Mdadm Library - Device Linearization API in C

Why use our Mdadm API?

- **Device Linearization:** If you have multiple disk devices and you are worrying about memory management and storage capacity, this can be solved through our simple API that will linearize all your devices' storage space.

- **Cache Functionality:** Our querying capacity includes a cache functionality for a low-latency solution for reading and writing to your device spaces.

- **High Avalibility:** We are running thousands of backup servers that will make you confident that the API functionalities we offer are highly reliable and always ready for usage. Our API has beyond-believable flexibility.

API Functions

mdadm_mount

- ➢ Parameters: none
- ➢ Usage: Calling this function will mount all your devices.
- ➢ Potential Error: Calling mdadm_mount while already mounted returns an error
- ➢ Return Type: -1 on fail, 1 on success.

mdadm_unmount

- ➢ Parameters: none
- ➢ Usage: Calling this function will unmount all your devices.
- ➢ Potential Error: Calling mdadm_unmount while not mounted returns an error
- ➢ Return Type: -1 on fail, 1 on success.

mdadm_read

- ➢ Parameters:
  - *uint32_t addr* - The starting address across your devices to start reading from.
  - *uint32_t len* - The amount of bytes to be read starting at addr position.
  - *uint8_t *buf* - The content of the read bytes will be here after function call.
- ➢ Usage: Calling this function will read len amount of bytes starting at the specified addr position, and then will put hte content to the buffer pointer.

- ➢ An Error will be thrown if:
    - Devices are unmounted
    - The buffer pointer is NULL
    - len + addr is above maximum capacity (currently: 1048576 bytes)
    - Len to be read is too large. (too large is bigger than 1024 bytes)
- ➢ Return Type: 0 if len is 0, -1 on failure, len on success.

mdadm_write

- ➢ Parameters:
    - *uint32_t addr* - The starting address across your devices to start writing from.
    - *uint32_t len* - The amount of bytes to be written from the starting address
    - *uint8_t *buf* - The content to be written into the devices.
- ➢ Usage: Calling this function will write len bytes from buf starting at the address location specified by addr into the devices.
- ➢ An Error will be thrown if:
    - Devices are unmounted
    - The buffer pointer is NULL
    - len + addr is above maximum capacity (currently: 1048576 bytes)
    - Len to be read is too large. (too large is bigger than 1024 bytes)
- ➢ Return Type: 0 if len is 0, -1 on failure, len on success.

Additional Power Feature Functions

mdadm_cache_create

- ➢ Parameters:
    - *int num_entries* - The size of the cache you desire
- ➢ Usage: Calling this function will dynamically add a cache system to speed up your querying.
- ➢ An Error will be thrown if:
    - You already have a cache created
    - Your cache size is less than two or bigger than 4096
- ➢ Return Type: -1 on fail, 1 on success.

mdadm_cache_destroy

- ➢ Parameters: none
- ➢ Usage: Calling this function will free the dynamically allocated cache space and will delete the cache.
- ➢ An Error will be thrown if:
  - There is no cache currently active
- ➢ Return Type: -1 on fail, 1 on success.

mdadm_connect

- ➢ Parameters:
  - const char *ip - The IP address in ASCII to connect to the server
  - *uint16_t port* - The port to be connected to the server.
- ➢ Usage: Calling this function will connect to our highly reliable servers, ensuring that your calls will always come, even if world war three starts.
- ➢ An Error will be thrown if:
  - If the socket fails to create
  - If the sockaddr_in object failed to initialize
  - If it fails to connect to the server. (ha, it will never happen!)
- ➢ Return Type: A boolean false on failure, a boolean true on success

mdadm_disconnect

- ➢ Parameters: none
- ➢ Usage: When you wish to disconnect from our servers.
- ➢ Return Type: VOID (none)

Summary: The Mdadm Library provides an extremely simple to use Device Linearization API in C language that can linearize multiple disk devices' storage space, thus helping with memory management and storage capacity. The API includes cache functionality for low-latency solutions for reading and writing to device spaces. Additionally, the API's functionalities are highly reliable and flexible, with thousands of backup servers to ensure high availability.

We are committed to making the best application interfaces for you.

[EXAMPLE OF CLASSIC API USE BELOW]

API Example Usage

```
#include <stdio.h>

// include our package at the top of the file after downloading
#include <mdadm.h>

int main()
{
    // Mount all devices
    mdadm_mount();

    // Define buffer to read into
    uint8_t read_buffer[1024];

    // Read 512 bytes starting at address 0 and put content into buffer
    mdadm_read(0, 512, read_buffer);

    // Define buffer to write from
    uint8_t write_buffer[1024] = "This test string will be written starting at byte
512.";

    // Write the buffer content starting at address 512
    mdadm_write(512, 512, write_buffer);

    // Unmount all devices
    mdadm_unmount();

    return 0;
}
```