MHI Topology

This document provides information about the MHI topology modeling and representation in the kernel.

MHI Controller

MHI controller driver manages the interaction with the MHI client devices such as the external modems and WiFi chipsets. It is also the MHI bus master which is in charge of managing the physical link between the host and device. It is however not involved in the actual data transfer as the data transfer is taken care by the physical bus such as PCIe. Each controller driver exposes channels and events based on the client device type.

Below are the roles of the MHI controller driver:

- Turns on the physical bus and establishes the link to the device
- Configures IRQs, IOMMU, and IOMEM
- Allocates struct mhi_controller and registers with the MHI bus framework with channel and event configurations using mhi_register_controller.
- Initiates power on and shutdown sequence
- Initiates suspend and resume power management operations of the device.

MHI Device

MHI device is the logical device which binds to a maximum of two MHI channels for bi-directional communication. Once MHI is in powered on state, the MHI core will create MHI devices based on the channel configuration exposed by the controller. There can be a single MHI device for each channel or for a couple of channels.

Each supported device is enumerated in:

```
/sys/bus/mhi/devices/
```

MHI Driver

MHI driver is the client driver which binds to one or more MHI devices. The MHI driver sends and receives the upper-layer protocol packets like IP packets, modem control messages, and diagnostics messages over MHI. The MHI core will bind the MHI devices to the MHI driver.

Each supported driver is enumerated in:

```
/sys/bus/mhi/drivers/
```

Below are the roles of the MHI driver:

- Registers the driver with the MHI bus framework using mhi_driver_register.
- Prepares the device for transfer by calling mhi_prepare_for_transfer.
- Initiates data transfer by calling mhi_queue_transfer.
- Once the data transfer is finished, calls mhi_unprepare_from_transfer to end data transfer.