CONTENTS

Introduction ........................................................................................................... 4
  Intended Audience ............................................................................................ 4
  Guide Conventions ............................................................................................ 4

About RAIDIX ERA 3.3. ........................................................................................ 5
  RAIDIX ERA 3.3 Specifications ..................................................................... 5

1. ERA RAID Management .................................................................................. 6
  1.1 Command Line Interface (CLI) Description ............................................. 6
  1.2 License Management .................................................................................. 7
  1.3 RAID ............................................................................................................ 9
    Creating a RAID ............................................................................................... 9
    Showing RAID State ...................................................................................... 11
    Restoring a RAID ............................................................................................ 14
    Deleting a RAID .............................................................................................. 14
    Unloading a RAID ........................................................................................... 15
    RAID Reconstruction ...................................................................................... 15
    RAID Initialization .......................................................................................... 16
    RAID Restripping ............................................................................................ 16
    Changing RAID Parameters ......................................................................... 19
    Importing RAID Operations ......................................................................... 20
    Managing RAID Configurations ................................................................... 24
  1.4 Drives ........................................................................................................... 24
    Manual Drive Replacement or Excluding ...................................................... 24
    Automatic Drive Replacement ...................................................................... 25
    Deleting Drive Metadata ............................................................................... 26
    Managing Drive LED Indication and Scanning ............................................ 27
  1.5 Notifications ................................................................................................ 28
    Error Messages ............................................................................................... 28
    Collecting System Logs .................................................................................. 28
    Setting up Email Notifications .................................................................... 29
  1.6 Number of CPU Threads for eraraid Module ........................................... 30

2. ERA RAID Configuration Recommendations ................................................. 33
  2.1 RAID Creation ............................................................................................. 33
  2.2 RAID Usage ................................................................................................ 33
2.3  Recommendations for RAID Parameter Setup..........................................................34
2.4  File System Mounting Aspects ..............................................................................37
      systemd.mount....................................................................................................37
      /etc/fstab..........................................................................................................38
3.   DKMS Specifics when Updating Linux Kernel .......................................................39
4.   Troubleshooting.......................................................................................................40
INTRODUCTION

Intended Audience

This guide is intended for administrators and users of RAIDs based on the RAIDIX ERA 3.3 software. The guide contains instructions on how to configure and manage RAIDs in RAIDIX ERA 3.3.

Guide Conventions

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

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

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 3.3

RAIDIX ERA 3.3 is high-performance software RAID developed specifically for NVMe storage devices and new types of SAN networks. RAIDIX ERA 3.3 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 3.3 is a Linux kernel module and a management utility, which are built and configured for the most popular distributions (see the document RAIDIX ERA 3.3 System Requirements). The software is installed on servers with slots for Flash memory devices or with connected JBOFs. RAIDIX ERA 3.3 enables you to combine drives into high-performance fault-tolerant RAIDs.

RAIDIX ERA 3.3 Specifications

<table>
<thead>
<tr>
<th>Supported RAID levels</th>
<th>0</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>N+M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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. ERARAID MANAGEMENT

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

1.1 Command Line Interface (CLI) Description

Conventions on CLI command syntax

<table>
<thead>
<tr>
<th>Item's format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Required item (object, method, parameter, attribute).</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td>Placeholder variable.</td>
</tr>
<tr>
<td>[item]</td>
<td>Optional item.</td>
</tr>
<tr>
<td>{item1</td>
<td>item2}</td>
</tr>
<tr>
<td>(items)</td>
<td>Multiple values for the argument.</td>
</tr>
</tbody>
</table>

In the CLI, enter commands in the following format:

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

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

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>Manage a configuration file.</td>
</tr>
<tr>
<td>cpuignore</td>
<td>Manage number of PCU threads in eraraid.</td>
</tr>
<tr>
<td>create</td>
<td>Create a RAID.</td>
</tr>
<tr>
<td>destroy</td>
<td>Delete a RAID completely.</td>
</tr>
<tr>
<td>drive-clean</td>
<td>Clean a drive metadata.</td>
</tr>
<tr>
<td>error-log</td>
<td>Show errors in ERA RAID.</td>
</tr>
<tr>
<td>import</td>
<td>Import the configuration of a selected RAID from the configuration file to</td>
</tr>
<tr>
<td></td>
<td>selected drives.</td>
</tr>
<tr>
<td>import-all</td>
<td>Import all possible RAIDs.</td>
</tr>
<tr>
<td>import-show</td>
<td>Show importable RAIDs.</td>
</tr>
<tr>
<td>init</td>
<td>Manage RAID initialization.</td>
</tr>
<tr>
<td>license</td>
<td>Manage a license.</td>
</tr>
<tr>
<td>locate</td>
<td>Manage LED indication of drives.</td>
</tr>
<tr>
<td>logs</td>
<td>Collect system logs.</td>
</tr>
<tr>
<td>mail</td>
<td>Set up mail notifications.</td>
</tr>
<tr>
<td>modify</td>
<td>Modify RAID parameters.</td>
</tr>
<tr>
<td>pool</td>
<td>Manage sparepools for automatic drive replacement in RAIDs.</td>
</tr>
<tr>
<td>pool-show</td>
<td>Show sparepool info.</td>
</tr>
<tr>
<td>recon</td>
<td>Manage RAID reconstruction.</td>
</tr>
</tbody>
</table>
replace  Replace or delete a RAID drive.
restore  Restore a RAID.
restripe  Manage RAID restriping.
scannner  Controls the parameters of the disk presence and S.M.A.R.T. health scanning intervals, as well as the parameters of the automatic LED indication of drives.
show  Show information on a RAID.
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:
Command output example when a license was added:

```
[root@eratestl ~]# erraid license --show
Kernel version: 3.10.0-957.27.2.el7.x86_64
hwkey: 2EE10A9FL12DD2662
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
```

Description of the "license" command output

<table>
<thead>
<tr>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel version</td>
<td>Kernel version.</td>
</tr>
<tr>
<td>hwkey</td>
<td>Hardware key.</td>
</tr>
<tr>
<td>license_key</td>
<td>License key.</td>
</tr>
<tr>
<td>version</td>
<td>Software version.</td>
</tr>
<tr>
<td>crypto_version</td>
<td>Version of crypto-API for the license generator.</td>
</tr>
<tr>
<td>created</td>
<td>The date when the license was created.</td>
</tr>
<tr>
<td>expired</td>
<td>License expire date.</td>
</tr>
<tr>
<td>disks</td>
<td>Maximum number of drives.</td>
</tr>
<tr>
<td>levels</td>
<td>Maximum RAID level.</td>
</tr>
<tr>
<td>type</td>
<td>Drive type.</td>
</tr>
<tr>
<td>disks in use</td>
<td>Number of used drives in the system.</td>
</tr>
<tr>
<td>status</td>
<td>License state.</td>
</tr>
</tbody>
</table>
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](mailto:support@raidix.com).

After you got your license file, copy it to the server (for example, to the /root 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

<table>
<thead>
<tr>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-s --show</td>
</tr>
<tr>
<td>-u --update</td>
</tr>
<tr>
<td>-r --reset</td>
</tr>
</tbody>
</table>

### 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 <raid_name> -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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n --name</td>
<td>RAID name.</td>
</tr>
<tr>
<td>-l --level</td>
<td>RAID level: 0, 1, 5, 6, 7, 10, 50, 60, 70, or N+M.</td>
</tr>
<tr>
<td>-d --drives</td>
<td>Space-separated list of block devices.</td>
</tr>
<tr>
<td>-gs --group_size</td>
<td>Only for RAID 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.</td>
</tr>
<tr>
<td>-bs --block-size</td>
<td>RAID block size: 512 or 4096 bytes.</td>
</tr>
<tr>
<td>-sc --synd-cnt</td>
<td>Only for RAID N+M. Number of syndromes M. Possible values are integers from 4 to 32. Additional conditions: N+M ≤ 64 and M ≤ N.</td>
</tr>
</tbody>
</table>

### Optional parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ss --strip-size</td>
<td>Strip size in KiB. Possible values are 16, 32, 64, 128, or 256. The default is 16.</td>
</tr>
<tr>
<td>-ip --init-prio</td>
<td>Initialization priority. Possible values are from 0 to 100%. The default is 100.</td>
</tr>
<tr>
<td>-rp --recon-prio</td>
<td>Reconstruction priority. Possible values are from 0 to 100%. The default is 100.</td>
</tr>
<tr>
<td>-me --merge-enabled</td>
<td>Activate (1) or deactivate (0) the Merge parameter. The default is 0.</td>
</tr>
<tr>
<td>-se --sched-enabled</td>
<td>Activate (1) or deactivate (0) the Scheduling parameter. The default is 0.</td>
</tr>
<tr>
<td>-sp --sparepool</td>
<td>Name of a sparepool that is assigned to the RAID. The value “null” removes the sparepool from the RAID.</td>
</tr>
<tr>
<td>-ml --memory-limit</td>
<td>RAM usage limit in MiB. The default is 0 (unlimited).</td>
</tr>
<tr>
<td>-rl --request-limit</td>
<td>Activate (1) or deactivate (0) limitation of requests per RAID. The default is 0.</td>
</tr>
<tr>
<td>-mw --merge-wait</td>
<td>The waiting time (in microseconds) between requests when Merge is enabled. The default: 300.</td>
</tr>
<tr>
<td>-mm --merge-max</td>
<td>Maximum waiting time (in microseconds) for stripe accumulation with Merge enabled. The default: 1000.</td>
</tr>
</tbody>
</table>

---

1 See the chapter “Set up RAID Parameters. Recommendations”
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.

```bash
# 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.

Warning: 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:

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

Description of the "show" command parameters

<table>
<thead>
<tr>
<th>Optional parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n --name</td>
<td>RAID name.</td>
</tr>
<tr>
<td>-o --online</td>
<td>Show only RAIDs, which weren't unloaded via the <code>unload</code> command.</td>
</tr>
<tr>
<td>-u --units</td>
<td>Size units: <code>s</code> – sector, <code>k</code> – kilobyte, <code>m</code> – megabyte, <code>g</code> – gigabyte.</td>
</tr>
<tr>
<td>-f --format</td>
<td>Output format:</td>
</tr>
<tr>
<td></td>
<td>• <code>table</code> – as a table;</td>
</tr>
<tr>
<td></td>
<td>• <code>json</code> – json;</td>
</tr>
<tr>
<td></td>
<td>• <code>prettyjson</code> – human-readable json.</td>
</tr>
<tr>
<td>-e --extended</td>
<td>Show the extended table</td>
</tr>
</tbody>
</table>

Example: Showing information on the RAID "era5":

```bash
# eraraid show -n era5 -e
```
### Description of the “show” command output

<table>
<thead>
<tr>
<th>Row</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>RAID name.</td>
</tr>
<tr>
<td>static</td>
<td>Static RAID parameters:</td>
</tr>
<tr>
<td></td>
<td>• size.</td>
</tr>
<tr>
<td></td>
<td>• level.</td>
</tr>
<tr>
<td></td>
<td>• synd_cnt – only for RAIDs N+M – number of syndromes.</td>
</tr>
<tr>
<td></td>
<td>• block_size – RAID block size.</td>
</tr>
<tr>
<td></td>
<td>• group_size – only for RAIDs 50, 60, and 70 – size of the corresponding RAID group.</td>
</tr>
<tr>
<td></td>
<td>• strip_size.</td>
</tr>
<tr>
<td></td>
<td>• sparepool – name of the assigned sparepool.</td>
</tr>
<tr>
<td></td>
<td>• active:</td>
</tr>
<tr>
<td></td>
<td>- True, if the RAID’s block device is in the system.</td>
</tr>
<tr>
<td></td>
<td>- False, if:</td>
</tr>
<tr>
<td></td>
<td>- The RAID was not loaded after reboot. Restore the RAID.</td>
</tr>
<tr>
<td></td>
<td>- The RAID is unloaded. Restore the RAID.</td>
</tr>
<tr>
<td></td>
<td>• config:</td>
</tr>
<tr>
<td></td>
<td>- True, if the RAID is in the configuration file.</td>
</tr>
<tr>
<td></td>
<td>- False, if the RAID is missing.</td>
</tr>
<tr>
<td>Row</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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. |
<table>
<thead>
<tr>
<th>Row</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>To show, use the command with the (-e) parameter.</td>
</tr>
<tr>
<td></td>
<td><strong>Editable RAID parameters:</strong></td>
</tr>
<tr>
<td></td>
<td>- init_prio – (except RAID 0) initialization priority: from 0% to 100%.</td>
</tr>
<tr>
<td></td>
<td>- init_depth – the queue depth of initialization requests.</td>
</tr>
<tr>
<td></td>
<td>- recon_prio – (except RAID 0) reconstruction priority: from 0% to 100%.</td>
</tr>
<tr>
<td></td>
<td>- recon_depth – the queue depth of reconstruction requests.</td>
</tr>
<tr>
<td></td>
<td>- memory_limit_mb – the value limited RAM usage, in megabytes.</td>
</tr>
<tr>
<td></td>
<td>- merge-enabled – is the parameter Merge enabled (1) or disabled (0).</td>
</tr>
<tr>
<td></td>
<td>- sched-enabled – is the parameter Scheduling enabled (1) or disabled (0).</td>
</tr>
<tr>
<td></td>
<td>- req_limit_enabled – is the parameter Request-limit enabled (1) or</td>
</tr>
<tr>
<td></td>
<td>disabled (0).</td>
</tr>
<tr>
<td></td>
<td>- restripe_prio – priority for restriping: from 0% to 100%.</td>
</tr>
<tr>
<td></td>
<td>- merge_wait_usecs – waiting time between requests when Merge is</td>
</tr>
<tr>
<td></td>
<td>enabled.</td>
</tr>
<tr>
<td></td>
<td>- merge_max_usecs – maximum time to wait for stripe accumulation with</td>
</tr>
<tr>
<td></td>
<td>Merge enabled.</td>
</tr>
<tr>
<td>info</td>
<td><strong>Dynamic RAID values:</strong></td>
</tr>
<tr>
<td></td>
<td>- init_progress – initialization progress: from 0% to 100%.</td>
</tr>
<tr>
<td></td>
<td>- recon_progress – reconstruction progress: from 0% to 100%.</td>
</tr>
<tr>
<td></td>
<td>- memory_usage_mb – amount of RAM usage; if memory_limit_mb = 0 (not</td>
</tr>
<tr>
<td></td>
<td>limited), then memory_usage_mb is not displayed.</td>
</tr>
<tr>
<td></td>
<td>- restripe_progress – restriping progress: from 0% to 100%.</td>
</tr>
</tbody>
</table>

**Restoring a RAID**

RAID restores automatically after any failure. If any problem arise, restore the RAID manually by running the command:

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

Description of the “restore” command parameters

### Required parameters

<table>
<thead>
<tr>
<th>Required parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n</code> or <code>--name</code></td>
<td>RAID name.</td>
</tr>
<tr>
<td><code>-a</code> or <code>--all</code></td>
<td>Restore all RAIDs.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n</code> --name</td>
</tr>
<tr>
<td><code>-a</code> --all</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n</code> --name</td>
</tr>
<tr>
<td><code>-a</code> --all</td>
</tr>
</tbody>
</table>

**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.

To manage the reconstruction process, run:

```
# eraraid recon -n <raid_name> {--start |--stop}
```
Description of the "recon" command parameters

<table>
<thead>
<tr>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n  --name</td>
</tr>
<tr>
<td>--start</td>
</tr>
<tr>
<td>--stop</td>
</tr>
</tbody>
</table>

**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.

To manage the initialization process, run:

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

Description of the "init" command parameters

<table>
<thead>
<tr>
<th>Required parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n  --name</td>
</tr>
<tr>
<td>--start</td>
</tr>
<tr>
<td>--stop</td>
</tr>
</tbody>
</table>

**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 is 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:

<table>
<thead>
<tr>
<th>RAID level change</th>
<th>Minimal number of drives you should add</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID 0 to RAID 1</td>
<td>RAID 0 must contain only 1 drive.</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>RAID 0 to RAID 10</td>
<td>If a RAID 0 contains only 1 drive:</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>If a RAID 0 contains more than 1 drive:</td>
</tr>
<tr>
<td></td>
<td>The number of drives to be added must be equal to the number of drives in the RAID 0.</td>
</tr>
<tr>
<td>RAID 1 to RAID 10</td>
<td>2</td>
</tr>
<tr>
<td>RAID 1 to RAID 5</td>
<td>2</td>
</tr>
<tr>
<td>RAID 10 to RAID 5</td>
<td>1</td>
</tr>
<tr>
<td>RAID 5 to RAID 6</td>
<td>1</td>
</tr>
<tr>
<td>RAID 6 to RAID 7.3</td>
<td>1</td>
</tr>
</tbody>
</table>

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

To restripe, run

```bash
# 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**

<table>
<thead>
<tr>
<th>Required parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n</code> --name</td>
<td>RAID name.</td>
</tr>
</tbody>
</table>
| `-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/sgd, /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/sgd
   ```
# 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
Description of the "modify" command parameters

<table>
<thead>
<tr>
<th>Required parameter</th>
<th>Optional parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n --name</td>
<td>RAID name.</td>
</tr>
</tbody>
</table>
| -ip --init-prio    | Initialization priority.  
|                    | Possible values are from 0% to 100% (maximum rate of initialization). |
| -rp --recon-prio   | 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| Activate (1) or deactivate (0) Merge. |
| -se --sched-enabled| Activate (1) or deactivate (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. |

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

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

**Importing RAID Operations**

RAIDIX ERA 3.3 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" state.
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

<table>
<thead>
<tr>
<th>Optional parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-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

<table>
<thead>
<tr>
<th>Optional parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-d</code> <strong>--drives</strong></td>
<td>Import RAIDs from specified drives.</td>
</tr>
</tbody>
</table>

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

All rights reserved © RAIDIX, 2021
Description of the "import" command parameters

<table>
<thead>
<tr>
<th>Required parameter</th>
<th>Optional parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-id --uuid</td>
<td>RAID UUID.</td>
</tr>
<tr>
<td>-n --name</td>
<td>New RAID name.</td>
</tr>
<tr>
<td>-d --drives</td>
<td>Import RAIDs from selected drives.</td>
</tr>
</tbody>
</table>

**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:

![RAID Information](image)

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
```

![System RAID Information](image)
To import all non-conflict RAIDs, run:

```
# eraraid import-all
```

<table>
<thead>
<tr>
<th>RAIDs name</th>
<th>static</th>
<th>state</th>
<th>devices</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>era1</td>
<td></td>
<td>online</td>
<td>0 /dev/sdb online 1 /dev/sdn online 2 /dev/sdc online</td>
<td>recon_progress : 0 memory_usage_mb: -</td>
</tr>
<tr>
<td></td>
<td>size: 4 GiB level: 1 strip_size: 16 active: True config: True</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
Managing RAID Configurations

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

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

Description of the “config” command parameters

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

**Example:** Restoring the configuration file from the drives:

- To read configuration files from the drives, run:

  ```bash
  # eraraid config -d
  ```

- To restore the configuration file, run:

  ```bash
  # 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:

```bash
# 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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n --name</td>
<td>RAID name.</td>
</tr>
<tr>
<td>-no --number</td>
<td>Drive number in the RAID. To see the drive number, use the command eraraid show.</td>
</tr>
<tr>
<td>-d --drive</td>
<td>New block device. To exclude the drive or mark it as missing, set the path to the drive as &quot;null&quot;.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--create</td>
<td>Create a sparepool.</td>
</tr>
<tr>
<td>--delete</td>
<td>Delete a sparepool.</td>
</tr>
<tr>
<td>--add-drives</td>
<td>Add drives to a sparepool.</td>
</tr>
<tr>
<td>--delete-drives</td>
<td>Remove drives from a sparepool.</td>
</tr>
<tr>
<td>--set-replace-delay</td>
<td>Set delay timer (in seconds) for drive replacement from the sparepools. The timer works for all sparepools in the system. The default: 180 seconds.</td>
</tr>
</tbody>
</table>

Optional parameters

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n --name</td>
<td>Name of a sparepool. Use with the parameters --create, --delete, --add-drives, --delete-drives.</td>
</tr>
</tbody>
</table>
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
```

Possible drive states:

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

**Deleting Drive Metadata**

To remove metadata from the drives, run:

```
# eraraid drive-clean {-d|--drives} (block_devices)
```
Description of the "drive-clean" command parameter

**Required parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d</td>
<td>List of drives to clean metadata.</td>
</tr>
</tbody>
</table>

**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 disks 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 disks 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--scan-interval</td>
<td>Drive scan interval, in seconds.</td>
</tr>
<tr>
<td></td>
<td>The parameter affects the auto-start delay for RAID initialization, reconstruction, and restriping.</td>
</tr>
<tr>
<td></td>
<td>Possible values: integers from 1.</td>
</tr>
<tr>
<td></td>
<td>The default: 1.</td>
</tr>
<tr>
<td>--scan-smart-health-interval</td>
<td>S.M.A.R.T. drive health scan interval, in seconds.</td>
</tr>
<tr>
<td></td>
<td>Possible values: integers from 1.</td>
</tr>
<tr>
<td></td>
<td>The default: 86400 (24 hours).</td>
</tr>
<tr>
<td>--leds-enabled</td>
<td>Enable (1) or disable (0) automatic LED indication of drives in the system.</td>
</tr>
<tr>
<td></td>
<td>The parameter doesn't affect the manual LED indication.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Required parameter</th>
</tr>
</thead>
</table>
| `-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

<table>
<thead>
<tr>
<th>Optional parameter</th>
</tr>
</thead>
</table>
| `-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

<table>
<thead>
<tr>
<th>Required parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c</code> <code>--collect</code></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Required arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--add-receiver</td>
<td>Add a receiver email and set its notification level. Possible levels are:</td>
</tr>
<tr>
<td></td>
<td>• info – Info, Warning, and Error types</td>
</tr>
<tr>
<td></td>
<td>• warning – Info and Warning types</td>
</tr>
<tr>
<td></td>
<td>• error – Error type</td>
</tr>
<tr>
<td>--remove-receiver</td>
<td>Delete a receiver email and stop notify it.</td>
</tr>
<tr>
<td>--set-scanning-interval</td>
<td>Set up interval for scanning RAIDs and drives, in seconds.</td>
</tr>
<tr>
<td></td>
<td>The default is 10 seconds.</td>
</tr>
<tr>
<td>--set-progress-notify-frequency</td>
<td>Set up interval for initialization and reconstruction process, in minutes.</td>
</tr>
<tr>
<td></td>
<td>The default is 10 minutes.</td>
</tr>
<tr>
<td>--config</td>
<td>Show current notification settings, in prettyjson.</td>
</tr>
</tbody>
</table>

**Notification types:**

<table>
<thead>
<tr>
<th>Info</th>
<th>Warning</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization completed on RAID (...)</td>
<td>Initialization not completed on RAID (...)</td>
<td>RAID (...) is offline now</td>
</tr>
<tr>
<td>Initialization progress on RAID (...) is (...) percent</td>
<td>RAID (...) is read-only now</td>
<td>RAID (...) is unrecovered now</td>
</tr>
<tr>
<td>Initialization started on RAID (...)</td>
<td>System is up after reboot/crash</td>
<td>After reboot/crash, RAID (...) not restored</td>
</tr>
<tr>
<td>RAID (...) is healthy now</td>
<td>Reconstruction not completed on RAID (...)</td>
<td>After reboot/crash, RAID (...) has restored in read only mode</td>
</tr>
</tbody>
</table>
### Info

- RAID (... is online now
- Reconstruction completed on RAID (...
- Reconstruction progress on RAID (...
- Reconstruction started on RAID (...
- Drive (... was returned to RAID (...
- Drive (... from SparePool (...
- Drive (... in RAID (...
- Drive (... in RAID (...
- Drive (... from SparePool (...
- Drive (... from SparePool (...
- Drive (... in RAID (...
- Drive (... in RAID (...
- Drive (... from SparePool (...
- Drive (... from SparePool (...
- The (...) has reached a critical percentage used endurance indicator (90%). Current percentage used: (...)%.

### Warning

- RAID (... is degraded now
- After reboot/crash, RAID (...
- ERA RAID license expired
- ERA RAID license error: number of disks in use exceeds the allowed disks number
- Drive (... in RAID (...
- Drive (... from SparePool (...
- Could not automatically replace drive (... in RAID (...
- Could not automatically replace drive (... in RAID (...

### Error

- Drive (... is offline now
- Drive (... was disconnected
- Drive (... in RAID (...
- Could not automatically replace drive (... in RAID (...
- There was no suitable drive in SparePool (...)

---

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

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

**Example:** Show all receivers of notifications:

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

```json
{
    "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

```bash
# 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:

   ```bash
   # rmmod eraraid
   ```

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

   ```bash
   # 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

```bash
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 Usage

Init-prio, Recon-prio, and Restripe-prio

The modify command allows to change initialization, reconstruction, and restriping priorities. If the priority value of the corresponding function is zero, reconstruction, initialization, or restriping starts and continues only if there is no workload. By default, all priorities are set to 100% that stands for the highest possible rate of reconstruction, initialization, or restriping process. To improve the system workload performance, try decreasing initialization and reconstruction priorities.

Scheduling

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.

Recommendation: Enable the --sched-enabled parameter when the access pattern is low threaded.

Merge

The --merge-enabled parameter allows 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.

Recommendation: Use 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 amount of incoming requests per RAID. For example, writing files on file system without synchronization.

Recommendation: To improve system workload performance, enable the parameter when you are working with file system and the writing is performed with the direct parameter equals to 0.

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.

2.3 Recommendations for RAID Parameter Setup

1. 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.

2. 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.

3. Recommended RAID strip size is 16 KiB.

4. 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.

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:

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

   or via `lspci`:

   ```bash
   # 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

   ```bash
   # 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 bm1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat umip arch_capabilities
   ```

   To check if HT is enabled, run

   ```bash
   # lscpu
   ```

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

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture:</td>
<td>x86_64</td>
</tr>
<tr>
<td>CPU op-mode(s):</td>
<td>32-bit, 64-bit</td>
</tr>
<tr>
<td>Byte Order:</td>
<td>Little Endian</td>
</tr>
<tr>
<td>Address sizes:</td>
<td>40 bits physical, 48 bits virtual</td>
</tr>
<tr>
<td>CPU(s):</td>
<td>4</td>
</tr>
<tr>
<td>On-line CPU(s) list:</td>
<td>0-3</td>
</tr>
<tr>
<td>Thread(s) per core:</td>
<td>1</td>
</tr>
</tbody>
</table>

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

```bash
# 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. Starting from RAIDIX ERA 3.3, 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.
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.

2.4 File System Mounting Aspects

Since the system restores RAIDX ERA RAIDs after loading an appropriate Linux core and sending a RAID-restore command, to perform an automounting of file systems on these RAIDs, use the following instructions.

**systemd.mount**

When automounting 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:

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

```ini
[Unit]
    Description = Mount filesystem on RAIDX ERA
    Requires = eraraid-restore.service
    After = eraraid-restore.service

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

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

Before work with the mount service, after creating the /etc/systemd/system/mnt-raid.mount file, run the command

```
# systemctl daemon-reload
```

Enable automatic mounting at system startup:
# systemctl enable mnt-raid.mount

Start the service to mount the device:

    # systemctl start mnt-raid.mount

/etc/fstab

When automounting via /etc/fstab, point out the following option:

- `x-systemd.automount` — mounts a file system on first access after all system services started (including the device recovering service RAIDIX ERA).

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

The string from the file /etc/fstab

<table>
<thead>
<tr>
<th>dev/era_raid</th>
<th>/mnt/raid/</th>
<th>xfs</th>
<th>x-systemd.after=eraraid-restore.service</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>


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 3.3 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:

* 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 3.2 Installation Guide).

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

```bash
# 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 3.3
  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 3.3.0 --force