Devlink Flash

The `devlink-flash` API allows updating device firmware. It replaces the older `ethtool-flash` mechanism, and doesn't require taking any networking locks in the kernel to perform the flash update. Example use:

```
$ devlink dev flash pci/0000:05:00.0 file flash-boot.bin
```

Note that the file name is a path relative to the firmware loading path (usually `/lib/firmware/`). Drivers may send status updates to inform user space about the progress of the update operation.

Overwrite Mask

The `devlink-flash` command allows optionally specifying a mask indicating how the device should handle subsections of flash components when updating. This mask indicates the set of sections which are allowed to be overwritten.

List of overwrite mask bits

Name	Description
`DEVLINK_FLASH_OVERWRITE_SETTINGS`	Indicates that the device should overwrite settings in the components being updated with the settings found in the provided image.
`DEVLINK_FLASH_OVERWRITE_IDENTIFIERS`	Indicates that the device should overwrite identifiers in the components being updated with the identifiers found in the provided image. This includes MAC addresses, serial IDs, and similar device identifiers.

Multiple overwrite bits may be combined and requested together. If no bits are provided, it is expected that the device only update firmware binaries in the components being updated. Settings and identifiers are expected to be preserved across the update. A device may not support every combination and the driver for such a device must reject any combination which cannot be faithfully implemented.

Firmware Loading

Devices which require firmware to operate usually store it in non-volatile memory on the board, e.g. flash. Some devices store only basic firmware on the board, and the driver loads the rest from disk during probing. `devlink-info` allows users to query firmware information (loaded components and versions).

In other cases the device can both store the image on the board, load from disk, or automatically flash a new image from disk. The `fw_load_policy` devlink parameter can be used to control this behavior (:ref:`Documentation/networking/devlink/devlink-params.rst <devlink_params_generic>`).

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\[linux-master][Documentation][networking][devlink]devlink-flash.rst`, line 56); *backlink***
>
> Unknown interpreted text role "ref".

On-disk firmware files are usually stored in `/lib/firmware/`.

Firmware Version Management

Drivers are expected to implement `devlink-flash` and `devlink-info` functionality, which together allow for implementing vendor-independent automated firmware update facilities.

`devlink-info` exposes the `driver` name and three version groups (`fixed`, `running`, `stored`).

The `driver` attribute and `fixed` group identify the specific device design, e.g. for looking up applicable firmware updates. This is why `serial_number` is not part of the `fixed` versions (even though it is fixed) - `fixed` versions should identify the design, not a single device.

`running` and `stored` firmware versions identify the firmware running on the device, and firmware which will be activated after reboot or device reset.

The firmware update agent is supposed to be able to follow this simple algorithm to update firmware contents, regardless of the device vendor:

```
# Get unique HW design identifier
$hw_id = devlink-dev-info['fixed']

# Find out which FW flash we want to use for this NIC
$want_flash_vers = some-db-backed.lookup($hw_id, 'flash')
```

```
# Update flash if necessary
if $want_flash_vers != devlink-dev-info['stored']:
    $file = some-db-backed.download($hw_id, 'flash')
    devlink-dev-flash($file)

# Find out the expected overall firmware versions
$want_fw_vers = some-db-backed.lookup($hw_id, 'all')

# Update on-disk file if necessary
if $want_fw_vers != devlink-dev-info['running']:
    $file = some-db-backed.download($hw_id, 'disk')
    write($file, '/lib/firmware/')

# Try device reset, if available
if $want_fw_vers != devlink-dev-info['running']:
   devlink-reset()

# Reboot, if reset wasn't enough
if $want_fw_vers != devlink-dev-info['running']:
   reboot()
```

Note that each reference to `devlink-dev-info` in this pseudo-code is expected to fetch up-to-date information from the kernel.

For the convenience of identifying firmware files some vendors add `bundle_id` information to the firmware versions. This meta-version covers multiple per-component versions and can be used e.g. in firmware file names (all component versions could get rather long.)