

Xinnor xiRAID 4.0.0 User Documentation

Contents

Xinnor xiRAID 4.0.0 Administrator's Guide.....	3
Introduction.....	3
Intended Audience.....	3
Guide Conventions.....	3
About Xinnor xiRAID 4.0.0.....	3
Xinnor xiRAID 4.0.0 Specifications.....	4
Using xicli.....	4
Command-Line Interface (CLI) Overview.....	4
Accepting EULA.....	5
License.....	6
RAID.....	8
Drives.....	24
Configuration Files and Metadata.....	31
Scanning RAIDs and Drives, LED Indication.....	37
Notifications.....	39
Number of CPU Threads of the xiraid Module.....	43
General Configuration Recommendations.....	44
RAID Creation.....	44
RAID and System Setup Recommendations.....	45
File System Mounting Aspects.....	49
DKMS Specifics when Updating Linux Kernel.....	51
Authenticating gRPC Client.....	52
Troubleshooting.....	53

Xinnor xiRAID 4.0.0 Administrator's Guide

Instructions, practical guides, and tips for administering Xinnor xiRAID 4.0.0

Introduction

Intended Audience

This guide is intended for administrators and users of RAIDs based on the Xinnor xiRAID 4.0.0 software.

The guide contains instructions on how to configure and manage RAIDs in Xinnor xiRAID 4.0.0.

Guide Conventions

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

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

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

Tip:

Note – a note, which provides valuable information.



Attention:

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

About Xinnor xiRAID 4.0.0

Xinnor xiRAID 4.0.0 is high-performance software RAID developed specifically for NVMe storage devices and new types of SAN networks. Xinnor xiRAID 4.0.0 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.

Xinnor xiRAID 4.0.0 is a Linux kernel module and a management utility, which are built and configured for the most popular distributions (see the Xinnor xiRAID 4.0.0 System Requirements document). The software is installed on servers with slots for Flash memory devices or with connected JBOFs. Xinnor xiRAID 4.0.0 enables you to combine drives into high-performance fault-tolerant RAID.

Xinnor xiRAID 4.0.0 Specifications

Supported RAID levels	<ul style="list-style-type: none"> • 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 100 MiB and the last 100 MiB of each drive in a RAID.

Using xicli

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

The `managing` program is still available, but it does not include all the commands that are available through `xicli` (see the Xinnor xiRAID 4.0.0 Command Reference for details on the differences between `managing` and `xicli` commands). The `managing` program won't be supported in the future releases of the Xinnor xiRAID.

We have added “EULA” and “Restripping for RAIDs 50, 60, and 70” features to the `managing` commands. To learn more, see the corresponding chapters in this document.

Command-Line Interface (CLI) Overview

Table 1: Conventions on CLI command syntax

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

Table 2: The list of available commands <command>

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

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

Accepting EULA

The first time after the installation you run any `xicli` command (except `settings eula modify`, `settings eula show`), you will be prompted to accept the EULA.

After accepting the EULA, the ran command executes, and you can use Xinnor xiRAID 4.0.0.

Tip:

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

Tip:

This feature has been added to the `config` commands. To learn more, see

```
# eula -h
```

To change the acceptance status of the EULA, run

```
# xicli settings eula modify
```

Table 3: Argument for the `eula modify` subcommand

Required argument

<code>-s</code>	<code>--status</code>	The status of the EULA acceptance. Possible values: <code>accepted</code> , <code>not_accepted</code> .
-----------------	-----------------------	---

To view the acceptance status of the EULA, run

```
# xicli settings eula show
```

Table 4: Argument for the `eula show` subcommand

Optional argument

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

License

You can manage the license with the command

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

Subcommands for the `license` command:

<code>delete</code>	Delete the current license.
<code>show</code>	Show info on the current license.
<code>update</code>	Update the current license.

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

```
# xicli license show
```

Command output example when no license was added:

```
Kernel version: 3.10.0-957.27.2.el7.x86_64
hwkey: 2EE10A9F12DD2662
license_key: null
version: 0
crypto_version: 0
created: 0-0-0
expired: 0-0-0
disks: 0
levels: 0
type: nvme
disks_in_use: 0
status: expired
```

Command output example when a license was added:

```
Kernel version: 3.10.0-957.27.2.el7.x86_64
hwkey: 2EE10A9F12DD2662
license_key: E3C6C0C0EEABE4274DBACACAE8D31E
76C2FB30CF18A3EB832BC007BC0B32DC4CDF9D4C135
D78344C19DF3E0D9AA995F64C4E0AFA9A441A51292D
version: 1
crypto_version: 0
created: 2019-4-14
expired: 2020-12-31
disks: 128
levels: 60
type: nvme
disks_in_use: 0
status: valid
```

Table 5: Description of the license command output

Kernel version	Kernel version.
hwkey	Hardware key.
license_key	License key.
version	Software 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.

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
```

Tip:

Additional actions may be required to apply the license for OS Debian 9. To learn more, see the [Troubleshooting](#) chapter.

To check the applied license, run:

```
# xicli license show
```

RAID

In this chapter you will learn how to create and manage xiRAID 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.



Attention:

Creating xiRAID RAID over xiRAID RAID devices is not recommended. 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, or 7 – at least 4 drives;
- of level 10 – at least 2 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, or 70 – at least 8 drives (make sure the total drives number is multiple of the --group-size parameter value);
- of level N+M – at least 8 drives.

RAM is limited from 1024 MiB to the maximum system capacity.

Table 6: Arguments for the create subcommand

Required arguments

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

-l	--level	The level of the RAID: 0, 1, 5, 6, 7, 10, 50, 60, 70, or nm.
-d	--drives	The list of block devices (/dev/sd*, /dev/mapper/mpath*, /dev/nvme*, /dev/dm-*) separated by a space.
-gs	--group_size	Only for RAID5, 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 RAID5 N+M. The number of syndromes M. Possible values are integers from 4 to 32. Additional conditions: $N+M \leq 64$ and $M \leq N$.
Optional arguments		
-bs	--block_size	RAID block size: 512 or 4096 bytes. The default: 4096.
-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 RAID5 0, 1, 10. Enable (1) or disable (0) the Merge function for write operations. The default: 0.
-mre	--merge_read_enabled	Except RAID5 0, 1, 10. Enable (1) or disable (0) the Merge function for read operations. The default: 0.

-ml	--memory_limit	<p>RAM usage limit in MiB.</p> <p>Possible values: 0 and integers from 1024 to 1048576.</p> <p>The 0 value sets unlimited RAM usage.</p> <p>The default: 0.</p>
-mm	--merge_max	<p>Except RAIDs 0, 1, 10.</p> <p>Maximum wait time (in microseconds) for stripe accumulation for the Merge functions.</p> <p>Possible values: integers from 1 to 100000.</p> <p>The default: 1000.</p>
-mw	--merge_wait	<p>Except RAIDs 0, 1, 10.</p> <p>Wait time (in microseconds) between requests for the Merge functions.</p> <p>Possible values: integers from 1 to 100000.</p> <p>The value must be less than the merge_max value.</p> <p>The default: 300.</p>
-rcp	--recon_prio	<p>Except RAID 0.</p> <p>Reconstruction priority in %.</p> <p>Possible values are from 0 to 100 (maximum rate of reconstruction).</p> <p>The default: 100.</p>
-re	--resync_enabled	<p>Except RAIDs 0, 1, 10.</p> <p>Enable (1) or disable (0) the Resync function.</p> <p>The default: 1.</p>
-rl	--request_limit	<p>Number of simultaneous I/O requests on RAID.</p> <p>Possible values: from 0 (unlimited) to 4294967295.</p> <p>The 0 value disables the restriction.</p> <p>The default: 0.</p>

-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. The default: 16.

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
```

Showing RAID State

You can view info about the RAID with the command

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

Table 7: Arguments for the show subcommand

Optional arguments

-n	--name	The name of the RAID. Without the argument, show info on all xiRAID RAIDs.
-o	--online	Only show RAIDs that are in the "online" state. The argument takes no value.
-u	--utils	Dimension: <ul style="list-style-type: none"> • s – sectors (1 sector=512 bytes); • k – kilobytes; • m – megabytes; • g – gigabytes. The default: g.

<code>-f</code>	<code>--format</code>	Output format:
		<ul style="list-style-type: none"> • <code>table</code>; • <code>json</code>; • <code>prettyjson</code> – human-readable json.
		The default: <code>table</code> .
<code>-e</code>	<code>--extended</code>	Show extended output.
		The argument takes no value.

Example: Show information on the "media5" RAID:

```
# xicli raid show -n media5 -e
```

RAID name	static	state	devices	health	wear	serials	params	info
media1	size: 4 GiB level: 1 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/sdh online 1 /dev/sdd online	100% 100%	N/A N/A	drive-scsi7 drive-scsi3	init_prio : 100 init_depth : 128 recon_prio : 100 recon_depth : 64 memory_limit_mb : 0 sched_enabled : 0 request_limit : 0 restripe_prio : 100	memory_usage_mb : -

Table 8: Description of the show subcommand output

Row	Description
name	RAID name.

Row	Description
static	<p>Static RAID parameters:</p> <ul style="list-style-type: none"> • size. • level. • synd_cnt – only for RAID5s and RAID6s – number of syndromes. • block_size – RAID block size. • group_size – only for RAID50, RAID60, and RAID70 – size of the corresponding RAID group. • strip_size. • sparepool – name of the assigned spare pool. • active: <ul style="list-style-type: none"> • True, if the RAID's block device is in the system. • False, if: <ul style="list-style-type: none"> • The RAID was not loaded after reboot. • The RAID is unloaded. • config: <ul style="list-style-type: none"> • True, if the RAID is in the configuration file. • False, if the RAID is missing.
state	<p>RAID state:</p> <ul style="list-style-type: none"> • 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. • 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 <code>unload</code> 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.

Row	Description
devices	<p>The list of devices included in the RAID, and their current states:</p> <ul style="list-style-type: none">• 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.
health	<p>To show, use the command with the -e parameter.</p> <p>Percent of valid drive data.</p> <p>When health is 100% – no reconstruction required.</p>
wear	<p>To show, use the command with the -e parameter.</p> <p>The wear percentage of the SSD or NVMe drive.</p> <p>When the drive reaches the 90% threshold, the system sends an error message to the mail.</p> <p>The S.M.A.R.T. values "Percentage used endurance indicator" and "Percentage Used" are used to check SSD and NVMe drives respectively.</p>
serials	<p>To show, use the command with the -e parameter.</p> <p>Serial numbers of drives in RAID.</p>

Row	Description
params	<p>To show, use the command with the <code>-e</code> parameter.</p> <p>Editable RAID parameters:</p> <ul style="list-style-type: none"> • <code>init_prio</code> – (except RAID 0) initialization priority: from 0% to 100%. • <code>init_depth</code> – the queue depth of initialization requests. • <code>recon_prio</code> – (except RAID 0) reconstruction priority: from 0% to 100%. • <code>recon_depth</code> – the queue depth of reconstruction requests. • <code>memory_limit_mb</code> – the value limited RAM usage, in megabytes. • <code>merge_write_enabled</code> – is the function Merge enabled (1) or disabled (0) for write operations. • <code>merge_read_enabled</code> – is the function Merge enabled (1) or disabled (0) for read operations. • <code>resync_enabled</code> – is the function Resync enabled (1) or disabled (0). • <code>sched_enabled</code> – is the function Scheduling enabled (1) or disabled (0). • <code>request_limit</code> – number of simultaneous I/O requests on RAID (0 for no limit). • <code>restripe_prio</code> – priority for restriping: from 0% to 100%. • <code>merge_wait_usecs</code> – waiting time between requests when Merge is enabled. • <code>merge_max_usecs</code> – maximum time to wait for stripe accumulation with Merge enabled.
info	<p>Dynamic RAID values:</p> <ul style="list-style-type: none"> • <code>init_progress</code> – initialization progress: from 0% to 100%. • <code>recon_progress</code> – reconstruction progress: from 0% to 100%. • <code>memory_usage_mb</code> – amount of RAM usage; if <code>memory_limit_mb = 0</code> (not limited), then <code>memory_usage_mb</code> is not displayed. • <code>restripe_progress</code> – restriping progress: from 0% to 100%.

Deleting the RAID



Attention:

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>
```

Table 9: Arguments for the `destroy` subcommand

Mutually exclusive required arguments

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

Example: Delete 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>
```

Table 10: Arguments for the `unload` subcommand

Mutually exclusive required arguments

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

Example: Unloading the RAID "media5":

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

To restore unloaded RAIDs, run:

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

RAID Reconstruction

Reconstruction of a RAID (except RAID 0) will start automatically when you programmatically replace a drive in the RAID. While a RAID is being reconstructed, the functions initialization (or resync) and restriping are paused.

Tip:To improve the system performance under the workload, try [decreasing reconstruction priority](#) by changing the corresponding RAID parameter.

To start the RAID reconstruction, run

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

Table 11: Argument for the `recon start` subcommand

Required argument

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).

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).

For an operation to increase the size of the RAID through larger disks, only a resize (raid resize command) is required.

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.

Table 15: The available options for RAID level changes and the minimum required number of drives

Current level	New level	Minimal number of drives you should add
RAID 0	RAID 1	RAID 0 must contain only 1 drive. 1
	RAID 10	Depending on the number of drives in the RAID: <ul style="list-style-type: none"> • If a RAID 0 contains only 1 drive: 3 • If a 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 10	2
	RAID 5	2
RAID 10	RAID 5	1

Current level	New level	Minimal number of drives you should add
	RAID 50	1, and moreover: <ul style="list-style-type: none"> • and the number of RAID 50 disks must be a multiple of its group size.
RAID 5	RAID 6	1
	RAID 10	1, and moreover: <ul style="list-style-type: none"> • the number of RAID 10 disks must be an even number; • the number of RAID 10 disks must not be less than 2 usable RAID 5 capacities.
	RAID 50	1, and moreover: <ul style="list-style-type: none"> • 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 usable capacity of RAID 50 must not become less than the usable capacity of RAID 5.
RAID 6	RAID 7.3	1
	RAID 60	1, and moreover: <ul style="list-style-type: none"> • 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 usable capacity of RAID 60 must not become less than the usable capacity of RAID 6.
RAID 7.3	RAID 70	1, and moreover: <ul style="list-style-type: none"> • the number of disks in RAID 70 must be at least 8; • the number of disks in RAID 70 must be a multiple of its group size; • the usable capacity of RAID 70 must not become less than the usable capacity of RAID 7.3.

Current level	New level	Minimal number of drives you should add
RAID 50	RAID 60	1, and moreover: <ul style="list-style-type: none"> 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 usable capacity of RAID 60 must not become less than the usable capacity of RAID 50.
RAID 60	RAID 70	1, and moreover: <ul style="list-style-type: none"> the number of disks in RAID 70 must be at least 8; the number of disks in RAID 70 must be a multiple of its group size; the usable capacity of RAID 70 must not become less than the usable capacity of RAID 60.

raid restripe

The chapter describes the available subcommands for the restripe operation.

Tip:

Restripping for RAID 50, 60, and 70 has been added to the `xicli` commands. To learn more, see

```
# restripe -h
```

To start RAID restripping, run

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

Table 16: Arguments for the `restripe start` subcommand

Required arguments		
-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 RAID 50, 60, and 70. The new group size for the RAID. Possible values: integers from 4 to 32.

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

To pause RAID restriping, run

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

Table 17: Argument for the `restripe stop` subcommand

Required argument

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

To continue RAID restriping, run

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

Table 18: Argument for the `restripe continue` subcommand

Required argument

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

raid resize

The chapter describes the command for the resizing operation.

Tip:

We recommend that you run the `raid resize` command when there is no or minimal workload on the RAID.

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

Table 19: Argument for the `resize` subcommand

Required argument

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

Examples of Restriping and Resizing

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
```

Example: Increasing the RAID “media5” size by replacing the drives (3 GB each) with drives of larger size (5 GB each): `/dev/sde`, `/dev/sdf`, `/dev/sdg`, `/dev/sdh`):

In this example, the RAID size will be increased from 9 GB to 15 GB.

Perform a RAID replacement and reconstruction for each drive in a turn, waiting for reconstruction to complete:

1. Change the first drive:

```
# xicli raid replace -n media5 -no 0 -d /dev/sde
# xicli raid recon start -n media5
```

Wait for reconstruction to complete.

2. Change the second drive:

```
# xicli raid replace -n media5 -no 1 -d /dev/sdf
# xicli raid recon start -n media5
```

Wait for reconstruction to complete.

3. Change the third drive:

```
# xicli raid replace -n media5 -no 2 -d /dev/sdg
# xicli raid recon start -n media5
```

Wait for reconstruction to complete.

4. Change the fourth drive:

```
# xicli raid replace -n media5 -no 3 -d /dev/sdh
# xicli raid recon start -n media5
```

Wait for reconstruction to complete.

5. Run resize:

```
# xicli raid resize -n media5
```

RAID size is increased to 15 GB, RAID is in the `need_init` state.

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
```

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]
```

Table 20: Arguments for the `modify` subcommand

Required argument

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

Optional arguments

-inp	--init_prio	<p>Except RAID 0.</p> <p>Initialization priority in %.</p> <p>Possible values are from 0 to 100 (maximum rate of initialization).</p> <p>The default: 100.</p>
-mwe	--merge_write_enabled	<p>Except RAIDs 0, 1, 10.</p> <p>Enable (1) or disable (0) the Merge function for write operations.</p> <p>The default: 0.</p>
-mre	--merge_read_enabled	<p>Except RAIDs 0, 1, 10.</p> <p>Enable (1) or disable (0) the Merge function for read operations.</p> <p>The default: 0.</p>
-ml	--memory_limit	<p>RAM usage limit in MiB.</p> <p>Possible values: 0 and integers from 1024 to 1048576.</p> <p>The 0 value sets unlimited RAM usage.</p> <p>The default: 0 (unlimited).</p>
-mm	--merge_max	<p>Except RAIDs 0, 1, 10.</p> <p>Maximum wait time (in microseconds) for stripe accumulation for the Merge functions.</p> <p>Possible values: integers from 1 to 100000.</p> <p>The default: 1000.</p>
-mw	--merge_wait	<p>Except RAIDs 0, 1, 10.</p> <p>Wait time (in microseconds) between requests for the Merge functions.</p> <p>Possible values: integers from 1 to 100000.</p> <p>The value must be less than the merge_max value.</p> <p>The default: 300.</p>

-rcp	--recon_prio	<p>Except RAID 0.</p> <p>Reconstruction priority in %.</p> <p>Possible values: from 0 to 100 (maximum rate of reconstruction).</p> <p>The default: 100.</p>
-re	--resync_enabled	<p>Except RAIDs 0, 1, 10.</p> <p>Enable (1) or disable (0) the resync function.</p> <p>The default: 1.</p>
-rl	--request_limit	<p>Number of simultaneous I/O requests on RAID.</p> <p>Possible values: integers from 0 to 4294967295.</p> <p>The 0 value disables the restriction.</p> <p>The default: 0.</p>
-rsp	--restripe_prio	<p>Restriping priority in %.</p> <p>Possible values are from 0 to 100 (maximum rate of restriping).</p> <p>The default: 100.</p>
-se	--sched_enabled	<p>Enable (1) or disable (0) the scheduling function.</p> <p>The default: 0.</p>
-sp	--sparepool	<p>Name of the spare pool to assign to the RAID.</p> <p>The <code>null</code> value removes the spare pool from the RAID.</p>
	--force_online	<p>Change RAID state to online if the RAID has unrecoverable sections.</p> <p>I/O operations on unrecoverable sections may lead to data corruption.</p> <p>The argument takes no value.</p>
	--force_resync	<p>Force RAID re-initialization.</p> <p>The argument takes no value.</p>

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

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

Drives

Manual Drive Replacement or Excluding

To exclude or replace a drive in a RAID, run:

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

Tip:

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

Table 21: 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 <pre># xicli raid show</pre>
-d	--drive	The new block device. To remove the drive (to mark it as missing) set the <code>null</code> value.

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

1. Mark the drive "0" as "missing":

```
# xicli raid replace -n media5 -no 0 -d null
```

2. Replace 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

- physically removed from a RAID;
- exceeded the threshold value of wear;
- exceeded the threshold value of I/O errors.

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. One spare pool can have only SSDs or only NVMe drives.

If the system has a spare pool, you can assign it to an existing RAID or when creating a new RAID.

Commands for managing spare pools

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

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

Table 22: 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>
```

Table 23: 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>
```

Table 24: 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>
```

Table 25: 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]
```

Table 26: Arguments for the `show` subcommand

Optional arguments

<code>-n</code>	<code>--name</code>	The name of the spare pool. Without the argument, show info on all spare pools.
<code>-f</code>	<code>--format</code>	Output format: <ul style="list-style-type: none"> • <code>table</code>; • <code>json</code>; • <code>prettyjson</code> – human-readable json. The default: <code>table</code> .
<code>-u</code>	<code>--units</code>	Size units: <ul style="list-style-type: none"> • <code>s</code> – sectors (1 sector=512 bytes); • <code>k</code> – kilobytes; • <code>m</code> – megabytes; • <code>g</code> – gigabytes. The default: <code>g</code> .

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

Table 27: Argument for the `pool modify` subcommand

Required argument

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

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

```
# xicli settings pool show
```

Table 28: Argument for the pool show subcommand

Optional argument

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

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.

Tip:

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 or replaced with a suitable drive from the spare pool. You can set the fault threshold value in the range from 1 to 1000. 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>
```

Table 29: Argument for the `faulty-count modify` subcommand

Required argument

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

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
```

Table 30: Argument for the `faulty-count show` subcommand

Optional argument

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

To reset the current numbers of faults for drives, run

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

Table 31: Arguments for the `faulty-count reset` subcommand

Required argument

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

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]
```

Table 32: 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 (<code>/dev/sd*</code> , <code>/dev/mapper/mpath*</code> , <code>/dev/nvme*</code> , <code>/dev/dm-*</code>) separated by a space to show their current numbers of faults. If neither of the two arguments is specified, show the values for all drives.

Optional argument

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

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



Attention:

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

Metadata is xiRAID RAID 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 that was removed from a RAID due to exceeding the I/O error threshold.
To remove the metadata from such disks, add a new disk to the RAID to replace the removed one.
- on a disk included in a RAID that is present in the current configuration file.

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

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

Table 33: Argument for the `clean` subcommand

Required argument

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

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

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

Configuration Files and Metadata

In this chapter, you can learn:

- how the information about xiRAID RAID objects is stored;
- how to use this information to import a RAID to another system;
- how to restore the configuration if a system disk fails;
- how to restore a RAID.

xiRAID RAID Configuration

After creating a xiRAID RAID device, the information about the created device automatically goes to the following locations in the system:

- the current configuration file `/etc/xiraid/raid.conf`;
- the metadata on the disks included in the created xiRAID RAID device.

Additionally, before changing the current configuration file, the system automatically creates a backup file of the current configuration `/etc/xiraid/raid.conf.bak`. Thus, if necessary, you can restore the previous version of the device configuration.

The current configurations file

The current configurations file stores the most recent changes to the xiRAID RAID devices and is used when working with created xiRAID RAID devices.

The example of the configuration file:

```
{
  "raids": {
    "raid_one": {
      "name": "raid_one",
      "uuid": "C2875DCE-971E-4401-9D2C-7EBC36422B6A",
      "level": "0",
      "synd_cnt": 0,
      "strip_size": 16,
      "block_size": 4096,
      "drives": [
        "drive-scsi9"
      ],
      "size": 20578304,
      "group_size": 1
    },
    "raid_two": {
```

```

        "name": "raid_two",
        "uuid": "3967F298-893A-45E3-A79F-051FE6C499F7",
        "level": "1",
        "synd_cnt": 0,
        "strip_size": 16,
        "block_size": 4096,
        "drives": [
            "drive-scsi8",
            "drive-scsi7"
        ],
        "size": 20578304,
        "group_size": 2
    }
},
"drives": {
    "drive-scsi8": "1",
    "drive-scsi7": "2"
},
"faulty_count_threshold": 3,
"version": "4.0.0",
"timestamp": 1648628802.796359
}

```

The "raids" object contains the settings of the created xiRAID RAIDs.

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

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

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

Configurations metadata on disks

Since the current 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 RAID devices.

Each disk contains data that enables a complete copy of the configuration file to be restored.

Importing RAID

RAID Import is the transfer of RAID-containing disks from one system to another. Once imported, the xiRAID RAID device becomes available for management and its configuration information is added to the current 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 current 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]
```


Table 34: Arguments for the `import show` subcommand

Optional arguments

<code>-d</code>	<code>--drives</code>	<p>The list of block devices (<code>/dev/sd*</code>, <code>/dev/mapper/mpath*</code>, <code>/dev/nvme*</code>, <code>/dev/dm-*</code>) separated by a space to show the info.</p> <p>Without the argument, shows the info from all drives.</p>
<code>-f</code>	<code>--format</code>	<p>Output format:</p> <ul style="list-style-type: none"> • <code>table</code>; • <code>json</code>; • <code>prettyjson</code> – human-readable json. <p>The default: <code>table</code>.</p>
	<code>--offline</code>	<p>Show non-recoverable RAID(s) in the import list.</p> <p>The argument takes no value.</p>

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 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 RAID(s) that are available for import, run:

```
# xicli raid import show
```

The command will find and display information about founded RAID(s) on drives that can be imported. Three RAID(s) 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) medial
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 RAID(s) 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]
```

Table 35: 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
```

Configuration File Recovery

If the system disk fails, you can recover xiRAID RAID objects from the metadata on the disks by using the command

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

Subcommands for the `config` command:

apply	Apply the configuration file for all restoring RAIDs.
backup	Save the current configuration file (create the backup file <code>backup_raid.conf</code> 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



Attention:

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 applies the current configuration file and restores all RAIDs with the status "None" from the file and deletes all xiRAID RAIDs that are not in the file.

config backup

To save the current configuration file (to create the backup file `backup_raid.conf` at the current directory), run

```
# xicli config backup
```

config restore

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 current configuration file from a specified location (from a file or disk metadata), but does not apply it.

Table 36: Arguments for the `restore` subcommand

Mutually excluded required arguments

-f	--file	A file to restore or replace the configuration file. 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 one newest configuration file to /etc/xiraid.conf.drive. If no disks are specified, restore one newest configuration file from all disks.

config show

To show configuration files stored on the disks, run

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

Table 37: 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 Current Configuration File

Example: restore the current configuration file from disk metadata.

1. Restore the most recent configuration found among all disks:

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

2. Apply the restored configuration:

```
# xicli config apply
```

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

1. Change the current directory to flash drive:

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

2. Create a copy of the current configuration file:

```
# xicli config backup
```

Restoring the RAID

The RAID restores automatically after any failure.

If this does not happen and the RAID state is "None", you can restore such RAIDs from the current configuration file using

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

Table 38: Arguments for the `restore` subcommand

Mutually exclusive required arguments

-n	--name	The name of the RAID.
-a	--all	Restore all available xiRAID RAIDs. Argument takes no value.

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>
```

Table 39: Arguments for the `scanner modify` subcommand

At least one argument is required

-pi	--polling_interval	The polling interval for xiRAID 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
```

Table 40: Argument for the `scanner show` subcommand

Optional argument

-f	--format	Output format: <ul style="list-style-type: none"> • 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>
```

Table 41: Argument for the `locate` subcommand

Required argument

-d	--drives	<p>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).</p> <p>The argument doesn't affect the automatic indication.</p>
----	----------	---

Example: Turn location indication on for drives “/dev/nvme0n1” and “/dev/nvme1n1”:

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

Notifications

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).

You can:

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

Table 42: Argument for the log modify subcommand

Required argument

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

To collect all logs into a file, run

```
# xicli log collect
```

The log file will be available in the /tmp directory.

After collecting is complete, the message “xiRAID logs have been collected and saved in: /tmp/xiraid_logs_\${hostname}_YYYY.MM.DD_hh-mm-ss.tar.gz” shows.

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

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

Table 43: Argument for the `log show` subcommand

Optional argument

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

To see the latest error messages on the system, run

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

Table 44: Argument for the `show` subcommand

Optional argument

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

Setting up Email Notifications



Attention:

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

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

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

Table 45: Arguments for the `add` subcommand

Required arguments

<code>-a</code>	<code>--address</code>	Receiver's email.
<code>-l</code>	<code>--level</code>	<p>The notification level.</p> <p>Possible values:</p> <ul style="list-style-type: none"> <code>info</code> – Info notifications; <code>warning</code> – Error and Warning notifications; <code>error</code> – Error notifications.

Table 46: 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
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 license expired
Reconstruction started on RAID (...)		xiRAID 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
		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>
```

Table 47: 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
```

Table 48: Argument for the `mail show` subcommand

Optional argument

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

To manage email notification settings, run

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

Table 49: Arguments for the `mail modify` subcommand

At least one argument is required

-pi	--polling_interval	The polling interval for xiRAID RAID's and the drives in seconds. Possible values: integers from 1 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 1 to 1440 (24 hours). The default: 10.

To show email notification settings, run

```
# xicli settings mail show
```

Table 50: Argument for the `mail show` subcommand

Optional argument

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

Number of CPU Threads of the xiraid Module

There are two ways to control the number of CPU threads for xiraid:

- the `cpuignore` command

Specifics:

- possible to select specific CPUs for the limitation;
- does not require restarting the module;
- the limits set by this command are reset after the system or module restart.

- the `modprobe` command

Specifics:

- not possible to select specific CPUs. For limitation, OS starts to assign CPUs with the first CPU (ID 0);
- requires restarting the module;
- possible to configure the module to load with the restriction parameters at OS startup.

To select the CPUs that will not be used for the xiraid module, run

```
# xicli settings cpu-ignore modify <arg>
```

Table 51: Argument for the `cpu-ignore modify` subcommand

Required argument

	--id
	The list of CPU IDs (separated by a comma or a hyphen) that will not be used for xiraid.
	The <code>null</code> value removes the restriction on using threads for xiraid.

Example: Disable CPUs with the IDs 0, 1, 2, 3, 6 for the module:

```
# xicli settings cpu-ignore modify --id 0-3,6
```

Example: Remove all restrictions on CPUs for the module:

```
# xicli settings cpu-ignore modify --id null
```

To show the list of CPUs that are not used for the xiraid module, run

```
# xicli settings cpu-ignore show
```

Table 52: Argument for the `cpu-ignore show` subcommand

Optional argument

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

To limit the number of threads for the xiraid module:

1. Unload the module from the kernel:

```
# rmmod xiraid
```

2. Load the module in the kernel with the `cpu_cnt` parameter:

```
# modprobe xiraid cpu_cnt=<cnt>
```

where `<cnt>` is the number of CPUs used by the module. The operating system assigns CPUs starting with the very first CPU ID.

3. Restart xiraid target:

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

To configure the xiraid module to load automatically with CPU thread parameters, in the `modprobe.conf` file (file name and location depend on your OS, see `man modprobe.conf` for details) add the line

```
options xiraid cpu_cnt=<cnt>
```

where `<cnt>` is the number of CPUs used by the module. The operating system assigns CPUs starting with the very first CPU ID.

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.

- The recommended stripe size for the xiRAID RAID is 16 KiB (set by default).

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.

Tip:

Enable Scheduling when the access pattern is low threaded.

--merge_write_enabled

The `--merge_write_enabled` improves the system workload performance when access pattern is sequential and high threaded, and the block sizes are small. This parameter sets a waiting time for all incoming requests in sequential areas. During waiting time, requests to this area are not intentionally transferred to the drives. Instead of immediate data transfer, incoming requests are formed into a tree structure. At the end of waiting time, requests are merged together if possible. This function reduces the number of read-modify-write operations on syndrome RAIDs.

Despite the extra waiting time, this function can improve the system workload performance. If the access pattern is mainly random or queue depth is small, the waiting time will not allow merging requests. In this case enabling `--merge_write_enabled` will decrease the system workload performance.

Tip:

Enable merge by the `--merge_write_enabled` parameter when the access pattern is sequential and high threaded and the block sizes are small.

Since the time between incoming I/O depends on the workload intensity, size, and other parameters, it may be necessary to change `--merge_wait` and `--merge_max` parameters for better query consolidation. Usually, large I/O sizes require large values for these parameters.

The function only works when the condition is met:

```
data_drives * stripe_size ≤ 1024
```

where

- “data_drives” is a number of drives in the RAID (for RAID 5, 6, or 7.3) or in one RAID group (for RAID 50, 60, or 70) that are dedicated for data;
- “stripe_size” is a selected stripe size for the RAID (stripe_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 queue depth of user's workload is not enough to merge a full stripe. Activate Merge, if

```
iodepth * block_size ≥ data_drives * stripe_size
```

where “block_size” is a block size of the RAID (the block_size value in RAID parameters) in KiB.

--request_limit

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

Tip:

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 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.

--resync_enabled

The function starts a RAID re-initialization after an unsafe system shutdown, thereby protecting syndromic RAID 5 (all but RAID 0, RAID 1, and RAID 10) from data loss caused by a write hole.

To disable resync for all RAIDs, run

```
# modprobe xiraid resync=0
```

Strip Size

Recommended RAID strip size is 16 KiB.

RAM Limit

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

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

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

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

NUMA

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

To figure out the NUMA node drive, run:

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

or via `lspci`:

```
# lspci -vvv
```

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

System

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

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

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

In the flags field, check for the ht flag.

Command output example:

```
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss
ht syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon rep_good nopl xtopology cpuid tsc_known_freq pni
pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti tpr_shadow vnmi flexpriority ept
vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat unip
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 Xinnor xiRAID 4.0.0, 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 below.

NVMe-oF

1. Xinnor xiRAID 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 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 a SPDK NVMe-oF target, set the InCapsuleDataSize parameter denoting at by what value requests should be split.

Swap File

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

File System Mounting Aspects

Since the system restores xiRAID 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.

Tip:

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

`systemd.mount`

When automatic mounting at system startup via `systemd`, in the `[Unit]` section, put the following strings:

- `Requires = xiraid-restore.service`
- `After = xiraid-restore.service`

Example: mounting xfs located on a RAID `/dev/xi_raidname` into `/mnt/raid/` through `systemd.mount`:

1. Set a timeout of 5 minutes for the xiRAID device in the unit file:

a. Run

```
# systemctl edit --force --full /dev/xi_raidname
```

b. Add the following lines:

```
[Unit]
JobRunningTimeoutSec=5m
```

Save the changes.

c. Check the changes:

```
#systemctl cat /dev/xi_raidname
```

2. 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"`).

Table 53: The example file `/etc/systemd/system/mnt-raid.mount`

```
[Unit]
Description=Mount filesystem on Xinnor xiRAID
Requires=xiraid-restore.service
After=xiraid-restore.service
DefaultDependencies=no
Before=umount.target
Conflicts=umount.target

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

[Install]
WantedBy=multi-user.target
```

3. 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`

When setting up automatic mounting at system startup via `/etc/fstab`, point out one of the following sets of options:

- `x-systemd.requires=xiraid-restore.service,x-systemd.device-timeout=5m,_netdev`
- `x-systemd.requires=xiraid-restore.service,x-systemd.device-timeout=5m,nofail`

The parameter “`x-systemd.requires`”

The value "xiraid-restore.service" for this parameter sets a strict dependency of device mounting on service execution.

The parameter “`x-systemd.device-timeout`”

The parameter `x-systemd.device-timeout=` configures how long systemd should wait for a device to show up before giving up on an entry from `/etc/fstab`. Specify a time in seconds or explicitly append a unit such as "s", "min", "h", "ms".

Note that this option can only be used in `/etc/fstab`, and will be ignored when part of the `Options=` setting in a unit file.

The value “_netdev”

The value `_netdev` sets that the filesystem resides on a device that requires network access (used to prevent the system from attempting to mount these filesystems until the network has been enabled on the system).

The value “nofail”

If the device is not permanently connected and may not be present when the system starts, mount it with the value `nofail`. This will prevent from errors when mounting such a device.

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

Table 54: The string from the file `/etc/fstab`

```
/dev/xi_raidname /mnt/raid/ xfs x-systemd.requires=xiraid-restore.service,x-systemd.device-
timeout=5m,_netdev 0 0
```

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

Table 55: The string from the file `/etc/fstab`

```
/dev/xi_raidname /mnt/raid/ xfs x-systemd.requires=xiraid-restore.service,x-systemd.device-
timeout=5m,nofail 0 0
```

DKMS Specifics when Updating Linux Kernel



Attention:

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 Xinnor xiRAID 4.0.0 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:

- 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 CentOS, RHEL, Rocky Linux, Alma Linux, Oracle Linux (EL));
- `kernel-uek-devel` (for Oracle Linux);
- `kernel-default-devel` (for OpenSUSE, SLES)
- `linux-headers` (for Ubuntu, Debian);
- `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 Xinnor xiRAID 4.0.0 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-5.4.0-56-generic linux-headers-5.4.0-56-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



Attention:

When you change any parameter of the settings auth modify command, the xiraid.target service restarts.

Tip:

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

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

Table 56: Arguments for the auth modify subcommand

At least one argument is required

--host	The host name or IP address that will be used for the connection. After changing the host, you must regenerate and replace the certificate. The default: localhost.
--port	The port that will be used for the connection. The default: 6066.

To view client-server connection settings, run the command

```
# xicli settings auth show
```

Table 57: 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:

**Attention:**

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

Error on applying license on OS Debian 9:

```
Error: License not updated. Bad license key!
```

More indicators:

- `dmesg` has these messages:

```
xiraid: alg: akcipher: Failed to load tfm for rsa: -2
xiraid: Cannot decrypt license key
```

- The file `/boot/config-$(uname -r)` has the line "`CONFIG_CRYPTORSA=m`".

This means that the driver responsible for RSA encryption is not built as part of the kernel but as a loadable module.

Solution:

1. Load the `rsa_generic` driver:

```
# modprobe rsa-generic
```

2. To load the driver automatically at system startup, add the line `rsa-generic` to the file `/etc/modules`.

Error: Missing xiRAID RAID system module

Possible reasons:

- After updating the OS kernel, the packages with the header files (`kernel-devel`, `kernel-uek-devel`, `linux-headers`, `pvc-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 Xinnor xiRAID 4.0.0 Installation Guide for details.

After that, run the command

```
# dkms autoinstall
```

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

WARNING! Diff between built and installed module!

The message is shown when you check DKMS status:

```
# dkms status
```

Possible reason:

DKMS installs not the current kernel version but the newest.

Solution:

Run

```
# dkms remove  
# dkms install -m xiraid -v 4.0.0 --force
```

Error: failed to connect to all addresses

Possible reason:

The client tried accessing the server by the old address or with incorrect credentials.

Solutions:

- Restart xiraid target:

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

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