XINNOR

xiRAID Classic 4.1.0
Administrator's Guide

Contents

xiRAID Classic 4.1.0 Administrator's Guide 4

Introduction 4

About xiRAID Classic 4.1.0 5

Using xicli 6

Command-Line Interface (CLI) Overview 7

Accepting EULA 8

License 10

xiRAID RAIDs explained 12

RAID components 12

RAID levels 14

RAID lifecycle 17

xiRAID RAIDs administration 17

Creating the RAID 17

RAID Initialization 23

Showing RAID State 24

Managing RAID configuration 31

Deleting the RAID 45

Drives 46

RAID Reconstruction 57

CPU Management 58

xiRAID system administration 59

Scanning RAIDs and Drives, LED Indication 59

Configuration Files and Metadata 62

Update Management 74

Notifications 75

Using RAID in HA cluster 82

General Configuration Recommendations 82

RAID Creation 82

Using NVMe-oF devices to create a RAID 83

RAID and System Setup Recommendations....83
File System Mounting Examples....92
DKMS Specifics when Updating Linux Kernel....93
Authenticating gRPC Client....95
Troubleshooting....97

xiRAID Classic 4.1.0 Administrator's Guide

Instructions, practical guides, and tips for administering xiRAID Classic 4.1.0

Introduction

Intended Audience

This guide is intended for administrators and users of RAIDs based on the xiRAID Classic 4.1.0 software.

The guide contains instructions on how to configure and manage RAIDs in xiRAID Classic 4.1.0.

Guide Conventions

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

Convention	Uses
Bold	Documentation titles, section titles, GUI controls, option value, minor titles.
Italic	Emphasis, term references, file paths.
Text color	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 4.1.0

xiRAID Classic 4.1.0 is high-performance software RAID developed specifically for NVMe storage devices and new types of SAN networks. xiRAID Classic 4.1.0 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 4.1.0 is a Linux kernel module and a management utility, which is built and configured for the most popular distributions (see the xiRAID Classic 4.1.0 System Requirements document). The software is installed on servers with slots for Flash memory devices or with connected JBOFs. xiRAID Classic 4.1.0 enables you to combine drives into high-performance fault-tolerant RAIDs.

xiRAID Classic 4.1.0 Specifications

Supported RAID levels	• RAID 0 • RAID 1 • RAID 10 • RAID 5 • RAID 6 • RAID 7.3 • RAID 50 • RAID 60 • RAID 70 • RAID N+M
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.

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, 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 `xicli` 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 4.1.0.

Commands related to the EULA don't require running xiraid.target service.

To change the acceptance status of the EULA, run

```
# 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`**.

To view the acceptance status of the EULA, run

```
# xicli settings eula show
```

Argument for the `eula show` subcommand

Optional argument		
-f	--format	Output format: • **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. The default: **`table`**.

Accepting the EULA is not required after updating xiRAID Classic.

License

You can manage the license with the command

```
# 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.

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
```

Command output example when no license was added:

```
Kernel version: 5.14.0-362.24.1.el9_3.0.1.x86_64

hwkey: E83967DE5D549AA2
license_key: null
version: 0
crypto_version: 0
created: 0-0-0
expired: 0-0-0
disks: 4
levels: 0
type: nvme
disks_in_use: 4
status: trial
```

Command output example when a license was added:

```
Kernel version: 5.14.0-362.24.1.el9_3.0.1.x86_64

hwkey: E83967DE5D549AA2
license_key: BD26D9E40E3D06386AE53A30E74FA0037251804271E77B0
44ED8539B0CAA584E19E4C4703FDB9EE00FB609FD2EFC0F944B603775F20
87491D437A6538E3D11B779BF7C2CE90529FA5D23AB1A7A652760DF0B3EE
BEE7057BA30BDA73783E9C286C39E1DC3D24576112B7A778A34071C09811
E3CB6AFD1C5F495899DE7B82034BF
version: 1
crypto_version: 1
created: 2024-4-19
expired: 2026-4-19
disks: 64
levels: 70
type: nvme
disks_in_use: 4
status: valid
```

Description of the `license` command output

Kernel version	Kernel version.
hwkey	Hardware key.
license_key	License key.
version	License version.
crypto_version	Version of crypto-API for the license generator.
created	The date when the license was created.
expired	License expiration date.
disks	Maximum number of drives.
levels	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).
type	Drive type.

Description of the `license` command output (continued)

disks in use	Number of used drives in the system.
status	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.

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

```
# xicli license update -p </path/to/>license.txt
```

To check the applied license, run:

```
# xicli license show
```

xiRAID 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.

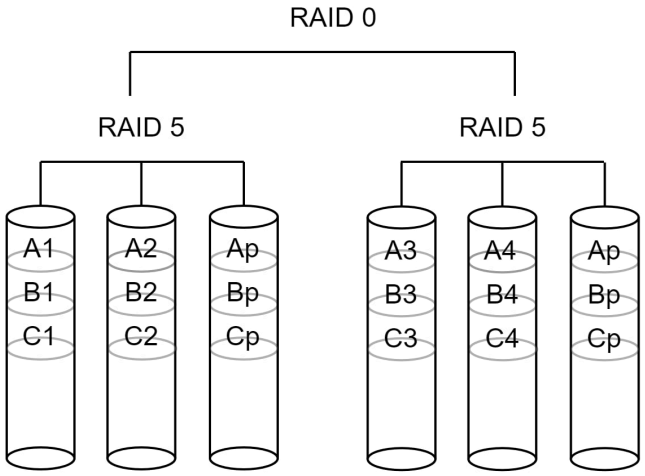
RAID strip size. In RAID striping, data is divided into strips. When data is written to a RAID level with striping, it is broken down into pieces (strips), and each piece is written to a single disk in the array (*the set of strips spanning across all the drives in that RAID set is called a stripe)*. 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, with 16 being the default size.

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 – RAIDs 10, 50, 60 and 70 (RAID 0 combination of several RAIDs 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 50

RAID levels

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

RAID level	Description	Features	Redundancy	Requirements
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 per-	This level offers the highest redundancy by creating a 1-for-1 copy of all data.	RAID 1 requires at least 2 drives.

(continued)

RAID level	Description	Features	Redundancy	Requirements
		formance is limited by the slowest drive.		
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, the capacity of three drives is allocated for checksums.	Sustains the complete failure of three drives.	RAID 7 requires at least 4 drives.
10	RAID 0, which components are RAIDs 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		Recoverable from 1 drive	At least 8 drives, the drive number must

(continued)

RAID level	Description	Features	Redundancy	Requirements
	across multiple RAIDs level 5.		failure in each group.	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 RAIDs level 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.
70	RAID 0 striping combination across multiple RAIDs 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.	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 <= 64 and M <= N.

(continued)

RAID level	Description	Features	Redundancy	Requirements
		The number of checksums can range from 4 to 32.		

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.

xiRAID RAIDs administration

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

Creating the 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		
-n	--name	The name of the RAID. The maximum RAID name length is **28** charaters.
-l	--level	The level of the RAID: **`0`**, **`1`**, **`5`**, **`6`**, **`7`**, **`10`**, **`50`**, **`60`**, **`70`**, or **`nm`**. Use the value **7** to create RAID 7.3.
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) separated by spaces.

Arguments for the `create` subcommand (continued)

-gs	--group_size	**Only for RAIDs 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 <= 64 and M <= 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. The default: **4096**.

Arguments for the `create` subcommand (continued)

-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 **0** to **100** (maximum rate of initialization). The default: **100**.
-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. Possible values: integers from **1** to **100000**.

Arguments for the `create` subcommand (continued)

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

Arguments for the `create` subcommand (continued)

		Reconstruction priority in %. Possible values are from **0** to **100** (maximum rate of reconstruction). The default: **100**.
-re	--resync_enabled	**Except RAIDs 0, 1, 10.** Enable (**1**) or disable (**0**) the Resync function. The default: **1**.
-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 **0** to **100** (maximum rate of restriping). The default: **100**.
-se	--sched_enabled	Enable (**1**) or disable (**0**) the scheduling function. The default: **0**.
-sp	--sparepool	Name of the spare pool to assign to the RAID.
-ss	--strip_size	Strip size in KiB. Possible values: **16**, **32**, **64**, **128**, or **256**.

Arguments for the `create` subcommand (continued)

		The default: **16**.
	--force_metadata	Allow overwriting metadata on disks.

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.

ⓘ The random write performance is higher on an initialized RAID.

To start or continue the RAID initialization, run

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

Argument for the `init start` subcommand

Required argument		
-n	--name	The name of the RAID.

To stop the RAID initialization, run

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

Argument for the `init stop` subcommand

Required argument		
-n	--name	The name of the RAID.

Example: Start initialization of the RAID "media5":

```
# xicli raid init start -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		
-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:

Arguments for the show subcommand (continued)

		• **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
```

RAIDs

name	static	state	devices	health	wear	serials	params	info
dimec	size: 29 GiB level: 5 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/sde online 1 /dev/sdc online 2 /dev/sdd online 3 /dev/sdf online	100% 100% 100% 100%	N/A N/A N/A N/A	drive-scsi12 drive-scsi10 drive-scsi9 drive-scsi8	init_prio : 100 recon_prio : 100 memory_limit_mb : 0 merge_read_enabled : 0 merge_write_enabled : 0 merge_read_wait_usecs : 300 merge_read_max_usecs : 1000 merge_write_wait_usecs: 300 merge_write_max_usecs : 1000 resync_enabled : 1 sched_enabled : 0 request_limit : 0 restripe_prio : 100 cpu_allowed : 0-7 adaptive_merge : False	memory_usage_mb : -

Description of the show subcommand output

Row	Description
name	RAID name.

Description of the show subcommand output (continued)

Row	Description
static	Static RAID parameters: • size. • level. • synd_cnt – only for RAIDs N+M – number of syndromes. • block_size – RAID block size. • group_size – only for RAIDs 10, 50, 60, and 70 – size of the corresponding RAID group. • strip_size. • sparepool – name of the assigned spare pool. • active: ◦ True, if the RAID's block device is in the system. ◦ False, if: ▪ The RAID was not loaded after reboot. ▪ The RAID is unloaded. • config: ◦ True, if the RAID is in the configuration file. ◦ False, if the RAID is missing.
state	RAID state:

Description of the show subcommand output (continued)

Row	Description
	• online – the RAID is available and ready to work. • initialized – initialization is finished. • initing – the RAID is initializing. • degraded – the RAID is available and ready for work but some drives are missing or failed. • reconstructing – the RAID is reconstructing. • 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. • need_recon – the RAID needs reconstruction. • need_init – the RAID needs initialization. • read_only – the license has expired. The RAID is read-only. • unrecovered – RAID can't complete reconstruction because of unrecoverable sections. • none – RAID was unloaded via the unload command or was not restored after reboot. • restriping – RAID is restriping. • need_resize – restriping was finished, the RAID size increase is available. • need_restripe – restriping was stopped and not finished.
devices	The list of devices included in the RAID, and their current states:

Description of the show subcommand output (continued)

Row	Description
	• online – the drive is active. • offline – the drive is missing or unavailable. • reconstructing – the drive is in process of reconstruction. • need_recon – the drive needs reconstruction. For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers. If the RAID is in 'restriping' or 'need-resize' state, the devices involved in the resize operation are highlighted in yellow.
health	To show, use the command with the -e parameter. Percent of valid drive data. When health is 100% – no reconstruction required. For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.
wear	To show, use the command with the -e parameter. The wear percentage of the SSD or NVMe drive. When the drive reaches the 90% threshold, the system sends an error message to the mail. The S.M.A.R.T. values "Percentage used endurance indicator" and "Percentage Used" are used to check SSD and NVMe drives respectively. For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.
serials	To show, use the command with the -e parameter.

Description of the show subcommand output (continued)

Row	Description
	Serial numbers of drives in RAID. For RAIDs 10, 50, 60, and 70, the devices are grouped according to their respective group numbers.
params	To show, use the command with the -e parameter. Editable RAID parameters:

Description of the show subcommand output (continued)

Row	Description
	• init_prio – (except RAID 0) initialization priority: from 0% to 100%. • recon_prio – (except RAID 0) reconstruction priority: from 0% to 100%. • memory_limit_mb – the value limited RAM usage, in megabytes. • merge_read_enabled – is the function Merge enabled (1) or disabled (0) for read operations. • merge_write_enabled – is the function Merge enabled (1) or disabled (0) for write operations. • merge_read_wait_usecs – waiting time between read requests when Merge is enabled. • merge_read_max_usecs – maximum time to wait for read requests accumulation with Merge enabled. • merge_write_wait_usecs – waiting time between write requests when Merge is enabled. • merge_write_max_usecs – maximum time to wait for write requests accumulation with Merge enabled. • resync_enabled – is the function Resync enabled (1) or disabled (0). • sched_enabled – is the function Scheduling enabled (1) or disabled (0). • request_limit – number of simultaneous I/O requests on RAID (0 for no limit). • restripe_prio – priority for restriping: from 0% to 100%. • cpu_allowed - the CPUs on which the RAID is allowed to run. • cpu_allowed_configured - the CPUs on which the RAID is allowed to run as specified in the configuration file. This parameter appears when the *cpu_allowed* parameter exceeds the number of available

Description of the show subcommand output (continued)

Row	Description
	CPUs in the system. Only available in High Availability configuration. • adaptive_merge - is the function Adaptive Merge enabled (True) or disabled (False).
info	Dynamic RAID values: • init_progress – initialization progress: from 0% to 100%. • recon_progress – reconstruction progress: from 0% to 100%. • memory_usage_mb – amount of RAM usage; if memory_limit_mb = 0 (not limited), then memory_usage_mb is displayed as '-'. • restripe_progress – restriping progress: from 0% to 100%.

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 4.1.0. Please refer to the "xiRAID Classic 4.1.0 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 Restriping 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 restriping by changing the corresponding RAID parameter.
- RAID state must not be one of the following:
  - offline;
  - need_restripe;
  - restriping;
  - 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)

Current level	New level	Requirements	Minimal number of drives you should add
	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		2
	RAID 10		2
	RAID 5		2
RAID 10	RAID 10		4
	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)

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

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

raid restripe

The chapter describes the available subcommands for the restripe operation.

Restriping 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 restriping, run

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

Arguments for the `restripe start` subcommand

Required arguments

Arguments for the `restripe start` subcommand (continued)

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

-n	--name	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 4.1.0. Please refer to the "xiRAID Classic 4.1.0 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		
-n	--name	The name of the RAID. The maximum RAID name length is **28** charaters.
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 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: **100**.
-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`**.
-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 `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).
-rcp	--recon_prio	**Except RAID 0.** Reconstruction priority in %. Possible values: from **0** to **100** (maximum rate of reconstruction). The default: **100**.
-re	--resync_enabled	**Except RAIDs 0, 1, 10.** Enable (**1**) or disable (**0**) the resync function. The default: **1**.
-rl	--request_limit	Number of simultaneous I/O requests on RAID. Possible values: integers from **0** to **4294967295**.

Arguments for the `modify` subcommand (continued)

		The **0** value disables the restriction. The default: **0**.
-rsp	--restripe_prio	Restriping priority in %. Possible values are from **0** to **100** (maximum rate of restriping). The default: **100**.
-se	--sched_enabled	Enable (**1**) or disable (**0**) the scheduling function. The default: **0**.
-sp	--sparepool	Name of the spare pool to assign to the RAID. The **null** value removes the spare pool from the RAID. Spare pool can not be assigned to RAID 0.
	--force_online	Change RAID state to online if the RAID has unrecoverable sections. I/O operations on unrecoverable sections may lead to data corruption. The argument takes no value.
	--force_resync	**Except RAIDs 0, 1, 10.** Force RAID re-initialization. The argument takes no value.

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

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

Deleting the 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		
-n	--name	The name of the RAID.
-a	--all	Delete all the xiRAID Classic RAIDs. The argument takes no value.
Optional arguments (only applied with the --all argument)		
	--force	Force the command execution.
	--config_only	Remove RAID only from config.

Example: Deleting the RAID "media5":

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

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		
-n	--name	The name of the RAID.
-a	--all	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.

Drives

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		
-n	--name	The name of the RAID.
-no	--number	The number of the drive. To find out the number of the drive, use `# xicli raid show`
-d	--drive	The new block device. To remove the drive (to mark it as missing) set the **`null`** 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 on 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. We recommend creating a sparepool with storage devices of the same type.

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 add drive(s) to the spare pool, run

```
# 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.

To create the spare pool, run

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

Arguments for the `create` subcommand

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

To delete the spare pool, run

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

Argument for the `delete` subcommand

Required argument		
-n	--name	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		
-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.

To show info on the spare pool, run

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

Arguments for the show subcommand

Optional arguments		
-n	--name	The name of the spare pool. Without the argument, show info on all spare pools.
-f	--format	Output format: • **table**; • **json**; • **prettyjson** – human-readable json. The default: **table**.
-u	--units	Size units: • **s** – sectors (1 sector=512 bytes); • **k** – kilobytes; • **m** – megabytes; • **g** – gigabytes. The default: **g**.

SparePools

name	devices	serials	sizes
pool1	0 /dev/sda ready 1 /dev/sdb ready	drive-scsi0 drive-scsi1	5 GiB 5 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.

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.

Argument for the `pool modify` subcommand

Required argument		
-rd	--replace_delay	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 **1** to **3600**. The default: **180**.

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		
-f	--format	Output format: • **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. The default: **`table`**.

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		
-t	--threshold	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 **1** to **1000**. The default: **3**.

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		
-f	--format	Output format: • **`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		
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) 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		
-n	--name	The RAID name for which drives the current number of faults will be shown. If neither of the two arguments is specified, show the values for all drives.
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) separated by a space to show their current numbers of faults. If neither of the two arguments is specified, show the values for all drives.
Optional argument		
-f	--format	Output format: • **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. 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		
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) 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		
-n	--name	The name of the RAID.

To start the RAID reconstruction, run

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

Argument for the `recon start` subcommand

Required argument		
-n	--name	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: `all`.

xiRAID 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

-spi	--scanner_polling_interval	The polling interval for xiRAID Classic RAIDs and drives in seconds. The parameter affects the auto-start delay for the RAID initialization, reconstruction, and restriping. Possible values: integers from **1** to **3600** (1 hour). The default: **1**.
-spi	--smart_polling_interval	S.M.A.R.T. drive health polling interval, in seconds. 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.

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

xiRAID Classic System and RAID Configuration

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

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

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.

The common configurations file

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

The example of the configuration file:

```
{
    "raid_autostart": 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
}
```


The "raid autostart" object indicates whether RAID autostart is enabled (1) or disabled (0) at the module loading. RAID autostart is disabled for High Availability configuration.

The "drives" object contains the serial numbers of the drives with an error count greater than 0 and their corresponding numbers of errors.

The "faulty_count_threshold" object contains the value of the error threshold value for the drives.

The "scanner_polling_interval" object contains the value of polling interval for xiRAID RAIDs and drives in seconds.

The "smart_polling_interval" object contains the value of S.M.A.R.T. drive health polling interval in seconds.

The "led_enabled" object indicates whether the automatic LED indication of drives in the system is enabled.

The "timestamp" object contains the creation date of this configuration file in timestamp format.

The 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": "test",
    "uuid": "F4144567-133C-421F-8898-CC11CCC6AF6C",
    "level": "5",
    "synd_cnt": 1,
    "strip_size": 16,
    "block_size": 4096,
    "drives": [
        "drive-scsi2",
        "drive-scsi3",
        "drive-scsi4",
        "drive-scsi6"
    ],
    "size": 30296064,
    "group_size": 4,
    "auto_init": true,
    "auto_recon": true,
    "auto_restripe": true,
    "cpu_allowed": []
    "timestamp": 1702916010.8023913
}
```


Configurations 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.

Importing 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	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) separated by a space to show the info. Without the argument, shows the info from all drives.
-f	--format	Output format:

Arguments for the `import show` subcommand (continued)

	• **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. The default: **`table`**.
--offline	Show non-recoverable RAIDs in the import list. 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 RAIDs that are available for import, run:

```
# xicli raid import show
```

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

RAIDs

uuid	info	devices	serials	import status
073DE372-94D0-43B3-9582-BFC67F3824CC	size: 9 GiB level: 0 synd_cnt: 0 strip_size: 16 block_size: 4096	0 /dev/sdl normal 1 /dev/sdm normal	drive-scsi11 drive-scsi12	name: OK drives: OK
61BF2210-6E1B-4F76-9F01-02D297E1C715	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-scsi7 drive-scsi8 drive-scsi9 drive-scsi10	name: Conflict with in-system RAID(s) drives: Conflict with in-system RAID(s) media1
D698D77D-1B10-48D6-95FE-B037E7B0C366	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-scsi1 drive-scsi2 null null drive-scsi5 drive-scsi6	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 RAIDs. The second RAID conflicts with in-system and import RAIDs 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		
-id	--uuid	UUID of the RAID.
Optional argument		
-nn	--new_name	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 disks 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

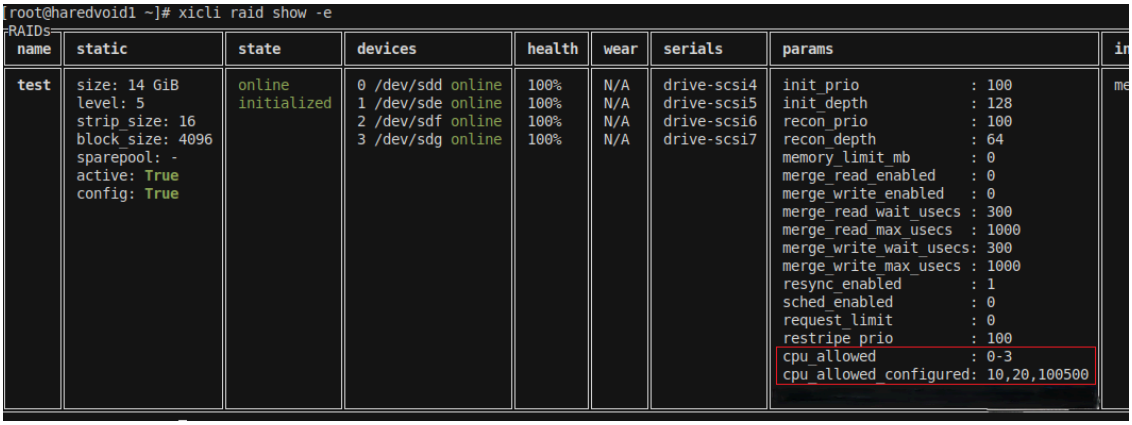
The number of CPUs specified for a RAID is stored in the RAID configuration file. However, when importing a RAID to a system or migrating it to another node in an HA cluster, it is important to note that the number of available CPUs may differ.

> 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 numbers (<CPU_numbers>) are greater than the allowed CPU number: <number_of_CPUs>. Ignoring 'cpu allowed' parameter."`

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.



'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 *backup_raid.conf* 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 all RAIDs in the system whose configuration files are present in the directory `/etc/xiraid/raids/*`, and then restores all RAIDs that are possible to restore (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	--common_file	A file to restore or replace the common configuration file `/etc/xiraid/raid.conf`.

Arguments for the `restore` subcommand (continued)

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

config show

To show configuration files stored on the RAID disks, run

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

Argument for the `show` subcommand

Optional argument		
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) 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.

1. Restore the most recent configuration amond all drives. At this stage, the `/etc/xiraid/raids.drive/` folder appears, containing the configuration files for different RAIDs 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
   ```

> While applying the configuration files, all RAIDs on the system will be recreated (unloaded and restored). Be careful if you have mounted devices on the system. Before executing 'xicli config apply', unmount all xiRAID RAIDs.

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 the 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, RAIDs 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, it will be restored to the 'offline' state. In such case, contact the xiRAID Classic support team at support@xinnor.io.
- If none of the drives specified in a RAID configuration are available in the host system, the RAID will not be restored. In such case, contact the xiRAID Classic support team at 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'.
```

You can restore RAIDs manually from the configuration files using

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

Arguments for the `restore` subcommand

Mutually exclusive required arguments		
-n	--name	The name of the RAID.

Arguments for the `restore` subcommand (continued)

-a	--all	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 4.1.0 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 4.1.0 to a new available version.

Please, follow the instructions provided at 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 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		
-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. The default: **`debug`**.

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

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:

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

To set the receiver's email and the notification level, run

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

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

Arguments for the add subcommand

Required arguments		
-a	--address	Receiver's email.
-l	--level	The notification level. Possible values: • **`info`** – Info notifications; • **`warning`** – Error and Warning notifications; • **`error`** – Error notifications.

Notification types:

Info	Warning	Error
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 reboot/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

Notification types: (continued)

Info	Warning	Error
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: number of disks in use exceeds the allowed disks number
Drive (...) was returned to RAID (...)		Drive (...) in RAID (...) is offline now
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

Notification types: (continued)

Info	Warning	Error
		The bdev (...) has critical wear out (...)%

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

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

To remove the email from the list of email notifications, run

```
# 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		
-a	--address	The email address to remove from the notifications.

To show the list of the email notifications, run

```
# xicli mail show
```

Argument for the `mail show` subcommand

Optional argument		
-f	--format	Output format:

Argument for the `mail show` subcommand (continued)

	• **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. The default: **`table`**.

To manage email notification settings, run

```
# 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		
-pi	--polling_interval	The polling interval for xiRAID Classic RAIDs and the drives in seconds. Possible values: integers from **`0`** to **`86400`** (24 hours). The default: **`10`**.
-ppi	--progress_polling_interval	Polling interval for the progress of initialization and reconstruction, in minutes. Possible values: integers from **`0`** to **`1440`** (24 hours). The default: **`10`**.

To show email notification settings, run

```
# xicli settings mail show
```

Argument for the `mail show` subcommand

Optional argument		
-f	--format	Output format: • **`table`**; • **`json`**; • **`prettyjson`** – human-readable json. The default: **`table`**.

Using RAID in HA cluster

xiRAID Classic RAID objects can be used in HA cluster configuration. To learn more, see Integrating xiRAID Classic 4.1.0 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.

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 **100**, 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. If the initialization priority is set to **0**, initialization requests are not created during user I/O.

--recon_prio

The reconstruction process starts automatically. By default, reconstruction priority equals to **100**, which means reconstruction has maximum priority among other processes. Setting the priority to **0** allows the user I/O processes running before the reconstruction process.

--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

This parameter limits the number of incoming requests per RAID. For example, writing files with a file system without synchronization.

To improve system workload performance, we recommend enabling the limit on the number of incoming requests by the --request_limit parameter when you are working with file system and the buffered writing is performed.

--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 before using the command.

--resync_enabled

The function starts a RAID re-initialization after an unsafe system shutdown, thereby protecting syndromic RAIDs (all but RAID 0, RAID 1, and RAID 10) from data loss caused by a write hole.

To disable resync for a RAID, run

```
# xicli raid modify -n <name> --resync_enabled=0
```

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 [1] 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_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.

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
```

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.*

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

The function only works when the 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 *minus* 1
RAID 6	Number of RAID drives *minus* 2
RAID 7.3	Number of RAID drives *minus* 3
RAID N+M	Number of RAID drives *minus* M
RAID 50	Number of drives in one RAID group *minus* 1
RAID 60	Number of drives in one RAID group *minus* 2
RAID 70	Number of drives in one RAID group *minus* 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 4.1.0 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 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

Strip size, in KiB	Amount of RAM, in MiB
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 3dnow-prefetch 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 4.1.0, 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 **--shed_enabled** section.

Swap File

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

File System Mounting Examples

Since the system restores xiRAID Classic RAIDs after loading an appropriate Linux core and sending a RAID-restore command, to perform automatic mounting at system startup of file systems for these RAIDs, use one of the following instructions.

> ⓘ To set up automatic mounting at system startup, we recommend using systemd.mount.

systemd.mount

Example: mounting xfs located on a RAID /dev/xi_*raidname* into /mnt/raid/ through systemd.mount:

1. Create a file at `/etc/systemd/system/` with the mount options.

   The file name must match the path of the mount directory with "/" replaced by "-" (for example, for `/mnt/raid` the file name will be "mnt-raid.mount").

   The example file `/etc/systemd/system/mnt-raid.mount`

```
[Unit]

Description=Mount filesystem on xiRAID Classic

DefaultDependencies=no

Before=umount.target

Conflicts=umount.target



[Mount]

What=/dev/xi_raidname
```

```
Where=/mnt/raid/

Options=defaults

Type=xfs


[Install]

WantedBy=multi-user.target
```

2. Run the command

   ```
   # systemctl daemon-reload
   ```

   Enable automatic mounting at system startup:

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

   Start the service to mount the file system:

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

/etc/fstab

> ⚠ Make sure to only use the block device name, such as `/dev/xi_raidname`, and not the UUID or LABEL.

Example: mounting xfs located on a RAID `/dev/xi_raidname` into `/mnt/raid/` through `/etc/fstab`:

The string from the file `/etc/fstab`

```
/dev/xi_raidname    /mnt/raid/   xfs   defaults  0   0
```

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 4.1.0 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 4.1.0 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
```

Authenticating gRPC Client

To authenticate a gRPC client, set up a host and/or 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
```

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 4.1.0 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.