Capability Checking
Table of contents

Device Capability
Library Capability
Core Capability
An application that uses a DOCA Device relies on a subset of features for it to function as designed. As such, it is recommended to check whether these features exist for the selected DOCA Device. To achieve this, DOCA SDK exposes capabilities which are a set of APIs with a common look and feel, as described on this page.

The application is expected to use these capability APIs prior to any use of DOCA SDK APIs (Core, libraries) to fail as soon as possible (before initializing any resource) and to be able to implement fallback flows instead of getting error unexpectedly in the application flow.

**Device Capability**

An application that uses DOCA Core APIs may need to identify the specific DOCA Device to work based on specific capabilities.

For that, `doca_devinfo` and `doca_devinfo_rep` expose APIs with the prefix `doca_devinfo_cap_*`/`doca_devinfo_rep_cap_*`. For example:

```c
doca_error_t doca_devinfo_cap_is_hotplug_manager_supported(const struct doca_devinfo *devinfo, uint8_t *is_hotplug_manager);
doca_error_t doca_devinfo_rep_cap_is_filter_emulated_supported(const struct doca_devinfo *devinfo, uint8_t *filter_emulated_supported);
```

**Library Capability**

Each DOCA library exposes a set of capability APIs for the following purposes:

- Querying the maximum/minimum valid values of a configuration property of the library context or a library task
- Validating whether a library task is supported for a specific DOCA Device

All library capability API starts with the prefix `doca_<library_name>_cap_*`. Moreover:

- Configuration limitation capability APIs start with the prefix `doca_<library_name>_cap_[task_<task_type>]*get_min/max_*`
Task supported capability APIs have the naming template
doca_<library_name>_cap_task_<task_type>_is_supported

For example, DOCA DMA exposes:

doca_error_t doca_dma_cap_task_memcpy_is_supported(const struct doca_devinfo *devinfo);
doca_error_t doca_dma_cap_get_max_num_tasks(struct doca_dma *dma, uint32_t *max_num_tasks);
doca_error_t doca_dma_cap_task_memcpy_get_max_buf_size(const struct doca_devinfo *devinfo, uint64_t *buf_size);

Core Capability

Like any other DOCA library, each DOCA Core module also exposes capability APIs.

For example:

- A hotplug (of emulated PCIe functions) oriented application can check if a specific DOCA Device information structure enables hotplugging emulated devices, by calling:

  doca_error_t doca_devinfo_cap_is_hotplug_manager_supported(const struct doca_devinfo *devinfo, uint8_t *is_hotplug_manager);

- An application that works with DOCA mmap to be shared between the host and BlueField, must export the doca_mmap from the host and import it from BlueField. Before starting the workflow, the application can check if those operations are supported for a given a doca_devinfo using the following APIs:

  doca_error_t doca_mmap_cap_is_export_pci_supported(const struct doca_devinfo *devinfo, uint8_t *mmap_export);
doca_error_t doca_mmap_cap_is_create_from_export_pci_supported(const struct doca_devinfo *devinfo, uint8_t *from_export);