Devlink Params

`devlink` provides capability for a driver to expose device parameters for low level device functionality. Since devlink can operate at the device-wide level, it can be used to provide configuration that may affect multiple ports on a single device.

This document describes a number of generic parameters that are supported across multiple drivers. Each driver is also free to add their own parameters. Each driver must document the specific parameters they support, whether generic or not.

Configuration modes

Parameters may be set in different configuration modes.

Possible configuration modes

Name	Description
`runtime`	set while the driver is running, and takes effect immediately. No reset is required.
`driverinit`	applied while the driver initializes. Requires the user to restart the driver using the `devlink` reload command.
`permanent`	written to the device's non-volatile memory. A hard reset is required for it to take effect.

Reloading

In order for `driverinit` parameters to take effect, the driver must support reloading via the `devlink-reload` command. This command will request a reload of the device driver.

Generic configuration parameters

The following is a list of generic configuration parameters that drivers may add. Use of generic parameters is preferred over each driver creating their own name.

List of generic parameters

Name	Type	Description
`enable_sriov`	Boolean	Enable Single Root I/O Virtualization (SRIOV) in the device.
`ignore_ari`	Boolean	Ignore Alternative Routing-ID Interpretation (ARI) capability. If enabled, the adapter will ignore ARI capability even when the platform has support enabled. The device will create the same number of partitions as when the platform does not support ARI.
`msix_vec_per_pf_max`	u32	Provides the maximum number of MSI-X interrupts that a device can create. Value is the same across all physical functions (PFs) in the device.
`msix_vec_per_pf_min`	u32	Provides the minimum number of MSI-X interrupts required for the device to initialize. Value is the same across all physical functions (PFs) in the device.
`fw_load_policy`	u8	Control the device's firmware loading policy. • `DEVLINK_PARAM_FW_LOAD_POLICY_VALUE_DRIVER` (0) Load firmware version preferred by the driver. • `DEVLINK_PARAM_FW_LOAD_POLICY_VALUE_FLASH` (1) Load firmware currently stored in flash. • `DEVLINK_PARAM_FW_LOAD_POLICY_VALUE_DISK` (2) Load firmware currently available on host's disk.
`reset_dev_on_drv_probe`	u8	Controls the device's reset policy on driver probe. • `DEVLINK_PARAM_RESET_DEV_ON_DRV_PROBE_VALUE_UNKNOWN` (0) Unknown or invalid value. • `DEVLINK_PARAM_RESET_DEV_ON_DRV_PROBE_VALUE_ALWAYS` (1) Always reset device on driver probe. • `DEVLINK_PARAM_RESET_DEV_ON_DRV_PROBE_VALUE_NEVER` (2) Never reset device on driver probe. • `DEVLINK_PARAM_RESET_DEV_ON_DRV_PROBE_VALUE_DISK` (3) Reset the device only if firmware can be found in the filesystem.
`enable_roce`	Boolean	Enable handling of RoCE traffic in the device.
`enable_eth`	Boolean	When enabled, the device driver will instantiate Ethernet specific auxiliary device of the devlink device.
`enable_rdma`	Boolean	When enabled, the device driver will instantiate RDMA specific auxiliary device of the devlink device.
`enable_vnet`	Boolean	When enabled, the device driver will instantiate VDPA networking specific auxiliary device of the devlink device.
`enable_iwarp`	Boolean	Enable handling of iWARP traffic in the device.

`internal_err_reset`	Boolean	When enabled, the device driver will reset the device on internal errors.
`max_macs`	u32	Typically macvlan, vlan net devices mac are also programmed in their parent netdevice's Function rx filter. This parameter limit the maximum number of unicast mac address filters to receive traffic from per ethernet port of this device.
`region_snapshot_enable`	Boolean	Enable capture of `devlink-region` snapshots.
`enable_remote_dev_reset`	Boolean	Enable device reset by remote host. When cleared, the device driver will NACK any attempt of other host to reset the device. This parameter is useful for setups where a device is shared by different hosts, such as multi-host setup.
`io_eq_size`	u32	Control the size of I/O completion EQs.
`event_eq_size`	u32	Control the size of asynchronous control events EQ.