request@raidix.com
raidix.com

RAIDIX ERA 3.4.0 ADMINISTRATOR'S GUIDE

Document version 1.0

2021

CONTENTS

INTRODUCTION

Intended Audience

This guide is intended for administrators and users of RAIDs based on the RAIDIX ERA 3.4.0 software.

The guide contains instructions on how to configure and manage RAIDs in RAIDIX ERA 3.4.0.

Guide Conventions

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

Convention	Uses
Bold	GUI controls, option value, minor titles.
Italic	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:

Note – a note, which provides valuable information.

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

ABOUT RAIDIX ERA 3.4.0

RAIDIX ERA 3.4.0 is high-performance software RAID developed specifically for NVMe storage devices and new types of SAN networks. RAIDIX ERA 3.4.0 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.

RAIDIX ERA 3.4.0 is a Linux kernel module and a management utility, which are built and configured for the most popular distributions (see the document *RAIDIX ERA 3.4.0 System Requirements*). The software is installed on servers with slots for Flash memory devices or with connected JBOFs. RAIDIX ERA 3.4.0 enables you to combine drives into high-performance fault-tolerant RAIDs.

RAIDIX ERA 3.4.0 Specifications

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

1. USING ERARAID

Manage your software ERA RAID in Linux by using `eraraid` utility.

1.1 Command Line Interface (CLI) Description

Conventions on CLI command syntax

Item's format	Description
`item`	Required item (object, method, parameter, attribute).
`<item>`	Placeholder variable.
`[item]`	Optional item.
`{item1|item2}`	Mutually exclusive arguments.
`(items)`	Multiple values for the argument.

In the CLI, enter commands in the following format:

```
# eraraid <mode> (required_parameters) [(optional_parameters)]
```

The `<mode>` parameter has the following values:

`config`	Manage a configuration file.
`cpuignore`	Manage number of PCU threads in eraraid.
`create`	Create a RAID.
`destroy`	Delete a RAID completely.
`drive-clean`	Remove drive metadata.
`error-log`	Show errors in ERA RAID.
`import`	Import the configuration of a selected RAID from the configuration file to selected drives.
`import-all`	Import all possible RAIDs.
`import-show`	Show importable RAIDs.
`init`	Manage RAID initialization.
`license`	Manage a license.
`locate`	Manage LED indication of drives.
`logs`	Collect system logs.
`mail`	Set up mail notifications.
`modify`	Modify RAID parameters.
`pool`	Manage sparepools for automatic drive replacement in RAIDs.
`pool-show`	Show sparepool info.
`recon`	Manage RAID reconstruction.

`replace`	Replace or delete a RAID drive.
`restore`	Restore a RAID.
`restripe`	Manage RAID restriping.
`scanner`	Controls the parameters of the drive presence and S.M.A.R.T. health scanning intervals, as well as the parameters of the automatic LED indication of drives.
`faulty-count`	I/O error threshold management for drives and resetting the current I/O error counts on drives.
`faulty-count-show`	Show the I/O error threshold for drives and current I/O error values of drives.
`show`	Show RAID info.
`unload`	Unload a RAID.

To show the full list of commands, run:

```
# eraraid -h
```

To show the ERA RAID version, run:

```
# eraraid -v
```

Console syntax peculiarities:

1. Type the command parameters in one line.
2. Commands parameters are separated by spaces.
3. Use short or long forms of command attributes.

   Example:

   ```
   # eraraid create {-n|--name} <raid_name> {-l|--level}
   <raid_level> {-d|--drives} (block_devices)
   ```

4. To get the list of all methods and objects of lower levels, add attribute `-h`:

   ```
   # eraraid <mode> -h
   ```

Detailed description of methods is presented below.

1.2 License Management

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

```
# eraraid license --show
```

Command output example when no license was added:

```
[root@eratest1 ~]# eraraid license --show
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:

```
[root@eratest1 ~]# eraraid license --show
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
```

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 expire date.
disks	Maximum number of drives.
levels	Maximum RAID level.
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:

```
# eraraid license --show > license_request.txt
```

To get your license key, send your hardware key to the RAIDIX support team by e-mail (support@raidix.com).

After you got your license file, copy it to the server (for example, to the /roo directory), and apply the license key by running the command:

```
# eraraid license --update /root/license.txt
```

> Additional actions may be required to apply the license for OS Debian 9. To learn more, see the chapter *3. Troubleshooting*.

To check the applied license, run:

```
# eraraid license --show
```

The full-parameter command:

```
# eraraid license {--show|--update {path}|--reset}
```

Description of the "license" command parameters

Required parameters		
-s	--show	Show license information.
-u	--update	Update license from the specific path.
-r	--reset	Delete current license.

1.3 RAID

Creating a RAID

To create a RAID of levels 0, 1, 5, 6, 7.3, or 10, run:

```
# eraraid create -n <raid_name> -l {0|1|5|6|7|10} -d (drives)
[-ss {16|32|64|128|256}] [-bs {512|4096}]
```

To create a RAID of levels 50, 60, or 70, you need in addition point the `-gs` parameter:

```
# eraraid create -n <raid_name> -l {50|60|70} -d (drives)
-gs <group_size> [-ss {16|32|64|128|256}] [-bs {512|4096}]
```

To create a RAID N+M, add the parameter `-sc`:

```
# eraraid create -n <имя_raid> -l nm -d (block_devices) -sc
<number of syndroms> [-ss {16|32|64|128|256}] [-bs {512|4096}]
```

Description of the "modify" command parameters

Required parameters		
`-n`	`--name`	RAID name.
`-l`	`--level`	RAID level: **0**, **1**, **5**, **6**, **7**, **10**, **50**, **60**, **70**, or **N+M**.
`-d`	`--drives`	Space-separated list of block devices.
`-gs`	`--group_size`	Only for RAIDs 50, 60, or 70. Set the drives number for one RAID group of level 5, 6, or 7.3 of an appropriate RAID 50, 60, or 70. Possible values are integers from **4** to **32**.
`-bs`	`--block-size`	RAID block size: **512** or **4096** bytes.
`-sc`	`--synd-cnt`	Only for RAIDs N+M. Number of syndromes M. Possible values are integers from 4 to 32. Additional conditions: N+M ≤ 64 and M ≤ N.
Optional parameters[1]		
`-ss`	`--strip-size`	Strip size in KiB. Possible values are **16**, **32**, **64**, **128**, or **256**. The default is **16**.
`-ip`	`--init-prio`	Except RAID 0. Initialization priority. Possible values are from **0** to **100**%. The default is **100**.
`-rp`	`--recon-prio`	Except RAID 0. Reconstruction priority. Possible values are from **0** to **100**%. The default is **100**.
`-me`	`--merge-enabled`	Enable (**1**) or disable (**0**) the Merge function. The default is **0**.
`-se`	`--sched-enabled`	Enable (**1**) or disable (**0**) the Scheduling function. The default is **0**.
`-sp`	`--sparepool`	Name of a sparepool that is assigned to the RAID. The value "**null**" removes the sparepool from the RAID.
`-ml`	`--memory-limit`	RAM usage limit in MiB. The default: **0** (unlimited).
`-rl`	`--request-limit`	Activate (**1**) or deactivate (**0**) limitation of requests per RAID. The default: **0**.
`-mw`	`--merge-wait`	The waiting time (in microseconds) between requests when Merge is enabled. The default: **300**.

[1] See the chapter "Set up RAID Parameters. Recommendations"

-mm	--merge-max	Maximum waiting time (in microseconds) for stripe accumulation with Merge enabled. The default: 1000.
-re	--resync-enabled	Only for syndrome RAIDs. Enable (1) or disable (0) the Resync function. The default: 1.

Example: Creation of a RAID 5 named "era5" consisting of 4 NVMe drives — "nvme0n1", "nvme1n1", "nvme2n1", "nvme3n1", strip size equal to 64 KiB and enabled Merge.

```
# eraraid create -n era5 -l 5 -d /dev/nvme0n1 /dev/nvme1n1
/dev/nvme2n1 /dev/nvme3n1 -ss 64 -me 1
```

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 – 8 drives.

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

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

Showing RAID State

To see information on a RAID state, run:

```
# eraraid show [{-n <raid_name>|-o}] [-u {s|k|m|g}]
[-f {table|json|prettyjson}] [-e]
```

Description of the "show" command parameters

Optional parameters		
-n	--name	RAID name.
-o	--online	Show only RAIDs, which weren't unloaded via the `unload` command.
-u	--units	Size units: **s** – sector, **k** – kilobyte, **m** – megabyte, **g** – gigabyte.

-f	--format	Output format: • **table** – as a table; • **json** – json; • **prettyjson** – human-readable json.
-e	--extended	Show the extended table

Example: Showing information on the RAID "era5":

```
# eraraid show -n era5 -e
```

RAIDs

name	static	state	devices	health	wear	serials	params	info
era5	size: 29 GiB level: 5 strip_size: 64 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/sda online 1 /dev/sdb online 2 /dev/sdc online 3 /dev/sdd online	100% 100% 100% 100%	1% 9% 1% 3%	drive-scsi0 drive-scsi9 drive-scsi8 drive-scsi7	init_prio : 100 init_depth : 128 recon_prio : 100 recon_depth : 64 memory_limit_mb : 0 merge_enabled : 0 resync_enabled : 1 sched_enabled : 0 req_limit_enabled: 0 restripe_prio : 100 merge_wait_usecs : 300 merge_max_usecs : 1000	memory_usage_mb : -

Description of the "show" command output

Row	Description
name	RAID name.

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 50, 60, and 70 – size of the corresponding RAID group.
	• strip_size.
	• sparepool – name of the assigned sparepool.
	• active:
	  ○ True, if the RAID's block device is in the system.
	  ○ False, if:
	  ▪ The RAID was not loaded after reboot. Restore the RAID.
	  ▪ The RAID is unloaded. Restore the RAID.
	• config:
	  ○ True, if the RAID is in the configuration file.
	  ○ False, if the RAID is missing.

Row	Description
state	RAID state: • online – the RAID is available and ready to work. • initialized – initialization 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 in the process of reconstruction. • offline – the RAID is unavailable. • need_recon – the RAID needs reconstruction. • need_init – the RAID needs initialization. • read Only – the license 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: • 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	To show, use the command with the `-e` parameter. Percent of valid drive data. When health is 100% – no reconstruction required.
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.
serials	To show, use the command with the -e parameter. Serial numbers of drives in RAID.

Row	Description
params	To show, use the command with the `-e` parameter. Editable RAID parameters: • init_prio – (except RAID 0) initialization priority: from 0% to 100%. • init_depth – the queue depth of initialization requests. • recon_prio – (except RAID 0) reconstruction priority: from 0% to 100%. • recon_depth – the queue depth of reconstruction requests. • memory_limit_mb – the value limited RAM usage, in megabytes. • merge_enabled – is the parameter Merge enabled (1) or disabled (0). • resync_enabled – is the function Resync enabled (1) or disabled (0). • sched_enabled – is the function Scheduling enabled (1) or disabled (0). • req_limit_enabled – is the function Request-limit enabled (1) or disabled (0). • restripe_prio – priority for restriping: from 0% to 100%. • merge_wait_usecs – waiting time between requests when Merge is enabled. • merge_max_usecs – maximum time to wait for stripe accumulation with Merge enabled.
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 not displayed. • restripe_progress – restriping progress: from 0% to 100%.

Restoring a RAID

RAID restores automatically after any failure. Also you can restore a RAID manually by running the command:

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

Description of the "restore" command parameters

Required parameters		
`-n`	`--name`	RAID name.
`-a`	`--all`	Restore all RAIDs.

Example: Restoring the RAID "era5":

```
# eraraid restore -n era5
```

Deleting a RAID

To delete a RAID completely, run:

```
# eraraid destroy {-n <raid_name>|-a}
```

Description of the "destroy" command parameters

Required parameters		
-n	--name	RAID name.
-a	--all	Destroy all RAIDs.

Example: Deleting the RAID "era5":

```
# eraraid destroy –n era5
```

Unloading a RAID

Unloading (or deactivation) is a deletion of a RAID while keeping the configuration file. Unlike complete deletion, unloading enables you to restore the RAID later. To unload a RAID, run:

```
# eraraid unload [-h] {-n <raid_name>|-a}
```

Description of the "unload" command parameters

Required parameters		
-n	--name	RAID name.
-a	--all	Unload all RAIDs.

Example: Unloading the RAID "era5":

```
# eraraid unload –n era5
```

To restore an unloaded RAID, run:

```
# eraraid 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 unavailable.

To manage the reconstruction process, run:

```
# eraraid recon -n <raid_name> {--start|--stop}
```

Description of the "recon" command parameters

Required parameters		
-n	--name	RAID name.
	--start	Start reconstruction.
	--stop	Stop reconstruction.

Example: Starting RAID "era5" reconstruction:

```
# eraraid recon --start -n era5
```

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

RAID Initialization

Initialization will start automatically when a RAID (except RAID 0) is created. While a RAID is being initialized, the functions reconstruction and restriping are unavailable.

To manage the initialization process, run:

```
# eraraid init -n <raid_name> {--start|--stop}
```

Description of the "init" command parameters

Required parameters		
-n	--name	RAID name.
	--start	Start initialization.
	--stop	Stop initialization.

Example: Starting initialization of the RAID "era5":

```
# eraraid init --start -n era5
```

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.

RAID Restriping

Restriping enables the next modifications of created RAIDs:

- Change a RAID level.
- *Increase* a RAID size by adding new drives.

Requirements and specifics:

- Only for RAIDs of levels 0, 1, 10, 5, 6, 7.3.
- Initialization of the RAID must be finished.
- Only one RAID can be restriped at a time.
- While restriping, reconstruction and resync are unavailable.
- 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;
  - reconstructing;
  - need_recon;
  - need_init;
  - initing;
  - need_restripe;
  - restriping;
  - degraded.

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

RAID level change	Minimal number of drives you should add
RAID 0 to RAID 1	*RAID 0 must contain only 1 drive.* 1
RAID 0 to RAID 10	• *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 to RAID 10	2
RAID 1 to RAID 5	2
RAID 10 to RAID 5	1
RAID 5 to RAID 6	1
RAID 6 to RAID 7.3	1

To change RAID level or size, use the commands `restripe` and `resize`.

To restripe, run

```
# eraraid restripe -n <raid_name> -l <level> -d (<drives>)
{--start|--stop|--continue}
```

You can use the command `resize` in the following cases:

- RAID is in the *need_resize* state after restriping;
- to increase RAID size without restriping or without increasing the number of drives.

```
# eraraid resize -n <raid_name>
```

Description of the "restripe" command parameters

Required parameters		
`-n`	`--name`	RAID name.
`-l`	`--level`	New RAID level. If you only increasing the RAID size, enter the current RAID level.
`-d`	`--drives`	Block devices to be added to the RAID.
	`--start`	Start restriping.
	`--stop`	Stop restriping.
	`--continue`	Continue restriping.

Example: Restriping of the RAID "era5" by adding a new drive /dev/sdi without changing the RAID level (increasing the RAID size):

```
# eraraid restripe -n era5 -l 5 -d /dev/sdi --start
```

Example: Increasing the RAID "era5" 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 turn, waiting for reconstruction to complete:

1. Change the first drive:

   ```
   # eraraid replace -n era5 -no 0 -d /dev/sde

   # eraraid recon -n era5 --start
   ```

   Wait for reconstruction to complete.

2. Change the second drive:

   ```
   # eraraid replace -n era5 -no 1 -d /dev/sdf

   # eraraid recon -n era5 --start
   ```

   Wait for reconstruction to complete.

3. Change the third drive:

   ```
   # eraraid replace -n era5 -no 2 -d /dev/sdg
   ```

```
# eraraid recon -n era5 --start
```

Wait for reconstruction to complete.

4. Change the fourth drive:

```
# eraraid replace -n era5 -no 3 -d /dev/sdh

# eraraid recon -n era5 --start
```

Wait for reconstruction to complete.

5. Run resize:

```
eraraid resize -n era5
```

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

Example: Restriping of the RAID "era5" with adding new drives /dev/sdf /dev/sdg /dev/sdh and RAID level changing from 5 to 6:

```
# eraraid restripe -n era5 -l 6 -d /dev/sdf /dev/sdg /dev/sdh
--start
```

You can pause and continue restriping:

```
# eraraid restripe -n <raid_name> {--stop|--continue}
```

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

```
# eraraid resize -n <raid_name>
```

Changing RAID Parameters

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

If the priority is equal to 0, reconstruction or initialization starts and continues only if there is no load. By default, all priorities are set to 100%, which stands for the highest possible rate of reconstruction, initialization, and restriping processes.

To increase user throughput-performance, reduce reconstruction and initialization priority.

- The `--merge-enabled` parameter reduces the amount of *read-modify-write*, increasing write performance, which is recommended for small blocks and intensive input stream.
- The `--sched-enabled` parameter optimizes RAID for low-flow loads.

To change the RAID dynamic parameters, run:

```
# eraraid modify -n <raid_name> [-ip <0..100>] [-rp <0..100>]
[--restripe-prio <0..100>] [-me {0|1}] [-se {0|1}] [-ml
```

```
<ram_limit>] [-rl {0|1}] [--force-online] [-mw <0..100000>]
[-mm <0..100000>]
```

Description of the "modify" command parameters

Required parameter		
-n	--name	RAID name.
Optional parameters		
-ip	--init-prio	Except RAID 0. Initialization priority. Possible values are from 0% to 100% (maximum rate of initialization).
-rp	--recon-prio	Except RAID 0. Reconstruction priority. Possible values are from 0% to 100% (maximum rate of reconstruction).
	--restripe-prio	Restriping priority. Possible values are from 0% to 100% (maximum rate of restriping).
-me	--merge-enabled	Enable (1) or disable (0) Merge.
-re	--resync-enabled	Enable (1) or disable (0) Resync.
-se	--sched-enabled	Enable (1) or disable (0) Scheduling.
-sp	--sparepool	Name of a sparepool that is assigned to the RAID. The value "null" removes the sparepool from the RAID.
-ml	--memory-limit	RAM usage limit in MiB.
-rl	--request-limit	Activate (1) or deactivate (0) requests limitation per RAID.
	--force-online	Changes RAID state to online if the RAID has unrecoverable sections. I/O operations on unrecoverable sections may lead to data corruption.
-mw	--merge-wait	Set the waiting time (in microseconds) between requests with Merge enabled.
-mm	--merge-max	Set the maximum waiting time (microseconds) for stripe accumulation with Merge enabled.
	--force-resync	Force RAID re-initialization.

Example: Setting reconstruction priority for the RAID "era5" equal to 50%:

```
# eraraid modify -n era5 -rp 50
```

Importing RAID Operations

RAIDIX ERA 3.4.0 provides users the opportunity to import RAIDs that are presented in drives metadata but are not presented in the system configuration file.

Use this feature in cases when you want to unite several ERA RAID systems into one.

RAIDs with identical names cannot exist in one system. Therefore, RAID import can require renaming in case of identical names. If there is drive conflict with the RAID system, RAID imports in *degraded* mode without the conflicting drive. In case of drive conflict with imported RAID, conflict drive places in the RAID that was imported first.

Possible conflicts:

- *name: Conflict with system raids*;
- *drives: Conflict with system raids*;
- *name: Conflict with import raids*;
- *drives: Conflict with import raids*;
- *name: Conflict with system and import raids*;
- *drives: Conflict with system and import raids*.

Drives statuses (messages in the devices row):

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

Import-related commands

```
# eraraid import-show [-f {table|json|prettyjson}] [-d
(block_devices)] [--offline]
```

Description of the "import-show" command parameters

Optional parameters		
`-f`	`--format`	Output format: • **table** – table format; • **json** – json format; • **prettyjson** – human-readable json.
`-d`	`--drives`	Show RAIDs available for import from the selected drives.
	`--offline`	Show unrecoverable RAIDs in the import list.

```
# eraraid import-all [-d (block_devices)]
```

Description of the "import-all" command parameter

Optional parameter		
-d	--drives	Import RAIDs from specified drives.

```
# eraraid import -id <uuid> [-n <name>] [-d (block_devices)]
```

Description of the "import" command parameters

Required parameter		
-id	--uuid	RAID UUID.
Optional parameters		
-n	--name	New RAID name.
-d	--drives	Import RAIDs from selected drives.

Example:

To execute the import-show command, run:

```
# eraraid import-show
```

Utility 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
076FED13-A107-4233-8880-6E1D063AE707	name: era1 level: 7 strip_size: 16 group_size: 6 size: 14 GiB date: 2019-07-29 12:18	0 /dev/sdi normal 1 /dev/sdc normal 2 null no_metadata 3 null no_metadata 4 /dev/sdm normal 5 /dev/sdl normal	drive-scsi8 drive-scsi2 null null drive-scsi12 drive-scsi11	name: Conflict with system raids drives: OK
465CC850-175E-40DB-A0EB-EC1A765E8D4E	name: era1 level: 6 strip_size: 16 group_size: 4 size: 9 GiB date: 2019-07-29 12:25	0 /dev/sdb in_use,no_metadata 1 /dev/sdd normal 2 /dev/sde normal 3 /dev/sdf normal	drive-scsi1 drive-scsi3 drive-scsi4 drive-scsi5	name: Conflict with system and import raids drives: Conflict with system raids era1
C45D0770-B571-48B1-8D5D-7B87EACD4A44	name: era2 level: 0 strip_size: 16 group_size: 2 size: 9 GiB date: 2019-07-29 12:18	0 /dev/sdk normal 1 /dev/sdj normal	drive-scsi10 drive-scsi9	name: OK drives: OK

Conflicts are highlighted in red color. The first RAID has a conflict with system RAIDs (present in the system) by name and drives. The second RAID conflicts with system and import RAIDs by name. The third RAID does not conflict with any RAIDs. To get information about system RAIDs, run:

```
# eraraid show
```

RAIDs

name	static	state	devices	info
era1	size: 4 GiB level: 1 strip_size: 16 active: True config: True	online	0 /dev/sdb online 1 /dev/sdn online 2 /dev/sdo online	recon_progress : 0 memory_usage_mb: -

To import all non-conflict RAIDs, run:

```
# eraraid import-all
```

RAIDs

name	static	state	devices	info
era1	size: 4 GiB level: 1 strip_size: 16 active: True config: True	online	0 /dev/sdb online 1 /dev/sdn online 2 /dev/sdo online	recon_progress : 0 memory_usage_mb: -
era2	size: 9 GiB level: 0 strip_size: 16 active: True config: True	online	0 /dev/sdk online 1 /dev/sdj online	

To import the first RAID, which contains name and drives conflicts, run:

```
# eraraid import -id 076FED13-A107-4233-8880-6E1D063AE707 -n
era3
```

The conflicting drive was replaced to a null drive. To get the RAID out of *degraded* mode, replace the null drive.

```
[root@raineratest10 ~]# eraraid show
RAIDs
```

name	static	state	devices	info
era1	size: 4 GiB level: 1 strip_size: 16 active: True config: True	online	0 /dev/sdb online 1 /dev/sdn online 2 /dev/sdo online	recon_progress : 0 memory_usage_mb: -
era2	size: 9 GiB level: 0 strip_size: 16 active: True config: True	online	0 /dev/sdk online 1 /dev/sdj online	
era3	size: 9 GiB level: 6 strip_size: 16 active: True config: True	online degraded initialized	0 null offline 1 /dev/sde online 2 /dev/sdf online 3 /dev/sdd online	init_progress : 100 recon_progress : 0 memory_usage_mb: 0

Managing RAID Configurations

To archive or restore (in case of its loss or damage) the configuration file by using metadata on the drives, run:

```
# eraraid config [{-r [<file>]|-a|-b|-d [(block_devices)]|-p
[(block_devices)]}]
```

Description of the "config" command parameters

Optional parameters		
-r	--restore	Restore configuration from the file (if not specified, restore configuration from */etc/eraraid/eraraid.conf.back*).
-a	--apply	Apply configuration by unloading and restoring all RAIDs.
-b	--backup	Copy the current configuration into backup file *backup_eraraid.conf*.
-d	--drives	Get configuration from drives and restore it to */etc/eraraid.conf.drive*. If no drive is selected, get configuration from all drives.
-p	--print	Print all configuration files from drives. If no drive is selected, print files from all drives.

Example: Restoring the configuration file from the drives:

- To read configuration files from the drives, run:

```
# eraraid config -d
```

- To restore the configuration file, run:

```
# eraraid config -r /etc/eraraid.conf.drive
```

1.4 Drives

Manual Drive Replacement or Excluding

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

```
# eraraid replace -n <raid_name> -no <drive_number>
-d <new_block_device>
```

> If you manually replace a drive that is part of a sparepool, the drive is removed from the sparepool.

Description of the "replace" command parameters

Required parameters		
`-n`	`--name`	RAID name.
`-no`	`--number`	Drive number in the RAID. To see the drive number, use the command `eraraid show`.
`-d`	`--drive`	New block device. To exclude the drive or mark it as missing, set the path to the drive as "**null**".

Example: In the RAID "era5", replacing the drive "0" with the drive "nvme4n1":

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

```
# eraraid replace -n era5 -no 0 -d null
```

2. Replace the drive "0" with the drive "nvme4n1":

```
# eraraid replace -n era5 -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 sparepool, then assign the created sparepool to the RAID. You can only assign one sparepool to each RAID. One sparepool can have only SSDs or only NVMe drives.

To manage sparepools, run the command

```
# eraraid pool {-n <pool_name> {-d (<drives>) {--create|--add-drives|--delete-drives}|--delete}|--set-replace-delay <delay>}
```

The "pool" command parameters

One of the following mutually exclusive parameters is required		
	`--create`	Create a sperepool.
	`--delete`	Delete a sparepool.
	`--add-drives`	Add drives to a sparepool.
	`--delete-drives`	Remove drives from a sparepool.
	`--set-replace-delay`	Set delay timer (in seconds) for drive replacement from the sparepools. The timer works for all sparepools in the sustem. The default: 180 seconds.
Optional parameters		
`-n`	`--name`	Name of a saprepool. Use with the parameters `--create`, `--delete`, `--add-drives`, `--delete-drives`.
`-d`	`--drives`	The list of block devices, comma-separated. Use with the parameters `--create`, `--add-drives` и `--delete-drives`.

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

Example: Creating a sparepool "pool1" and assigning it to the RAID "era5":

1. Create a sparepool:

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

2. Assign the created sparepool to the RAID:

```
# eraraid modify -n era5 -sp pool1
```

Example: Setting replacement timer for sparepools to 60 seconds:

```
# eraraid pool --set-replace-delay 60
```

Sparepool info

To view information about the sparepools, use the command

```
# eraraid pool-show
```

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 sparepool failed, the drive will not be used for replacement.

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 chapter Setting up Email Notifications) 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 SparePool. 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.

You can change the drive fault threshold value or reset current numbers of faults for drives with the command

```
# eraraid faulty-count {{-st|--set-threshold} <1..1000>}|
{{-r|--reset} (block_devices)}
```

Description of the "faulty-count" command parameters

Mutually exclusive required parameters		
`-st`	`--set-threshold`	Set the drive fault threshold value. Possible values: integers from **1** to **1000**. The default: **3**.
`-r`	`--reset`	List of block devices (separated by a space) to reset their current numbers of faults. The parameter without arguments resets current numbers of faults for all drives.

Example: Set the drive fault threshold value to 10:

```
# eraraid faulty-count -st 10
```

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

```
# eraraid faulty-count -r /dev/sd[a-b] /dev/sdd
```

To see the current drive fault threshold value or the current value of fault count for a particular block device, run

```
# eraraid faulty-count-show {{-ct|--current-threshold}|
{-d|--drives} (block_devices)}
```

Description of the "faulty-count-show" command parameters

Mutually exclusive required parameters		
`-ct`	`--current-threshold`	Show the current drive fault threshold value. The parameter doesn't require attributes.
`-d`	`--drives`	List of block devices (separated by a space) to show their current value of fault count. The parameter without arguments shows current values of fault count for all drives.
Optional parameter		
`-f`	`--format`	Output format: • **table** – as a table; • **json** – json; • **prettyjson** – human-readable json. The default: **table**.

Example: See the current drive fault threshold value:

```
# eraraid faulty-count-show -ct
```

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

```
# eraraid faulty-count -d /dev/sd[a-b] /dev/sdd
```

Deleting Drive Metadata

The command removes ERA RAID metadata and the current value of fault count on drive(s).

> Metadata doesn't remove from a drive that was removed from a RAID due to exceeding the I/O error threshold. The command only resets the current error count.
>
> To remove metadata from such drives, add a new drive to the RAID to replace the removed drive.

To remove metadata from the drives, run:

```
# eraraid drive-clean {-d|--drives} (block_devices)
```

Description of the "drive-clean" command parameter

Required parameter		
`-d`	`--drives`	List of drives to clean metadata and current value of fault count.

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

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

Managing Drive LED Indication and Scanning

You can manage the LED drive indication to correlate physical drives to their letters in the system, as well as to identify failed drives.

The indication management divides in two types:

- Automatic.

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

  You can set up the service with the command

  ```
  # eraraid scanner [--scan-interval <seconds>] [--scan-smart-health-interval <seconds>] [--leds-enabled {0|1}]
  ```

The "scanner" command parameters

Дополнительные параметры		
	`--scan-interval`	Drive scan interval, in seconds. The parameter affects the auto-start delay for RAID initialization, reconstruction, and restriping. Possible values: integers from 1. The default: **1**.
	`--scan-smart-health-interval`	S.M.A.R.T. drive health scan interval, in seconds. Possible values: integers from **1**. The default: **86400** (24 hours).
	`--leds-enabled`	Enable (**1**) or disable (**0**) automatic LED indication of drives in the system. The parameter doesn't affect the manual LED indication.

Example: Disable automatic LED indication of drives:

```
# eraraid scanner --leds-enabled 0
```

- Manual.

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

```
# eraraid locate -d {(block_devices)|null}
```

Description of the "locate" command parameter

Required parameter		
`-d`		Switch on the indication on selected drives or switch the indication off (the **null** value).

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

```
# eraraid locate -d /dev/nvme0n1 /dev/nvme1n1
```

1.5 Notifications

Error Messages

If a running command failed or have ran incorrectly, the system will display all error messages for the last 10 seconds. You can also view the error log (by default the last 10 errors) with the command

```
# eraraid error-log [-n <lines_count>]
```

Description of the "error-log" command parameter

Optional parameter		
`-n`	`--lines`	Output the last N lines. Possible values are from 1 to 100.

Example: Display the last 20 error messages:

```
# eraraid error-log -n 20
```

Collecting System Logs

Logs collecting may be useful to get information about the system state at particular time interval.

To collect logs, run

```
# eraraid logs {-c|--collect}
```

Description of the "logs" command parameter

Required parameter		
`-c`	`--collect`	Collect system logs and get it into a .tar.gz file.

After collecting complete, the message "ERA logs collected" shows and the file appears at the current path.

When asking RAIDIX support for assistance, attach the file to your email.

Setting up Email Notifications

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

To receive email notifications, add email addresses on which notifications will be sent.

```
# eraraid mail {--add-receiver <email>
<notification_level>|--remove-receiver <email>|--set-scanning-
interval <scan_interval>|--set-progress-notify-frequency
<progress_interval>|--config}
```

Description of the "mail" command parameters

Required arguments		
	`--add-receiver`	Add a receiver email and set its notification level. Possible levels are: • **info** – Info, Warning, and Error types • **warning** – Info and Warning types • **error** – Error type
	`--remove-receiver`	Delete a receiver email and stop notify it.

	`--set-scanning-interval`	Set up interval for scanning RAIDs and drives, in seconds. The default is 10 seconds.
	`--set-progress-notify-frequency`	Set up interval for initialization and reconstruction process, in minutes. The default is 10 minutes.
	`--config`	Show current notification settings, in prettyjson.

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		RAID *(...)* is degraded now
Reconstruction completed on RAID *(...)*		After reboot/crash, RAID *(...)* has restored in offline state
Reconstruction progress on RAID *(...)* is *(...)* percent		ERA RAID license expired
Reconstruction started on RAID *(...)*		ERA RAID 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 *(...)*
		Could not automatically replace drive *(...)* in RAID *(...)* because there was no suitable drive in SparePool *(...)*
		The *(...)* has reached a critical percentage used endurance indicator (90%). Current percentage used: *(...)*%.

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

```
# eraraid mail --add-receiver user2@email.com info
```

Example: Show all receivers of notifications:

```
# eraraid mail --config
```

```
{
      "scanning_interval_sec": 10,
      "progress_notification_frequency_min": 10,
      "receivers": {
                      "user1@email.com": "error",
                      "user2@email.com": "info"
                    }
}
```

1.6 Number of CPU Threads for eraraid Module

There are two ways to control the number of CPU threads in eraraid:

- the command `cpuignore`

  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 command `modprobe`

  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 change the number of threads for the eraraid module, use the command

```
# eraraid cpuignore (<cpus>)
```

where `<cpus>` is a comma-separated list of CPU IDs that will not participate in the eraraid module. The value "**null**" removes the restriction on using threads for eraraid.

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

```
# eraraid cpuignore 0-3,6
```

Example: Remove all restrictions on CPUs for the module:

```
# eraraid cpuignore null
```

To limit the number of threads in the eraraid module:

1. Unload the module from the kernel:

```
# rmmod eraraid
```

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

```
# modprobe eraraid 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.

To configure the eraraid 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 eraraid 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.

2. ERA RAID CONFIGURATION RECOMMENDATIONS

2.1 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 ERA RAID is 16 KiB (set by default).

2.2 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 allows 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 the `--sched-enabled` parameter when the access pattern is low-threaded.

--merge-enabled

The `--merge-enabled` parameter enables to improve 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-enabled` will decrease the system workload performance.

> Enable merge by the `--merge-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.

--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 the offline, unrecoverable state). To try to read available data, manually turn on the online mode for the RAID by running the command

```
# eraraid 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

> While a RAID is being restriped or reconstructed, the resync function is unavailable.

The functionality protects syndromic RAIDs (all but RAID 0 and RAID 1) from data loss caused by a write hole.

After an unclear system shutdown, a RAID re-initialization is started. All settings and information related to initialization also apply to resync.

To disable resync for all RAIDs, run

```
# modprobe eraraid 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	Amount of RAM
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 ERA RAID, you can use network adapter of the same NUMA node as NVMe drives.

System

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

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

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

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

   Command output example:

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

   To check if HT is enabled, run

   ```
   # lscpu
   ```

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

   Command output example:

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

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

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

Workload

In ERA RAID, 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 parameter depending on the access pattern recommendations are provided below.

Merge

1. If the access pattern is sequential and block sizes are less than *stripe_data_len*, you can activate Merge requests with small block size in order to avoid part of *read-modify-write* operations. To activate Merge, use the `--merge-enabled` parameter (1 – activated, 0 – deactivated).
2. Deactivate merge when queue depth of user's workload is not enough to Merge a full stripe. Activate Merge, if

   iodepth * block_size >= stripe_data_len

   where

   stripe_data_len = num_data_strips * stripe_size.

   For example, for RAID 6 out of 6 drives and strip_size =16 KiB, stripe_data_len = 4 * 16 KiB = 64 KiB.

NVMe-oF

1. ERA RAID 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.raidix12_1 -a 10.30.0.12 -s 4420
   --ctrl-loss-tmo=0
   ```

2. At the creation of NVMe-oF target for ERA 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.

2.3 File System Mounting Aspects

Since the system restores RAIDIX ERA 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

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

- Requires = eraraid-restore.service
- After = eraraid-restore.service

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

1. Set a timeout of 5 minutes for the ERA device in the unit file:
   1.1. Run

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

   1.2. Add the following lines:

   ```
   [Unit]

   JobRunningTimeoutSec=5m
   ```

   Save the changes.
   1.3. Check the changes:

   ```
   #systemctl cat /dev/era_raid
   ```

2. Create a file (for example, mnt-raid.mount) at */etc/systemd/system/*.

The file */etc/systemd/system/mnt-raid.mount*

```
[Unit]
Description=Mount filesystem on RAIDIX ERA
Requires=eraraid-restore.service
After=eraraid-restore.service
DefaultDependencies=no
Before=umount.target
Conflicts=umount.target

[Mount]
What=/dev/era_raid
Where=/mnt/raid/
Options=defaults
Type=xfs

[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=eraraid-restore.service,x-systemd.device-timeout=5m,_netdev
- x-systemd.requires=eraraid-restore.service,x-systemd.device-timeout=5m,nofail

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* 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* disables reporting errors for this device if it does not exist.

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

The string from the fle */etc/fstab*

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

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

The string from the fle */etc/fstab*

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

3. DKMS SPECIFICS WHEN UPDATING LINUX KERNEL

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

The kernel module *eraraid* uses DKMS (Dynamic Kernel Module Support) technology and is automatically built and is installed for the Linux kernel versions, listed in the document *RAIDIX ERA 3.4.0 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 kernel module *eraraid* 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 *eraraid* 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);
- kernel-uek-devel (for Oracle Linux);
- 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 *RAIDIX ERA 3.4.0 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
```

4. TROUBLESHOOTING

Error on applying license on OS Debian 9:

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

More indicators:

- `dmesg` has these messages:

  ```
  eraraid: alg: akcipher: Failed to load tfm for rsa: -2

  eraraid: Cannot decrypt license key
  ```

- The file /boot/config-$(uname -r) has the line *CONFIG_CRYPTO_RSA=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 driver *rsa_generic*:

   ```
   # modprobe rsa-generic
   ```

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

Error: Missing ERA 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 *RAIDIX ERA 3.4.0 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 eraraid -v 3.4.0 --force
```