Devlink Info

The `devlink-info` mechanism enables device drivers to report device (hardware and firmware) information in a standard, extensible fashion.

The original motivation for the `devlink-info` API was twofold:

- making it possible to automate device and firmware management in a fleet of machines in a vendor-independent fashion (see also :ref:`Documentation/networking/devlink/devlink-flash.rst <devlink_flash>`);

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\(linux-master) (Documentation) (networking) (devlink)devlink-info.rst, line 12); *backlink*

  Unknown interpreted text role "ref".

- name the per component FW versions (as opposed to the crowded ethtool version string).

`devlink-info` supports reporting multiple types of objects. Reporting driver versions is generally discouraged - here, and via any other Linux API.

List of top level info objects

Name	Description
`driver`	Name of the currently used device driver, also available through sysfs.
`serial_number`	Serial number of the device. This is usually the serial number of the ASIC, also often available in PCI config space of the device in the *Device Serial Number* capability. The serial number should be unique per physical device. Sometimes the serial number of the device is only 48 bits long (the length of the Ethernet MAC address), and since PCI DSN is 64 bits long devices pad or encode additional information into the serial number. One example is adding port ID or PCI interface ID in the extra two bytes. Drivers should make sure to strip or normalize any such padding or interface ID, and report only the part of the serial number which uniquely identifies the hardware. In other words serial number reported for two ports of the same device or on two hosts of a multi-host device should be identical.
`board.serial_number`	Board serial number of the device. This is usually the serial number of the board, often available in PCI *Vital Product Data*.
`fixed`	Group for hardware identifiers, and versions of components which are not field-updatable. Versions in this section identify the device design. For example, component identifiers or the board version reported in the PCI VPD. Data in `devlink-info` should be broken into the smallest logical components, e.g. PCI VPD may concatenate various information to form the Part Number string, while in `devlink-info` all parts should be reported as separate items. This group must not contain any frequently changing identifiers, such as serial numbers. See :ref:`Documentation/networking/devlink/devlink-flash.rst <devlink_flash>` to understand why. System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\(linux-master) (Documentation) (networking) (devlink)devlink-info.rst, line 64); *backlink* Unknown interpreted text role "ref".
`running`	Group for information about currently running software/firmware. These versions often only update after a reboot, sometimes device reset.
`stored`	Group for software/firmware versions in device flash. Stored values must update to reflect changes in the flash even if reboot has not yet occurred. If device is not capable of updating `stored` versions when new software is flashed, it must not report them.

Each version can be reported at most once in each version group. Firmware components stored on the flash should feature in both the `running` and `stored` sections, if device is capable of reporting `stored` versions (see :ref:`Documentation/networking/devlink/devlink-flash.rst <devlink_flash>`). In case software/firmware components are loaded from the disk (e.g. `/lib/firmware`) only the running version should be reported via the kernel API.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\(linux-master)(Documentation)(networking)(devlink)devlink-info.rst, line 81); *backlink*

Unknown interpreted text role "ref".

Generic Versions

It is expected that drivers use the following generic names for exporting version information. If a generic name for a given component doesn't exist yet, driver authors should consult existing driver-specific versions and attempt reuse. As last resort, if a component is truly unique, using driver-specific names is allowed, but these should be documented in the driver-specific file.

All versions should try to use the following terminology:

List of common version suffixes

Name	Description
`id`, `revision`	Identifiers of designs and revision, mostly used for hardware versions.
`api`	Version of API between components. API items are usually of limited value to the user, and can be inferred from other versions by the vendor, so adding API versions is generally discouraged as noise.
`bundle_id`	Identifier of a distribution package which was flashed onto the device. This is an attribute of a firmware package which covers multiple versions for ease of managing firmware images (see :ref:`Documentation/networking/devlink/devlink-flash.rst <devlink_flash>`). System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\(linux-master)(Documentation)(networking)(devlink)devlink-info.rst, line 114); *backlink* Unknown interpreted text role "ref". `bundle_id` can appear in both `running` and `stored` versions, but it must not be reported if any of the components covered by the `bundle_id` was changed and no longer matches the version from the bundle.

board.id

Unique identifier of the board design.

board.rev

Board design revision.

asic.id

ASIC design identifier.

asic.rev

ASIC design revision/stepping.

board.manufacture

An identifier of the company or the facility which produced the part.

fw

Overall firmware version, often representing the collection of fw.mgmt, fw.app, etc.

fw.mgmt

Control unit firmware version. This firmware is responsible for house keeping tasks, PHY control etc. but not the packet-by-packet data path operation.

fw.mgmt.api

Firmware interface specification version of the software interfaces between driver and firmware.

fw.app

Data path microcode controlling high-speed packet processing.

fw.undi

UNDI software, may include the UEFI driver, firmware or both.

fw.ncsi

Version of the software responsible for supporting/handling the Network Controller Sideband Interface.

fw.psid

Unique identifier of the firmware parameter set. These are usually parameters of a particular board, defined at manufacturing time.

fw.roce

RoCE firmware version which is responsible for handling roce management.

fw.bundle_id

Unique identifier of the entire firmware bundle.

Future work

The following extensions could be useful:

- on-disk firmware file names - drivers list the file names of firmware they may need to load onto devices via the `MODULE_FIRMWARE()` macro. These, however, are per module, rather than per device. It'd be useful to list the names of firmware files the driver will try to load for a given device, in order of priority.