRAID Classic license expired
Reconstruction started on RAID (...)		xiRAID Classic license error: The total number of used drives (...) exceeds the license limit of (...) drives.
Drive (...) was returned to RAID (...)		Drive (...) in RAID (...) is offline now

Info	Warning	Error
Drive (...) from SparePool (...) was reconnected		Drive (...) from SparePool (...) was disconnected
Drive (...) in RAID (...) was automatically replaced with drive (...) from SparePool (...)		Could not automatically replace drive (...) in RAID (...) with drive (...) from SparePool (...)
Can't replace the faulty bdev (...). Replacing (...) with null		Could not automatically replace drive (...) in RAID (...) because there was no suitable drive in SparePool (...)
		The number of faults on the bdev (...) reached the fault threshold
		The bdev (...) has critical wear out (...)%

## Using RAID in HA cluster

xiRAID Classic RAID objects can be used in HA cluster configuration. To learn more, see [Integrating xiRAID Classic Into a Pacemaker Cluster](#).

## General Configuration Recommendations

### RAID Creation

Next recommendations are appropriate and depend on drives' parameters and vendors.

- The appropriate RAID level depends on the required availability level.

Level of availability as high as 99.999% can be achieved by using RAID 6 if the RAID consists of less than 20 drives. Use RAID 7.3 with more than 20 drives.

Level of availability as high as 99.999% can be achieved by using RAID 50 if the RAID consists of less than 16 drives. With more drives, use RAID 60 or RAID 70.

## Using NVMe-oF devices to create a RAID

1. xiRAID Classic allows using NVMe-oF devices to create a RAID. Set the `--ctrl-loss-tmo` parameter to `0` to prevent command freezing because of connection loss when using these devices. It is relevant to `nvme-cli` version `>= 1.4`.

```
# nvme connect -t rdma -n nqn.Xinnor12_1 -a 10.30.0.12 -s 4420
--ctrl-loss-tmo=0
```

2. At the creation of NVMe-oF target for xiRAID Classic RAID, you can enable Merge if the access pattern assumably will be sequential write.

Depending on the version of Linux Kernel or Mellanox drivers, NVMe-oF targets may split big requests to 32 KiB + the rest. This kind of behavior leads to constant *read-modify-writes*. For an SPDK NVMe-oF target, set the `InCapsuleDataSize` parameter denoting at by what value requests should be split.



xiRAID Classic relies on serial numbers to identify connected NVMe devices (both connected by PCIe and NVMe-oF); therefore, it is essential that the serial number of each particular drive (NVMe subsystem) is unique and persists regardless of any system events taking place (including NVMe host reboot, NVMe subsystem reboot, and so on). The same requirement applies to identifiers of subsystem namespaces (NSID). These requirements are usually met automatically by locally connected PCIe NVMe and known EBOFs, but have to be satisfied by manually configured NVMe targets.



In xiRAID Classic 4.3.0, subsystems with multiple namespaces cannot be used to create RAID. An NVMe-oF device can be used to create a RAID only if it is a subsystem with a single namespace.

## RAID and System Setup Recommendations

### **--init\_prio**

Syndrome RAID creation starts the initialization process automatically. During it, RAID is available for reading and writing operations. Since initialization priority by default is set to **50**, you can wait until the initialization is finished, or if the access pattern is *not random write*, you can lower the initialization priority. Therefore, user I/O will be processed faster due to the reduction of initialization requests.

### **--recon\_prio**

The reconstruction process starts automatically. By default, reconstruction priority equals to **50**.

### **--restripe\_prio**

The `modify` command enables to change restriping priority. If the priority value of the function is zero, restriping starts and continues only if there is no workload. By default, priority is set to **100%** that stands for the highest possible rate of the restriping process. To improve the system workload performance, try decreasing restripe priority.

## --sched\_enabled

There are 2 possible ways of handling an incoming request:

- continue execution on the current CPU;
- transfer the request to the other CPU core and continue execution. Note that it takes time for the transferring.

If the access pattern uses less than half of the system CPU, it is efficient to use the `--sched_enabled` parameter. When a lot of requests are processed by the single CPU core, enabling scheduling allows to redistribute the workload equally between all system CPUs. On multithreading access patterns, scheduling is inefficient, because useless transfer of requests from one CPU core to another wastes time.



Enable Scheduling when the access pattern is low threaded.

## --request\_limit

Defines the maximum number of incoming read requests per RAID.

By default, the system does not limit the number of incoming read requests. In some cases, this may lead to delays in processing individual requests. Adjusting the `--request_limit` value may reduce the total read speed but improve delay issues. The optimal value for this parameter depends on multiple factors such as system load, hardware, and file system characteristics.

If you observe delays in processing individual requests, contact our support team and we will help you determine an appropriate `--request_limit` value for your environment.

## --force\_online

If a RAID has unrecoverable sections, then the RAID becomes unreadable (get in the offline, unrecoverable state). To try to read available data, manually turn on the online mode for the RAID by running the command

```
# xicli raid modify -n <raid_name> --force_online
```

While in the mode, I/O operations on unrecoverable sections of the RAID may lead to data corruption.



If the RAID is 'offline', we recommend contacting [support@xinnor.io](mailto:support@xinnor.io) before using the command.

## Merge function

Data can be written to or read from drives in a RAID in sequential or random order. When incoming requests are sequential and the recording block sizes are small, it is beneficial to wait for them to accumulate and form a large block. Then, merge these requests and write or read data from the RAID in large blocks. Applying such access patterns can improve system workload performance, as this approach reduces the number of read-modify-write operations<sup>1</sup> on syndromic RAIDs.



Enable Merge function when the access pattern is sequential and high threaded and the block sizes are small.

## Manual Merge configuration

The merge parameters for requests accumulation for a RAID can be configured manually using `xicli raid create` and `xicli raid modify` commands.



The Merge function cannot be activated for RAIDs with a stripe size greater than 1 MiB.

The `--merge_read_enabled` parameter activates the Merge function for incoming read requests, allowing their accumulation. You can specify a waiting time between incoming read requests in sequential areas using the `--merge_read_wait` parameter. At the end of the waiting time, the requests are merged together if possible. You can use the `--merge_read_max` parameter to specify the maximum waiting time for request accumulation. Usually, large I/O sizes require large values for these parameters.

---

1. *Writing data to a RAID in small blocks requires reading data from the drives, calculating new syndrome values, and writing them to the drives.*

*Example:* Create the RAID 5 named "media5" over 4 NVMe drives — "nvme0n1", "nvme1n1", "nvme2n1", "nvme3n1", with strip size equal to 64 KiB and enabled Merge function for read operations.

```
# xicli raid create -n media5 -l 5 -d /dev/nvme0n1 /dev/nvme1n1 /
dev/nvme2n1 /dev/nvme3n1 -ss 64 -mre 1
```

The **--merge\_write\_enabled** parameter activates the Merge function for incoming write requests, allowing their accumulation. You can specify a waiting time between incoming write requests in sequential areas using the **--merge\_write\_wait** parameter. At the end of the waiting time, the requests are merged together if possible. You can use the **--merge\_write\_max** parameter to specify the maximum waiting time for request accumulation. Usually, large I/O sizes require large values for these parameters.

*Example:* Change the RAID 5 named "media5" waiting time between write requests from 300 to 500 ms.

```
# xicli raid modify -n media5 -mww 500
```

If the access pattern is mainly random or request queue depth is small, the waiting time will not allow merging requests.

The Merge function can be enabled if the following condition is met:

```
data_drives * strip_size ≤ 1024
```

where

- "data\_drives" is a number of drives in the RAID (for RAIDs 5, 6, 7.3 or N+M) or in one RAID group (for RAIDs 50, 60, or 70) that are *dedicated for data*;
- "strip\_size" is a selected stripe size for the RAID (**strip\_size** value) in KiB.

The "data\_drives" value depending on a RAID level:

RAID level	Value of data_drives
RAID 5	Number of RAID drives <i>minus</i> 1
RAID 6	Number of RAID drives <i>minus</i> 2

RAID level	Value of data_drives
RAID 7.3	Number of RAID drives <i>minus</i> 3
RAID N+M	Number of RAID drives <i>minus</i> M
RAID 50	Number of drives in one RAID group <i>minus</i> 1
RAID 60	Number of drives in one RAID group <i>minus</i> 2
RAID 70	Number of drives in one RAID group <i>minus</i> 3

Deactivate Merge when the request queue depth of user's workload is not enough to merge a full stripe. Activate Merge, if

```
iodepth * block_size >= data_drives * strip_size
```

where "block\_size" is a block size of the RAID (the **block\_size** value in RAID parameters) in KiB.

## Automatic Merge configuration

Adaptive merge in xiRAID Classic 4.3.0 is considered as experimental feature. This means that we continue to run our tests on a variety of configurations and environments, and based on our results and your feedback, we will plan and implement further improvements of the algorithm.

We recommend to use it in situations when your business applications originate a significant continuous sequential write load:

- For the case when application logic generates a changing sequential load or there are several sources of concurrent sequential load, we recommend using the adaptive merge **without the --single\_run** option, which is also the default mode.
- The **--single\_run** option is recommended for those special cases where you want to adapt merge parameters for only one specific scenario that reproduces inconsistently or can be interrupted/changed by sporadic application-level activities. In this case, enable the adaptive merge with the **--single\_run** option during the desired scenario to permanently set the merge parameters.

In real business configurations, there could be additional factors unique to each particular setup. We highly recommend observing the performance of xiRAID with the adaptive merge option prior to enabling it in critical production environments to confirm that it brings a noticeable positive impact in each particular situation.

xiRAID offers an algorithm that automatically selects the waiting time for accumulating write requests. This algorithm determines whether the merging of write requests is enabled (`--merge_write_enabled` parameter), the wait time between write requests (`--merge_write_wait` parameter) and the maximum wait time for stripe accumulation (`--merge_write_max` parameter). The time between incoming requests is set to 0-3000 ms, and the maximum wait time for stripe accumulation is set to 20000 ms.

You can enable the Automatic Merge function during RAID creation and modify it later using the **--adaptive\_merge** parameter. Once the Automatic Merge algorithm determines the waiting time, you can prevent it from changing these settings later on using the **--single\_run** parameter.

*Example:* Enable the Adaptive Merge function for the RAID 5 named "media5" and prevent it from changing the determined settings later on. After the settings have been determined, the Adaptive Merge is automatically deactivated and displayed as 'False' in the `xicli raid show` output.

```
# xicli raid modify -n media5 --adaptive_merge 1 --single_run
```



Do not use the Automatic Merge parameters when Manual Merge is enabled, and vice versa.

## RAM Limit

Current memory usage is being monitored and controlled to be within the limit. You can modify the **--memory\_limit** parameter at any time. By default, memory usage is unlimited.

Deactivating monitoring of current memory usage and limitation control can improve system workload performance. Set **--memory\_limit** to **0** to deactivate monitoring with the modify command.

If it is necessary to limit the use of RAM, we recommend choosing amount of RAM depending on the selected strip size for the RAD:

Strip size, in KiB	Amount of RAM, in MiB
16	2048
32	2048
64	4096
128	8192
256	16384

## NUMA

1. Create a RAID out of drives belonging to the same NUMA node, if your systems are multiprocessor.

To figure out the NUMA node drive, run:

```
# cat /sys/block/nvme0n1/device/device/numa_node
```

or via lspci:

```
# lspci -vvv
```

2. At creation of NVMe-oF target for xiRAID Classic RAID, you can use network adapter of the same NUMA node as NVMe drives.

## System

1. xiRAID Classic shows better performance with enabled hyper-threading (HT).

To find out if there is HT support on the CPU, run

```
# cat /proc/cpuinfo | grep ht
```

In the flags field, check for the `ht` flag.

Command output example:

```
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
      cmov pat pse36 clflush
mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
      constant_tsc arch_perfmon rep_good
nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma
      cx16 pcid sse4_1 sse4_2
x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand
      hypervisor lahf_lm abm
3dnowprefetch cpuid_fault invpcid_single pti tpr_shadow vnmi
      flexpriority ept vpid ept_ad
fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm
      rdseed adx smap xsaveopt
arat umip arch_capabilities
```

To check if HT is enabled, run

```
# lscpu
```

If Thread(s) per core is 1, then HT is off. HT can be enabled in BIOS/UEFI.

Command output example:

```
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Byte Order:                  Little Endian
Address sizes:                40 bits physical, 48 bits
                             virtual
CPU(s):                      4
On-line CPU(s) list:         0-3
Thread(s) per core:          1
...
```

2. The tuned-adm profile set to **throughput-performance** provides better performance on most of the tests:

```
# tuned-adm profile throughput-performance
```

## Workload

In xiRAID Classic 4.3.0, user I/O tends to be executed on the same CPU on which the user sent them. However, for some access patterns, you can transfer I/O commands to other CPUs, so the commands will not idle. You can enable I/O Scheduling to all system CPU using a parameter `--sched-enabled` (**1** – activated, **0** – deactivated).

Activating and deactivating the Scheduling function depending on the access pattern recommendations are provided in the `--sched_enabled` section.

## Swap File

On high load servers, we recommend to disable swap file usage to increase server performance.

## File System Mounting Examples

To ensure file systems for xiRAID Classic RAIDs are automatically mounted at system startup, you can configure the system to mount these file systems using either `systemd` or `fstab`.

## systemd

Follow the instructions below to configure a systemd mount unit:

1. Create a new systemd unit file. The filename must match the path to the mount directory, replacing `/` with `-` and appending `.mount` at the end. For example, if the mount directory is `/mnt/raid`, the filename should be `mnt-raid.mount`. To create a new unit file, run the following command:

```
systemctl edit --full --force mnt-raid.mount
```

2. Use the following unit file as a template:

```
[Unit]
Description=Mount filesystem on xiRAID Classic
DefaultDependencies=no
Before=umount.target
Conflicts=umount.target

[Mount]
What=/dev/xi_raid
Where=/mnt/raid/
Options=defaults
Type=xf

[Install]
WantedBy=multi-user.target
WantedBy=dev-xi_raid.device
```

- `/dev/xi_raid` is the device to mount.
- `/mnt/raid/` is the mount point.
- The `WantedBy` option specifies when to mount the device. The value should be the device name with `dev-` prepended and `.device` appended. For example, if the device is `/dev/xi_raid`, the corresponding value for `WantedBy` is `dev-xi_raid.device`.

3. After saving the unit file, enable the mount unit to automatically mount the device at system startup by running the following command:

```
# systemctl enable mnt-raid.mount
```

4. To manually start the mount service and mount the file system, run the following command:

```
# systemctl start mnt-raid.mount
```

## fstab

Follow the instructions below to configure a mount using the `/etc/fstab` file:

1. Open the `/etc/fstab` file for editing.
2. Use the following entry as a template:

```
/dev/xi_raid /mnt/raid/ xfs
defaults,x-systemd.wanted-by=dev-xi_raid.device 0 0
```

- `/dev/xi_raid` is the device to mount. Use the block device name (e.g., `/dev/xi_raid`) rather than the UUID or LABEL.
  - `/mnt/raid/` is the mount point.
  - `x-systemd.wanted-by` specifies when to mount the device. The value should be the device name with `dev-` prepended and `.device` appended. For example, if the device is `/dev/xi_raid`, the corresponding value for `x-systemd.wanted-by` is `dev-xi_raid.device`.
3. Save the changes to `/etc/fstab` and run the following command to mount the device:

```
# mount -a
```

## DKMS Specifics when Updating Linux Kernel



If you downgrade kernel version, DKMS functionality depends on the specific distribution.

The *xiraid* kernel module uses DKMS (Dynamic Kernel Module Support) technology and is automatically built and is installed for the Linux kernel versions, listed in the

document xiRAID Classic 4.3.0 System Requirements, of the different patch versions (without kernel API or ABI changes).

For example:

- 3.10.0-**1062**.el7.x86\_64 >> 3.10.0-**1127**.el7.x86\_64;
- 4.15.0-**112**-generic >> 4.15.0-**124**-generic.

Notice, that if you update the kernel more than the patch update (with kernel API or ABI changes), the *xiraid* kernel module will not be loaded. For example these updates will not work:

- 3.10.0-1062.el7.x86\_64 >> 4.18.0-193.el8.x86\_64;
- 4.**15**.0-112-generic >> 4.**18**.0-13-generic;
- 4.15.0-112-generic >> 5.4.0-26-generic.

To update (or change) a Linux kernel version with the installed *xiraid* module with DKMS, the OS must have a package with the header files for the kernel version to be updated:

- kernel-devel (for RHEL and RHEL-based systems);
- kernel-uek-devel (for Oracle Linux);
- linux-headers (for Ubuntu);
- pve-headers (for Proxmox).

Since some OS distributions do not have by default a package with header files (and also some repositories may not have package versions for out-of-date kernel versions), we recommend to install a package with header files for a new kernel version manually before (or simultaneously with) installing a new kernel version (see examples of commands to install packages with headers for different operating systems in the xiRAID Classic 4.3.0 Installation Guide.)

For example, on Ubuntu 20.04, install the linux-image package at the same time as the linux-headers package:

```
# apt install linux-image-5.15.0-27-generic  
linux-headers-5.15.0-27-generic
```

**After updating to a newer kernel, the xiRAID Classic kernel module might not update correctly without matching kernel headers.**

After updating the system to a newer kernel, the xiRAID Classic kernel module may not update correctly, which could lead to existing RAID arrays not being restored after reboot, leaving them in the 'none' state.

To resolve this issue, install new headers and rebuild the xiRAID Classic kernel drivers using DKMS:

1. Install the header package for your current kernel for your OS using the xiRAID Classic Installation Guide.
2. Rebuild the xiRAID Classic kernel module with DKMS by running:

```
# dkms autoinstall
```

3. On success, restart the xiRAID Classic target service by running:

```
# systemctl restart xiraid.target
```

4. If these steps don't solve the issue, please boot your system on a previous kernel and contact [support@xinnor.io](mailto:support@xinnor.io).

## Authenticating gRPC Client

To authenticate a gRPC client, set up a host and a port for the connection, and replace a TLS/SSL public key certificate if necessary. For gRPC connections, only server needs to provide its certificate to client (the server-side TLS connection type).

To set up client-server connection settings, run the command



When you change any parameter of the `xicli settings auth modify` command, the `xiraid-target.service` restarts. Additionally, it will cause all RAIDs to unload. Please, run this command only after stopping all mounted devices.



The command neither requires acceptance of the EULA nor running `xiraid.target` service.

```
# xicli settings auth modify <args>
```

## Arguments for the `auth modify` subcommand

At least one argument is required

---

`--host`

The host name or IP address that will be used for the connection.

After changing the host, you must re-generate and replace the certificate.

The default: **localhost**.

---

`--port`

The port that will be used for the connection.

The default: **6066**.

To view client-server connection settings, run the command

```
# xicli settings auth show
```

## Argument for the `auth show` subcommand

Optional argument

---

`-f`

`--format`

Output format:

- **table**;
- **json**;
- **prettyjson** – human-readable json.

The default: **table**.

To replace the certificate:



The certificates require that the system time is not earlier than June 24, 2019.

1. copy the files to `/etc/xraid/crt/` that are strictly named

- `server-key.key` (or `server-key.pem`);
- `server-cert.crt` (or `server-cert.pem`);
- `ca-cert.crt` (or `ca-cert.pem`);

If there are `.pem` and `.key/.crt` files in `/etc/xraid/crt/` at the same time, the system will use `.key/.crt` files.

2. restart the xiraid service:

```
# systemctl restart xiraid.target
```

## Updating GPG Keys

On Ubuntu and Proxmox, you may encounter errors related to the expiry of the `pkg.xinnor.io` GPG keys when running `apt update` or updating from xiRAID Classic 4.2.0. Execute the commands below to update the keys for `pkg.xinnor.io`:

1. 

```
# gpg --no-default-keyring \  
--keyring gnupg-ring:/etc/apt/trusted.gpg.d/xraid.gpg \  
--fetch-keys https://pkg.xinnor.io/repository/Repository/keys/  
xiraid/public.gpg.key
```
2. 

```
# chmod 644 /etc/apt/trusted.gpg.d/xraid.gpg*
```

Run `apt update` to verify that everything is working as expected.

# Troubleshooting

1. When attempting to use `xicli raid` commands, the following error occurs:

## **Error: Missing xiRAID Classic RAID system module**

Possible reasons:

- After updating the OS kernel, the packages with the header files (kernel-devel, kernel-uek-devel, linux-headers, pve-headers) remain from the previous kernel version.
- Linux kernel update that is more than the patch update.

Solutions:

- Update or install the package with the kernel header files (kernel-devel, kernel-uek-devel, linux-headers, pve-headers) for the updated or installed OS kernel version. See the xiRAID Classic 4.3.0 Installation Guide for details.

After that, run the command

```
# dkms autoinstall
```

- Load on the Linux kernel version that was before the kernel update.

2. When attempting to use some `xicli` commands, the following error occurs:

## **Error: failed to connect to all addresses**

Possible reason:

The client attempted to access the server using incorrect credentials, incorrect SSL certificates, or the old address that had been changed.

Solutions:

- Restart xiraid target:

```
# systemctl restart xiraid.target
```

- If the previous step did not help, update the certificates.

## Command Reference

### Overview

Manage your software xiRAID Classic in Linux by using the `xicli` program.

Most of the commands listed in this document require superuser privileges. Please log in as an administrator or root to run these. However, the following commands can be run without superuser privileges: all commands with the `show` subcommand (`raid show`, `config show`, `drive faulty-count show`, `settings eula show`, `license show` etc) and any command with the `--help` parameter.

### Command Line Interface (CLI) Description

#### Conventions on CLI command syntax

Item format	Description
item	A required item (command, subcommand, argument, option).
<item>	A placeholder variable.
[item]	An optional item.

In the CLI, enter commands in the following format:

```
# xicli <command> <subcommand> <required_args> [optional_args]
```

To show the full list of commands, run

```
# xicli -h
```

To show the `xicli` version and the RAID driver version, run

```
# xicli -v
```

CLI syntax specifics:

1. Type the arguments of the subcommands in one line.
2. Subcommand arguments are separated by spaces.
3. Use short or long forms of subcommand argument options.
4. To get the list of all subcommands and arguments, add the `-h` option:

```
# xicli <command> <subcommand> -h
```

A detailed description of the commands and subcommands is presented in the corresponding sections of the document.

## config

Operations with the configuration file.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli config <subcommand> <args> [optional_args]
```

Subcommands for the `config` command:

<code>apply</code>	Apply the configuration files for all restoring RAIDs.
<code>backup</code>	Save the common configuration file (create the backup file <i>backup_raid.conf</i> in the current directory).
<code>restore</code>	Restore the configuration file from a file or from the drives.
<code>show</code>	Show configuration files stored on the drives.

## apply

The command unloads and destroys all RAIDs in the system whose configuration files are NOT present in the `/etc/xiraid/raids/*` directory, and then restores all RAIDs that can be restored (that depends on whether drives in a RAID configuration are available - see [Restoring the RAID](#)).

```
# xicli config apply
```

## backup

Save the common configuration file from `/etc/xiraid/raid.conf` (create the backup file `backup_raid.conf` in the current directory).

```
# xicli config backup
```

## restore

Restore the configuration file from a file or from the drives.

```
# xicli config restore <arg>
```

### Arguments for the `restore` subcommand

Mutually excluded required arguments

---

<code>-c</code>	<code>--common_file</code>	A file to restore or replace the common configuration file <code>/etc/xiraid/raid.conf</code> .  If no file is specified, restore or replace from <code>/etc/xiraid/raid.conf.bak</code> .
<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to restore the most recent RAID con-

---

## Arguments for the `restore` subcommand (continued)

figuration files to the `/etc/xi-raid/raids.drive/` directory.

If no disks are specified, restore the most recent RAID configuration files from all disks.

---

<code>-r</code>	<code>--raid_file</code>	A file to restore or replace the RAID configuration file in <code>/etc/xi-raid/raids/</code> .
-----------------	--------------------------	--

## show

Show configuration files stored on the RAID drives.

```
# xicli config show [optional_arg]
```

### Argument for the `show` subcommand

Optional argument

---

<code>-d</code>	<code>--drives</code>	<p>The list of block devices (<code>/dev/sd*</code>, <code>/dev/mapper/mpath*</code>, <code>/dev/nvme*</code>, <code>/dev/dm-*</code>) separated by a space.</p> <p>Without the argument, show from all disks.</p> <p>The command also shows the newest configuration file from the drives.</p>
-----------------	-----------------------	---

## drive

Operations with the drives.

The commands `clean` and `locate` require superuser privileges.

```
# xicli drive <subcommand> <args> [optional_args]
```

Subcommands for the `drive` command:

<code>clean</code>	Delete the metadata and reset the number of failures from the drives.
--------------------	---

---

<code>faulty-count reset</code>	Reset the current number of failures for drives.
---------------------------------	--

---

<code>faulty-count show</code>	Show the current number of failures for drives.
--------------------------------	---

---

<code>locate</code>	Manage the drive LED indication.
---------------------	----------------------------------

## clean

Delete the metadata and reset the fault counter from the drives.

```
# xicli drive clean <arg>
```

### Argument for the `clean` subcommand

Required argument

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to delete metadata and reset the fault counter.
-----------------	-----------------------	---

## faulty-count reset



When you change any parameter of the `xicli drive faulty-count reset` command, the `xiraid-scanner.service` restarts.

Reset the current numbers of failures for drives.

```
# xicli drive faulty-count reset <arg>
```

## Arguments for the `faulty-count reset` subcommand

### Required argument

---

-d	--drives	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to reset their current numbers of failures.
----	----------	---

## faulty-count show

Show the current numbers of failures for drives.

```
# xicli drive faulty-count show [optional_args]
```

## Arguments for the `faulty-count show` subcommand

### Mutually exclusive optional arguments

---

-n	--names	The RAID name for which drives the current number of failures will be shown.
		If neither of the two arguments is specified, show the values for all drives.

---

-d	--drives	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to show their current numbers of failures.
		If neither of the two arguments is specified, show the values for all drives.

---

### Optional argument

---

## Arguments for the `faulty-count show` subcommand (continued)

<code>-f</code>	<code>--format</code>	Output format:
		<ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul>

The default: **table**.

## locate

Manage the drive LED indication.

```
# xicli drive locate <arg>
```

### Argument for the `locate` subcommand

Required argument

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to switch the indication on, or switch the indication off (with the <b>null</b> value).
-----------------	-----------------------	---

The argument doesn't affect the automatic indication.

## license

Operations with the license.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli license <subcommand>
```

Subcommands for the `license` command:

`delete` Delete the current license.

---

`show` Show info on the current license.

---

`update` Update the current license.

## delete

Delete the current license.

```
# xicli license delete
```

## show

Show info on the current license.

```
# xicli license show
```

## update

Update the current license.

```
# xicli license update <arg>
```

### Argument for the `update` subcommand

Required argument

---

`-p`      `--path`      The path to the new license file.

## log

Operations with the event log.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli log <subcommand> <args>
```

Subcommands for the `log` command:

<code>collect</code>	Collect the event log entries into a file.
----------------------	--

---

<code>show</code>	Show the last entries in the event log.
-------------------	---

## collect

Collect the event log entries into a file in `/var/log/xiraid/`.

```
# xicli log collect
```

## show

Show the latest error or warning messages in the event log related to `xiraid`.

```
# xicli log show [optional_arg]
```

### Argument for the `show` subcommand

Optional argument

---

<code>-l</code>	<code>--lines</code>	The number of error and warning messages in the event log to show, starting from the last entry.  Possible values: integers from <b>1</b> to <b>1000</b> .  The default: <b>10</b> .
-----------------	----------------------	--

## mail

Operations with mail notifications.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli mail <subcommand> <args> [optional_args]
```

Subcommands for the `mail` command:

<code>add</code>	Add an email notification recipient.
<code>remove</code>	Remove an email from the list of notification recipients.
<code>show</code>	Show the list of email notification recipients.
<code>send</code>	Send a test email notification.

## add



When you change any parameter of the `xicli mail add` command, the `xiraid-mail.service` restarts.

Add an email notification recipient and configure their notification level.

```
# xicli mail add <args>
```

### Arguments for the `add` subcommand

Required arguments

<code>-a</code>	<code>--address</code>	Receiver's email.
<code>-l</code>	<code>--level</code>	The notification level.

## Arguments for the `add` subcommand (continued)

Possible values:

- **info** – All notifications;
- **warning** – Error and Warning notifications;
- **error** – Error notifications.

Refer to [Email Notification Categories](#) for notification examples.

## remove



When you change any parameter of the `xicli mail remove` command, the `xiraid-mail.service` restarts.

Remove an email from the list of notification recipients.

```
# xicli mail remove <arg>
```

## Argument for the `remove` subcommand

Required argument

---

<code>-a</code>	<code>--address</code>	The email address to remove from the notifications.
-----------------	------------------------	---

## show

Show the list of email notification recipients.

```
# xicli mail show
```

## Argument for the `show` subcommand

### Optional argument

---

`-f`            `--format`            Output format:

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

## send

Send a test email notification to all configured recipients in the system.

```
# xicli mail send
```

## pool

Operations with spare pools.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli pool <subcommand> <args> [optional_args]
```

Subcommands for the `pool` command:

<code>add</code>	Add drive(s) to a spare pool.
<code>create</code>	Create a spare pool.
<code>delete</code>	Delete a spare pool.
<code>remove</code>	Remove drive(s) from a spare pool.

---

show                      Show details about a spare pool.

---

activate                  Activate a spare pool.

---

deactivate                Deactivate a spare pool.

## add

Add drive(s) to the spare pool.

```
# xicli pool add <args>
```

### Arguments for the **add** subcommand

Required arguments

---

-n              --name              The name of the spare pool.

---

-d              --drives              The list of block devices (/dev/sd\*, /dev/mapper/mpath\*, /dev/nvme\*, /dev/dm-\*) separated by a space.

## create

Create the spare pool.

```
# xicli pool create <args>
```

### Arguments for the **create** subcommand

Required arguments

---

-n              --name              The name of the spare pool.

---

## Arguments for the `create` subcommand (continued)

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.
-----------------	-----------------------	--

## delete

Delete the spare pool.

```
# xicli pool delete <arg>
```

## Argument for the `delete` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

## remove

Remove drive(s) from the spare pool.

```
# xicli pool remove <args>
```

## Arguments for the `remove` subcommand

Required arguments

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

---

<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space.
-----------------	-----------------------	--

## show

Show info on the spare pool.

```
# xicli pool show [optional_args]
```

### Arguments for the `show` subcommand

#### Optional arguments

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.  Without the argument, show info on all spare pools.
-----------------	---------------------	--

---

<code>-f</code>	<code>--format</code>	Output format: <ul style="list-style-type: none"><li>• <code>table</code>;</li><li>• <code>json</code>;</li><li>• <code>prettyjson</code> – human-readable json.</li></ul> The default: <code>table</code> .
-----------------	-----------------------	--

---

<code>-u</code>	<code>--units</code>	Size units: <ul style="list-style-type: none"><li>• <code>s</code> – sectors (1 sector=512 bytes);</li><li>• <code>k</code> – kilobytes;</li><li>• <code>m</code> – megabytes;</li><li>• <code>g</code> – gigabytes.</li></ul> The default: <code>g</code> .
-----------------	----------------------	--

## activate

Use the following command to activate a spare pool.

```
# xicli pool activate <arg>
```

### Argument for the `activate` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

## deactivate

Use the following command to deactivate a spare pool.

```
# xicli pool deactivate <arg>
```

### Argument for the `deactivate` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the spare pool.
-----------------	---------------------	-----------------------------

## raid

Operations with the RAIDs.

Except for the `show` and `import show` subcommands, the commands listed in this chapter require superuser privileges.

```
# xicli raid <subcommand> <args> [optional_args]
```

Subcommands for the `raid` command:

create	Create the RAID.
destroy	Delete the RAID without possibility to restore the RAID and data on it.
import apply	Import (or restore) the RAID from drive metadata.
import show	Show info about the RAIDs that can be imported (restored) from the drives.
init start	Start or continue the RAID initialization.
init stop	Stop the RAID initialization.
init reset	Reset the initialization counter, set the state of the RAID to <code>need_init</code> , and start RAID reinitialization.
modify	Modify the parameters of the created RAID.
recon start	Start the raid reconstruction.
recon stop	Stop the RAID reconstruction.
replace	Replace or remove the drive from the RAID.
resize	Change the RAID size.
restore	Restore the RAID from the drive metadata.
restripe continue	Continue the RAID restripe.
restripe start	Start the RAID restripe.

restripe stop	Pause the RAID restripe.
show	Show info about the RAID.
unload	Remove (unload) the RAID with possibility to restore the RAID and save data on it.

## create

Create the RAID.

```
# xicli raid create <args> [optional_args]
```

### Arguments for the `create` subcommand

#### Required arguments

-n	--name	<p>The name of the RAID.</p> <p>The maximum RAID name length is <b>28</b> characters. The RAID name may contain Latin letters, numbers, and underscores (<code>_</code>). Additionally, the following names are not permitted: <b>power</b> and <b>uevent</b>.</p> <p>To avoid naming conflicts between RAIDs and their partitions, avoid names that could overlap with partition identifiers. For example, do not create a RAID named <code>test1</code> if a RAID named <code>test</code> already exists, as partitions of <code>/dev/xi_test</code> may generate identifiers such as <code>/dev/xi_test1</code>, leading to conflicts.</p>
-l	--level	<p>The level of the RAID: <b>0</b>, <b>1</b>, <b>5</b>, <b>6</b>, <b>7</b>, <b>10</b>, <b>50</b>, <b>60</b>, <b>70</b>, or <b>nm</b>.</p>



Use the value **7** to create RAID 7.3.

## Arguments for the `create` subcommand (continued)

`-d`      `--drives`

The list of block devices (`/dev/nvme*`) separated by spaces.

The order in which drives are specified during RAID creation directly influences how the drives are distributed across RAID groups. Drives are assigned to RAID groups in the exact sequence they are listed in the RAID creation command. For instance, in the case of a RAID 10 configuration, if drives are listed as `-d /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1`, the first two drives, `/dev/nvme0n1` and `/dev/nvme1n1`, will be placed in the first RAID group, while the next two drives, `/dev/nvme2n1` and `/dev/nvme3n1`, will be allocated to the second RAID group.

---

`-gs`      `--group_size`

**Only for RAIDs 10, 50, 60, or 70.**

The number of drives for one RAID group of level 5, 6, or 7.3 of the appropriate RAID 50, 60, or 70.

Possible values are integers from **4** to **32**.

---

`-sc`      `--synd_cnt`

**Only for RAIDs N+M.**

The number of syndromes M.

Possible values are integers from **4** to **32**.

Additional conditions:  $N+M \leq 64$  and  $M \leq N$ .

---

### Optional arguments

---

`-am`      `--adaptive_merge`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Adaptive Merge write function.

---

## Arguments for the `create` subcommand (continued)

`--single_run`

**Except RAIDs 0, 1, 10.**

Use this parameter to adjust **the Adaptive Merge values** once at startup. After that, the values are set and do not change at system reboot. The Adaptive Merge write function is then turned off.

Does not take any value.

**Can only be used with the `--adaptive_merge` parameter.**

---

`-bs`      `--block_size`

RAID block size: **512** or **4096** bytes.

The default: **4096**.

---

`-ca`      `--cpu_allowed`

Specify the CPUs on which the RAID will be allowed to run.

Possible values: a comma-separated list of CPUs, a range of CPUs indicated by a hyphen, or the value 'all' (the RAID will run on all available CPUs).

The default: `all`.

---

`-inp`      `--init_prio`

**Except RAID 0.**

Initialization priority in %.

Possible values are from **1** to **100** (maximum rate of initialization).

The default: **50**.

---

`-mwe`      `--merge_write_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for write operations.

## Arguments for the `create` subcommand (continued)

The default: **0**.

---

`-mre`      `--merge_read_enabled`

**Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge function for read operations.

The default: **0**.

---

`-mrm`      `--merge_read_max`

**Except RAIDs 0, 1, 10.**

Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for read requests.

Possible values: integers from **1** to **100000**.

The default: **1000**.

---

`-mrw`      `--merge_read_wait`

**Except RAIDs 0, 1, 10.**

Wait time (in microseconds) between read requests with the Merge function enabled.

Possible values: integers from **1** to **100000**.

The default: **300**.

---

`-mwm`      `--merge_write_max`

**Except RAIDs 0, 1, 10.**

Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for write requests.

Possible values: integers from **1** to **100000**.

The default: **1000**.

---

`-mww`      `--merge_write_wait`

**Except RAIDs 0, 1, 10.**

## Arguments for the `create` subcommand (continued)

Wait time (in microseconds) between write requests with the Merge function enabled.

Possible values: integers from **1** to **100000**.

The default: **300**.

---

`-ml`      `--memory_limit`

RAM usage limit in MiB.

Possible values: **0** and integers from **1024** to **1048576**.

The **0** value sets unlimited RAM usage.

The default: **0**.

---

`-mp`      `--memory_prealloc`

The amount of reserved memory to allocate in MiB. This amount cannot exceed the RAM usage limit (`--memory_limit`). Additionally, it must be ensured that the specified amount does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free.

Possible values: **0** and integers from **1024** to **65536**.

A value of **0** indicates that reserved memory will not be allocated.

The default: **2048** (as long as it does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free).

---

`-rcp`      `--recon_prio`

**Except RAID 0.**

Reconstruction priority in %.

Possible values are from **1** to **100** (maximum rate of reconstruction).

The default: **50**.

---

## Arguments for the `create` subcommand (continued)

<code>-rl</code>	<code>--request_limit</code>	Number of simultaneous I/O requests on RAID.  Possible values: from <b>0</b> (unlimited) to <b>4294967295</b> .  The <b>0</b> value disables the restriction.  The default: <b>0</b> .
<code>-rsp</code>	<code>--restripe_prio</code>	Restripping priority in %.  Possible values are from <b>1</b> to <b>100</b> (maximum rate of restripping).  The default: <b>100</b> .
<code>-sdcp</code>	<code>--sdc_prio</code>	SDC priority in %. The SDC priority cannot be set for RAIDs 0, 1, 10.  Possible values are from <b>1</b> to <b>100</b> (maximum rate of scanning).  The default: <b>50</b> .
<code>-se</code>	<code>--sched_enabled</code>	Enable ( <b>1</b> ) or disable ( <b>0</b> ) the scheduling function.  The default: <b>0</b> .
<code>-sp</code>	<code>--sparepool</code>	Name of the spare pool to assign to the RAID.
<code>-ss</code>	<code>--strip_size</code>	Strip size in KiB.  Possible values: <b>16</b> , <b>32</b> , <b>64</b> , <b>128</b> , or <b>256</b> .  The default: <b>16</b> .
<code>--force_metadata</code>		Allow overwriting of metadata on disks during the RAID creation process. Use this option only

## Arguments for the `create` subcommand (continued)

if you no longer need the data on the disks, as recovering data from a RAID without metadata can be extremely challenging.

---

`--trim`

This option runs the TRIM procedure on all disks in the RAID. However, the TRIM procedure will automatically run on RAID drives even without this option if the following conditions are met:

- All disks support TRIM.
- None of the disks contain any metadata. This check ensures that no data is accidentally deleted. If a disk contains metadata, performing the TRIM procedure will make it impossible to restore the data from that disk. This condition helps prevent unintentional data loss.
- The `--no_trim` option is not specified.

---

`--no_trim`

Use this option to prevent the TRIM procedure from being performed on disks in the RAID.

## destroy

Delete the RAID without possibility to restore the RAID and data on it.

```
# xicli raid destroy <arg>
```

## Arguments for the `destroy` subcommand

Mutually exclusive required arguments

---

`-n`

`--name`

The name of the RAID.

---

## Arguments for the `destroy` subcommand (continued)

`-a`            `--all`                    Delete all the xiRAID Classic RAIDs.

The argument takes no value.

## import apply

Import (or restore) the RAID from drive metadata.

```
# xicli raid import apply <arg> [optional_arg]
```

### Arguments for the `import apply` subcommand

Required argument

---

`-id`            `--uuid`                    UUID of the RAID.

---

Optional argument

---

`-nn`            `--new_name`                The new name for the RAID.

## import show

Show info about the RAIDs that can be imported (restored) from the drives.

```
# xicli raid import show [optional_args]
```

### Arguments for the `import show` subcommand

Optional arguments

---

`-d`            `--drives`                    The list of block devices (`/dev/sd*`, `/dev/mapper/mpath*`, `/dev/nvme*`, `/`

## Arguments for the `import show` subcommand (continued)

dev/dm-\*) separated by a space to show the info.

Without the argument, shows the info from all drives.

---

<code>-f</code>	<code>--format</code>	<p>Output format:</p> <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul> <p>The default: <b>table</b>.</p>
-----------------	-----------------------	---

---

<code>--offline</code>	<p>Show non-recoverable RAIDs in the import list.</p> <p>The argument takes no value.</p>
------------------------	---

## init start

Start or continue the RAID initialization.

```
# xicli raid init start <arg>
```

### Argument for the `init start` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## init stop

Stop the RAID initialization.

```
# xicli raid init stop <arg>
```

### Argument for the `init stop` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## init reset

Reset the initialization counter, set the state of the RAID to `need_init`, and start RAID reinitialization.

```
# xicli raid init reset <arg>
```

### Argument for the `init reset` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## modify

Modify the parameters of the created RAID.

```
# xicli raid modify <arg> [optional_args]
```

### Arguments for the `modify` subcommand

Required argument

---

## Arguments for the `modify` subcommand (continued)

<code>-n</code>	<code>--name</code>	The name of the RAID.  The maximum RAID name length is <b>28</b> characters.
-----------------	---------------------	--

---

### Optional arguments

---

<code>-am</code>	<code>--adaptive_merge</code>	<b>Except RAIDs 0, 1, 10.</b>  Enable ( <b>1</b> ) or disable ( <b>0</b> ) the Adaptive Merge write function.
------------------	-------------------------------	---

---

	<code>--single_run</code>	<b>Except RAIDs 0, 1, 10.</b>  Use this parameter to adjust <b>the Adaptive Merge values</b> once at start-up. After that, the values are set and do not change at system reboot. The Adaptive Merge write function is then turned off.  Does not take any value.  Can only be used with the <code>adaptive_merge</code> parameter.
--	---------------------------	---

---

<code>-ca</code>	<code>--cpu_allowed</code>	Change the CPUs on which the RAID will be allowed to run.  Possible values: a comma-separated list of CPUs, a range of CPUs indicated by a hyphen, or the value 'all' (the RAID will run on all available CPUs).  The default: <code>all</code> .
------------------	----------------------------	---

---

<code>-inp</code>	<code>--init_prio</code>	<b>Except RAID 0.</b>  Initialization priority in %.
-------------------	--------------------------	--

**Arguments for the `modify` subcommand (continued)**

Possible values are from **0** to **100**  
(maximum rate of initialization).

The default: **50**.

---

`-mwe`                    `--merge_write_enabled`    **Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge  
function for write operations.

The default: **0**.

---

`-mre`                    `--merge_read_enabled`    **Except RAIDs 0, 1, 10.**

Enable (**1**) or disable (**0**) the Merge  
function for read operations.

The default: **0**.

---

`-mrm`                    `--merge_read_max`        **Except RAIDs 0, 1, 10.**

Maximum wait time (in microsec-  
onds) for stripe accumulation with  
the Merge function enabled for read  
requests.

Possible values: integers from **1** to  
**100000**.

The default: **1000**.

---

`-mrw`                    `--merge_read_wait`       **Except RAIDs 0, 1, 10.**

Wait time (in microseconds) between  
read requests with the Merge func-  
tion enabled.

Possible values: integers from **1** to  
**100000**.

**Arguments for the `modify` subcommand (continued)**The default: **300**.

---

<code>-mwm</code>	<code>--merge_write_max</code>	<b>Except RAIDs 0, 1, 10.</b>
		Maximum wait time (in microseconds) for stripe accumulation with the Merge function enabled for write requests.
		Possible values: integers from <b>1</b> to <b>100000</b> .
		The default: <b>1000</b> .

---

<code>-mww</code>	<code>--merge_write_wait</code>	<b>Except RAIDs 0, 1, 10.</b>
		Wait time (in microseconds) between write requests with the Merge function enabled.
		Possible values: integers from <b>1</b> to <b>100000</b> .
		The default: <b>300</b> .

---

<code>-ml</code>	<code>--memory_limit</code>	RAM usage limit in MiB.
		Possible values: <b>0</b> and integers from <b>1024</b> to <b>1048576</b> .
		The <b>0</b> value sets unlimited RAM usage.
		The default: <b>0</b> (unlimited).

---

<code>-mp</code>	<code>--memory_prealloc</code>	The amount of reserved memory to allocate in MiB. This amount cannot exceed the RAM usage limit ( <code>--memory_limit</code> ). Additionally, it must be ensured that the specified amount
------------------	--------------------------------	---

## Arguments for the `modify` subcommand (continued)

does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free. If there is not enough available memory on the system, xiRAID Classic will attempt to change the allocated amount to its previous size. If this operation fails, an error message will be displayed, and the allocated amount will be set to **0**.

Possible values: **0** and integers from **1024** to **65536**.

Setting the value to **0** removes reserved memory for the specified RAID.

The default: **2048** (as long as it does not exceed the available system memory minus 1 GiB, leaving at least 1 GiB free).

---

`-rcp`                    `--recon_prio`

**Except RAID 0.**

Reconstruction priority in %.

Possible values: from **1** to **100** (maximum rate of reconstruction).

The default: **50**.

---

`-rl`                      `--request_limit`

Number of simultaneous I/O requests on RAID.

Possible values: integers from **0** to **4294967295**.

The **0** value disables the restriction.

The default: **0**.

---

**Arguments for the `modify` subcommand (continued)**

<code>-rsp</code>	<code>--restripe_prio</code>	<p>Restripping priority in %.</p> <p>Possible values are from <b>0</b> to <b>100</b> (maximum rate of restripping).</p> <p>The default: <b>100</b>.</p>
<code>-sdc</code>	<code>--sdc_prio</code>	<p>SDC priority in %.</p> <p>Possible values are from <b>1</b> to <b>100</b> (maximum rate of scanning).</p> <p>The default: <b>50</b>.</p>
<code>-se</code>	<code>--sched_enabled</code>	<p>Enable (<b>1</b>) or disable (<b>0</b>) the scheduling function.</p> <p>The default: <b>0</b>.</p>
<code>-sp</code>	<code>--sparepool</code>	<p>Name of the spare pool to assign to the RAID.</p> <p>The <b>null</b> value removes the spare pool from the RAID.</p> <p>Spare pool can not be assigned to RAID 0.</p>
	<code>--force_online</code>	<p>Change RAID state to online if the RAID has unrecoverable sections.</p> <p>I/O operations on unrecoverable sections may lead to data corruption.</p> <p>The argument takes no value.</p>
	<code>--force_resync</code>	<p>This parameter is deprecated and will be removed in a future release. Use the <b>raid init reset</b> command instead.</p>

## Arguments for the `modify` subcommand (continued)

**Except RAIDs 0, 1, 10.**

Force RAID re-initialization.

The argument takes no value.

## `recon start`

Start the RAID reconstruction.

```
# xicli raid recon start <arg>
```

### Argument for the `recon start` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## `recon stop`

Stop the RAID reconstruction.

```
# xicli raid recon stop <arg>
```

### Argument for the `recon stop` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## `replace`

Replace or remove the drive from the RAID.

```
# xicli raid replace <args>
```

## Arguments for the `replace` subcommand

### Required arguments

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-no</code>	<code>--number</code>	The number of the drive.  To find out the number of the drive, use <pre># xicli raid show</pre>
<code>-d</code>	<code>--drive</code>	The new block device.  To remove the drive (to mark it as missing) set the <b>null</b> value.

## resize

Change the RAID size.

```
# xicli raid resize <arg>
```

### Argument for the `resize` subcommand

#### Required argument

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## restore

Restore the RAID from the current configuration file.

```
# xicli raid restore <arg>
```

## Arguments for the `restore` subcommand

Mutually exclusive required arguments

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-a</code>	<code>--all</code>	Restore all available xiRAID Classic RAIDs.  Argument takes no value.

---

## restripe continue

Continue the RAID restripe.

```
# xicli raid restripe continue <arg>
```

## Argument for the `restripe continue` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## restripe start

Start the RAID restripe.

```
# xicli raid restripe start <args>
```

## Arguments for the `restripe start` subcommand

Required arguments

---

**Arguments for the `restripe start` subcommand (continued)**

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-l</code>	<code>--level</code>	The new level for the RAID.  If you are only increasing the RAID size, enter the current RAID level for this argument.
<code>-gs</code>	<code>--group_size</code>	<b>Only for RAIDs 50, 60, and 70.</b>  The new group size for the RAID.  Possible values: integers from <b>4</b> to <b>32</b> .
<code>-d</code>	<code>--drives</code>	The list of block devices ( <code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code> ) separated by a space to add to the RAID.

**restripe stop**

Pause the RAID restripe.

```
# xicli raid restripe stop <arg>
```

**Argument for the `restripe stop` subcommand**

Required argument

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

**show**

Show info about the RAID.

```
# xicli raid show [optional_args]
```

## Arguments for the `show` subcommand

### Optional arguments

---

`-n`      `--name`                      The name of the RAID.

Without the argument, show info on all xiRAID Classic RAIDs.

---

`-o`      `--online`                      Only show RAIDs that are in the “online” state (RAIDs that were not unloaded by the `raid unload` command and are not offline).

The argument takes no value.

---

`-u`      `--units`                      Dimension:

- **s** – sectors (1 sector=512 bytes);
- **k** – kilobytes;
- **m** – megabytes;
- **g** – gigabytes.

The default: **g**.

---

`-f`      `--format`                      Output format:

- **table**;
- **json**;
- **prettyjson** – human-readable json.

The default: **table**.

---

`-e`      `--extended`                      Show extended output.

The argument takes no value.

## unload

Remove (unload) the RAID with possibility to restore the RAID and save data on it.

```
# xicli raid unload <arg>
```

### Arguments for the `unload` subcommand

Mutually exclusive required arguments

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-a</code>	<code>--all</code>	Unload all available xiRAID Classic RAIDs.  The argument takes no value.

## settings

Operations with the additional settings of the `xicli` program.

Except for the `show` subcommand, the commands listed in this chapter require superuser privileges.

```
# xicli settings <subcommand> <args> [optional_args]
```

Subcommands for the `settings` command:

<code>auth modify</code>	Change client-server connection settings.
<code>auth show</code>	Show client-server connection settings.
<code>cluster modify</code>	Manage cluster settings.
<code>cluster show</code>	Show cluster settings.

eula modify	Manage the acceptance status of the EULA.
eula show	Show the acceptance status of the EULA.
faulty-count modify	Manage the threshold value of I/O errors for all drives.
faulty-count show	Show the threshold value of I/O errors.
log modify	Configure the type of system messages that will be added to the system log.
log show	Show the selected type of system messages for the system log.
mail modify	Manage email notification settings.
mail show	Show email notification settings.
pool modify	Manage delay timer (in seconds) for the drive replacement from the spare pools.
pool show	Show additional settings of the spare pools.
scanner modify	Manage RAID's monitoring, the LED indication and drive SMART settings.
scanner show	Manage the LED indication and drive scan settings.

## **auth modify**

Change client-server connection settings.



When you change any parameter of the `xicli settings auth modify` command, the `xiraid-target.service` restarts. Additionally, it will cause all RAIDs to unload. Please, run this command only after stopping all mounted devices.

```
# xicli settings auth modify <args>
```

## Arguments for the `auth modify` subcommand

At least one argument is required

---

`--host`

The host name or IP address that will be used for the connection.

After changing the host, you must regenerate and replace the certificate.

The default: **localhost**.

---

`--port`

The port that will be used for the connection.

The default: **6066**.

## `auth show`

Show client-server connection settings.

```
# xicli settings auth show
```

## Argument for the `auth show` subcommand

Optional argument

---

`-f`

`--format`

Output format:

## Argument for the `auth show` subcommand (continued)

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

## cluster modify

Manage xiRAID Classic cluster settings.

```
# xicli settings cluster modify <arg>
```

### Arguments for the `cluster modify` subcommand

At least one argument is required:

<code>-ra</code>	<code>--raid_autostart</code>	Activate ( <b>1</b> ) or deactivate ( <b>0</b> ) RAID autostart.
<code>-pa</code>	<code>--pool_autoactivate</code>	Activate ( <b>1</b> ) or deactivate ( <b>0</b> ) spare pool automatic activation.

## cluster show

Show xiRAID Classic cluster settings.

```
# xicli settings cluster show <arg>
```

### Argument for the `cluster show` subcommand

Optional argument

<code>-f</code>	<code>--format</code>	Output format:
-----------------	-----------------------	----------------

## Argument for the `cluster show` subcommand (continued)

- `table`;

- `json`;

- `prettyjson` – human-readable json.

The default: `table`.

## `eula modify`

Manage the acceptance status of the EULA.

```
# xicli settings eula modify
```

## Argument for the `eula modify` subcommand

Required argument

---

`-s`      `--status`      The status of the EULA acceptance.

Possible values: `accepted`, `not_accepted`.

## `eula show`

Show the acceptance status of the EULA.

```
# xicli settings eula show
```

## Argument for the `eula show` subcommand

Optional argument

---

`-f`      `--format`      Output format:

## Argument for the `eula show` subcommand (continued)

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

## faulty-count modify



When you change any parameter of the `xicli settings faulty-count modify` command, the `xiraid-scanner.service` restarts.

Manage the threshold value of I/O errors for all drives.

```
# xicli settings faulty-count modify <arg>
```

## Argument for the `faulty-count modify` subcommand

Required argument

---

<code>-t</code>	<code>--threshold</code>	<p>The threshold value for all drives.</p> <p>If you set a new fault threshold value, the current numbers of faults are reset for all the drives.</p> <p>Possible values: integers from <b>1</b> to <b>1000</b>.</p> <p>The default: <b>3</b>.</p>
-----------------	--------------------------	--

## faulty-count show

Show the threshold value of I/O errors.

```
# xicli settings faulty-count show
```

### Argument for the `faulty-count show` subcommand

Optional argument

---

`-f`

`--format`

Output format:

- **table**;
- **json**;
- **prettyjson** – human-readable json.

The default: **table**.

## log modify

Configure the type of system messages that will be added to the system log.

```
# xicli settings log modify <arg>
```

### Argument for the `log modify` subcommand

Required argument

---

`-l`

`--level`

The type of system messages that will be added to the system log.

Possible values: **error**, **warning**, **info**, **debug**.

Each next type includes the previous one.

## Argument for the `log modify` subcommand (continued)

The default: **info**.

## log show

Show the selected type of system messages for the system log.

```
# xicli settings log show [optional_arg]
```

## Argument for the `show` subcommand

Optional argument

---

<code>-l</code>	<code>--lines</code>	The number of error and warning messages in the event log to show, starting from the last entry.  Possible values: integers from <b>1</b> to <b>1000</b> .  The default: <b>10</b> .
-----------------	----------------------	--

## mail modify



When you change any parameter of the `xicli settings mail modify` command, the `xiraid-mail.service` restarts.

Manage email notification settings.

```
# xicli settings mail modify <args>
```

## Arguments for the `mail modify` subcommand

At least one argument is required

---

## Arguments for the `mail modify` subcommand (continued)

<code>-pi</code>	<code>--polling_interval</code>	<p>The polling interval for xiRAID Classic RAID's and the drives in seconds.</p> <p>Possible values: integers from <b>0</b> to <b>86400</b> (24 hours).</p> <p>The default: <b>10</b>.</p>
------------------	---------------------------------	--

---

<code>-ppi</code>	<code>--progress_polling_interval</code>	<p>Polling interval for the progress of initialization and reconstruction, in minutes.</p> <p>Possible values: integers from <b>0</b> to <b>1440</b> (24 hours).</p> <p>The default: <b>10</b>.</p>
-------------------	--	---

## mail show

Show email notification settings.

```
# xicli settings mail show
```

## Argument for the `mail show` subcommand

Optional argument

---

<code>-f</code>	<code>--format</code>	<p>Output format:</p> <ul style="list-style-type: none"> <li>• <b>table</b>;</li> <li>• <b>json</b>;</li> <li>• <b>prettyjson</b> – human-readable json.</li> </ul> <p>The default: <b>table</b>.</p>
-----------------	-----------------------	---

## pool modify



When you change any parameter of the `xicli settings pool modify` command, the `xiraid-scanner.service` restarts.

Manage delay timer (in seconds) for the drive replacement from the spare pools.

```
# xicli settings pool modify <arg>
```

### Argument for the `pool modify` subcommand

Required argument

---

<code>-rd</code>	<code>--replace_delay</code>	<p>Delay time (in seconds) for the drive replacement from the spare pools.</p> <p>Only one delay time is used for all the spare pools.</p> <p>Possible values: integers from <b>1</b> to <b>3600</b>.</p> <p>The default: <b>180</b>.</p>
------------------	------------------------------	---

## pool show

Show delay time used for the drive replacement from the spare pools.

```
# xicli settings pool show
```

### Argument for the `pool show` subcommand

Optional argument

---

<code>-f</code>	<code>--format</code>	Output format:
-----------------	-----------------------	----------------

## Argument for the `pool show` subcommand (continued)

- `table`;
- `json`;
- `prettyjson` – human-readable json.

The default: `table`.

## scanner modify



When you change any parameter of the `xicli settings scanner modify` command, the `xiraid-scanner.service` restarts.

Manage RAID's monitoring, the LED indication and drive SMART settings.

```
# xicli settings scanner modify <args>
```

## Arguments for the `scanner modify` subcommand

At least one argument is required

---

<code>-spi</code>	<code>--scanner_polling_interval</code>	<p>The polling interval for xiRAID Classic RAID's and drives in seconds.</p> <p>The parameter affects the auto-start delay for the RAID initialization, reconstruction, and restriping.</p> <p>Possible values: integers from <b>1</b> to <b>3600</b> (1 hour).</p> <p>The default: <b>1</b>.</p>
<hr/>		
<code>-spi</code>	<code>--smart_polling_interval</code>	S.M.A.R.T. drive health polling interval, in seconds.

## Arguments for the `scanner modify` subcommand (continued)

Possible values: integers from **60** to **86400** (24 hours).

The default: **86400**.

`-le`                    `--led_enabled`

Enable (**1**) or disable (**0**) the automatic LED indication of drives in the system.

The default: **1**.

The argument doesn't affect manual LED indication.

## scanner show

Show the LED indication and drive scan settings.

```
# xicli settings scanner show
```

## Argument for the `scanner show` subcommand

Optional argument

`-f`                    `--format`

Output format:

- **table**;
- **json**;
- **prettyjson** – human-readable json.

The default: **table**.

## sdc

Operations related to the SDC (Silent Data Corruption) scanner.

Subcommands for the `sdc` command:

<code>start</code>	Start an SDC scan.
<code>stop</code>	Pause an SDC scan.
<code>reset</code>	Stop an SDC scan and reset its progress.

## sdc start

Start or continue a Silent Data Corruption scan.

```
# xicli raid sdc start <arg>
```

### Argument for the `sdc start` subcommand

Required argument

<code>-n</code>	<code>--name</code>	The name of the RAID.
<code>-m</code>	<code>--mode</code>	Optionally, specify the scan mode: <ul style="list-style-type: none"><li><code>check</code> - Run a scan without fixing any inconsistencies found. SDC scans</li></ul>

## Argument for the `sdcli start` subcommand (continued)

run in this mode by default.

- `fixup` - Run a scan and resolve any inconsistencies.

## `sdcli stop`

Pause a Silent Data Corruption scan.

```
# xicli raid sdc stop <arg>
```

## Argument for the `sdcli stop` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## `sdcli reset`

Stop an SDC scan and reset its progress.

```
# xicli raid sdc reset <arg>
```

## Argument for the `sdcli reset` subcommand

Required argument

---

<code>-n</code>	<code>--name</code>	The name of the RAID.
-----------------	---------------------	-----------------------

## update

Operations with the Update Check service.

To check for an available update, run:

```
# xicli update check
```

Disable the Update Check Service to update xiRAID Classic 4.3.0 to a new available version.



Please, follow the instructions provided at [xinnor.io](https://xinnor.io) to safely update your xiRAID Classic. Ignoring these steps may result in filesystem panick and even data loss.

To disable the Update Check Service, run:

```
# xicli update prepare
```



Please, do not run this command unless there is a new available xiRAID Classic version. Otherwise, the proper functioning of xiRAID Classic cannot be guaranteed.



The Update Check Service will inform you of any detected mounted xiRAID devices. Please, unmount the devices before continuing the update process.