EVE VLAN support - create VLAN and bond interfaces

Background

EVE supports access VLANs for application instances connected to switch network instances. Virtual interfaces (vifs) corresponding to different applications (attached to the same switch network instance) can be made part of different VLANs with this feature. Access port VLAN support is specified in EVE VLAN support - switch network instances

This document describes the more complex part which includes using VLANs (and/or Link Aggregation Groups (LAGs), aka “bonds”) for EVE management traffic and for local network interfaces.

Use cases

We want to structure things so that both EVE management traffic and the app instances can access VLANs, LAGs, and VLANs over LAGs. The VLANs and LAGs part of the picture are new, but the other parts are existing. (We might have confusion and bugs due to the common practise of the name of the system adapter, logical label, phylabel, and Linux ifname being identical in most deployments and models.)

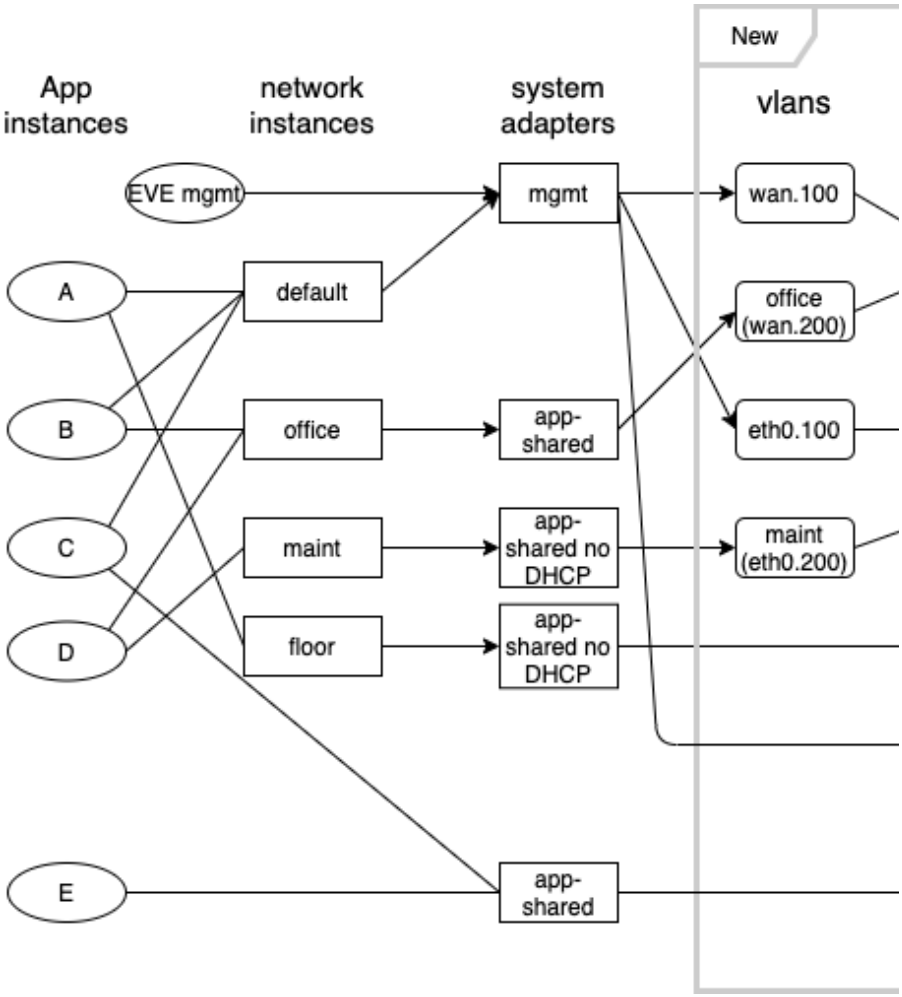
Today, when management and application traffic share the same physical port (eth/wwan/wlan), they both share the same network and cannot be isolated for preferential/special treatment. Use of vlans solves these problems by attaching the EVE vlan interfaces to separate networks, which helps in isolating the management traffic from application traffic. This allows the external networks to give preferential treatments as per their requirements. EVE can extend the isolation by for example configuring ACL rules that block traffic coming on management specific vlan interfaces to not get forwarded and vice versa. EVE can also implement special traffic metering logic on management interfaces to police certain types of traffic more than others without affecting the application traffic.

NOTE: There are many types of bond interfaces that linux bonding modules support. There are certain types of bond interfaces that will need the upstream network equipment (switch or router) to be configured with corresponding configuration eg: 802.3ad with Link Aggregation Control Protocol (LACP).
And there are other types of bond interfaces that will not need any upstream network equipment configuration eg: active-backup mode, adaptive load balancing mode, etc.
At this point it is not clear what type of bonding interfaces customers might ask for in the future. Within the scope of this document we define placeholder API definitions for bond interfaces and represent how bonds will look in the bigger picture, but not discuss any implementation specific details.

Object Model

Today’s object model has two layers, and we will discuss a new intermediate layer below:

1. A system adapter (tied to network object configuration) which specifies IP/DHCP configuration and where we also place information about network proxies which EVE should know.



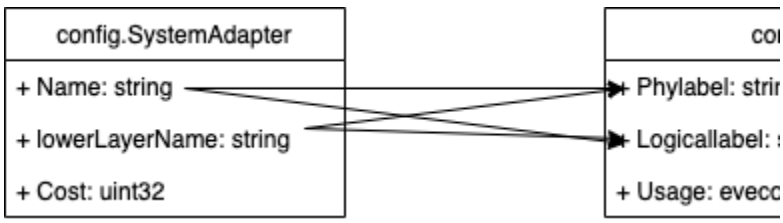
3. A physical I/O description, which has the physical ports/receptacles and associated information like their I/O addresses, the physical label (normally printed on the enclosure), a user-chosen logical label (such as “shopfloor” or “WAN”), and constraints on app-direct assignment (the assignment group).

The current relationship between the system adapter and physical I/O is based on the names which are used (and not things like UUIDs). Due to our gradual introduction of the PhysicalIO layer the implementations might look at several “name” and “label” fields but fundamentally we have:

- The System Adapter has a name for itself, and it also has a name referencing the layer below. In the EVE API that field is called lowerLayerName
- The PhysicalIO has several names for itself; a phylabel and a logicallabel in the EVE API. It seems like the EVE API has comments which are incorrect on this but, the intent is that the lowerLayerName in the System Adapter refers to the logicallabel in the PhysicalIO. (The physicalIO also has fields like ifname and PCI addresses, which serve the purpose of associating the object with the devices in the Linux kernel and hardware. They do not per se “name” the object.)
- The System Adapters need unique names (across all system adapters) and the PhysicalIO need unique names (across all of the PhysicalIO objects), but we commonly use the same name (like “eth0”) in the two objects.

As we introduce the L2 layer object we want it to be sufficiently general that we can have a SystemAdapter refer to a VLAN which refers to a LAG which refers to two or more PhysicalIO. We also want to support the old model where system adapters refer directly to PhysicalIO adapters. This recursion implies that the names need to be unique across vlans, bonds and PhysicalIO logical labels. The picture below shows how system adapters relate to PhysicalIO adapters currently.

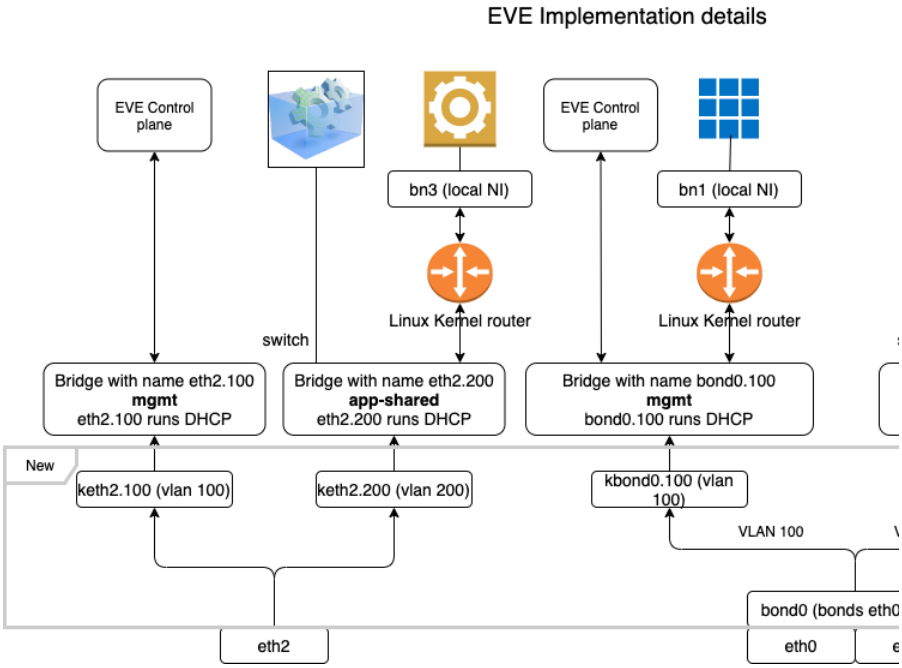
The current implementation of the reference relations between system adapters and physical IO is confusing. EVE code does not have a uniform way of looking up for referenced objects. It is sometimes the Name field that is used to lookup lower layer adapters and some other times it is the lowerLayerName field that gets used to lookup lower layer adapters. We should streamline the way we maintain references between different layers of adapters. Controllers should always have the upper layers reference lower layers making the lowerLayerName field of upper layers reference to the logicalLabel field of lower layers.



When we specify a Network Instance we have a port field (which can be a group like “uplink” which means a management port, or a specific interface). That reference can now be to a logical label of a PhysicalIO or to the name/label of a L2 object (vlans/bonds).

EVE Implementation reference

The picture below shows the new vlan and bond constructs that will be implemented in EVE OS. Without vlans or bond interfaces, physical IO interfaces are directly enslaved to shadow bridge interfaces (bridge created with name of physical IO and the name of physical IO itself changed to avoid conflicts). In a similar fashion with vlans/bonds it will be the vlan/bond adapter that gets enslaved to the shadow bridge. With vlans and bonds, the shadow bridge gets the name of the vlan/bond adapter and the name of the vlan/bond adapter itself gets mangled to avoid conflicts.



Device model changes

Current device model has an ioMemberList that describes the various physical IO interfaces present on the device. Two additional lists - vlanAdapters list, bondAdapters list will be included in the device model to describe the vlan adapters and LAGs respectively. Though there are different types of IO adapters like USB, ethernet audio etc, the sample device model below only shows IO adapters of type ethernet for brevity.

The model file below contains the newly added vlanAdapters list and bondAdapters list along with the existing ioMemberList. Each vlan adapter in the vlanAdapters list has a name (typically reflects the lower layer adapter name and vlan id), lowerLayerName (name of the underlying physical adapter) and vlan id. Similarly each of the bondAdapters list entries has a name field, lowerLayerAdapters list (list of underlying member physical IO adapters) and bond parameters. The example model file below shows three Vlan adapters, the first two of them has a physical IO adapter as parent and the third with a bond interface as it’s parent adapter. The bond interface in the example below splices two physical IO adapters (eth2, eth3) into a bond.

The Vlan and bond adapters specified as part of the device model file are fixed and cannot be deleted from UI. Changes can be made to adapter configuration from UI such as the cost, network attachments, usage type, etc.

Example device model

```
{
   "arch":2,
   "attr":{
      "memory":"4G",
      "storage":"30G",
      "Cpus":"4",
      "hsm":"2",
      "leds":"0",
```

```
   "watchdog":"true"
},
"vlanAdapters":[
   {
      "name":"eth1.100",
      "lowerLayerName":"eth1",
      "vlan_id":100
   },
   {
      "name":"shopfloor",
      "lowerLayerName":"eth1",
      "vlan_id":200
   },
   {
      "name":"bond0.100",
      "lowerLayerName":"bond0",
      "vlan_id":100
   }
],
"bondAdapters":[
   {
      "name":"bond0",
      "lowerLayerAdapters":[
         "eth2",
         "eth3"
      ],
      "bondParams":{

      }
   }
],
"ioMemberList":[
   {
      "ztype":1,
      "phylabel":"eth0",
      "usage":1,
      "assigngrp":"eth0",
      "phyaddrs":{
         "Ifname":"eth0",
         "PciLong":"0000:02:00.0"
      },
      "logicallabel":"eth0",
      "cost":0,
      "usagePolicy":{

      }
   },
   {
      "ztype":1,
      "phylabel":"eth1",
      "assigngrp":"eth1",
      "phyaddrs":{
         "Ifname":"eth1",
         "PciLong":"0000:03:00.0"
      },
      "logicallabel":"eth1",
      "cost":0,
      "usagePolicy":{

      }
   },
   {
      "ztype":1,
      "phylabel":"eth2",
      "assigngrp":"eth2",
      "phyaddrs":{
         "Ifname":"eth2",
         "PciLong":"0000:04:00.0"
      },
      "logicallabel":"eth2",
      "cost":0,
```

```
         "usagePolicy":{

         }
      },
      {
         "ztype":1,
         "phylabel":"eth3",
         "assigngrp":"eth3",
         "phyaddrs":{
            "Ifname":"eth3",
            "PciLong":"0000:05:00.0"
         },
         "logicallabel":"eth3",
         "cost":0,
         "usagePolicy":{

         }
      }
   ]
}
```

Example realization in the EVE API

New vlan adapter proto message:

```
message VlanAdapter {
        string name = 1;             // name of this VLAN adapter
        string lower_layer_name = 2; // name of the lower layer adapter bond/physicalIO
        uint16 vlan_id = 3;          // VLAN ID
}
```

New bond adapter proto message:

```
message BondAdapter {
        string bond_name = 1;                                // name of this bond adapter
        repeated string lower_layer_adapters = 2; // list of PhysicalIO adapter logical names
        // Add bond parameters below.
}
```

Additions to existing EdgeDevConfig message:

```
message EdgeDevConfig {
+         repeated VlanAdapter vlans = 30;
+         repeated BondAdapter bonds = 31;
}
```

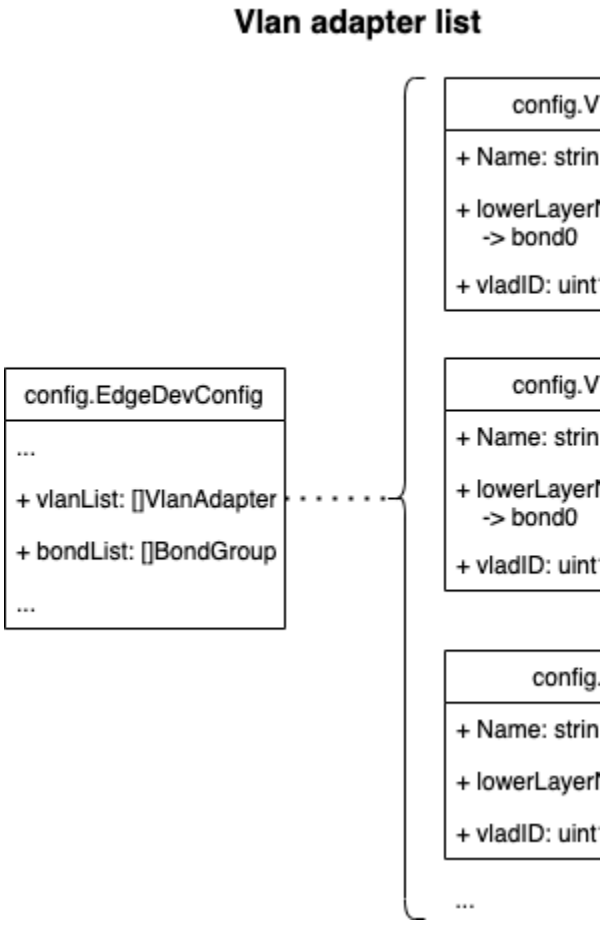
VlanAdapter and BondAdapter messages together comprise the System Adapter L2 layer sandwiched between the SystemAdapter layer and the PhysicalIO layer.

Edge node configuration (config.EdgeDevConfig) will have two additional lists (vlans, bonds) in addition to the existing SystemAdapterList and DeviceIoList. The cloud controller shall have an indication flag in config.PhysicalIO that tells edges nodes and user agents that a particular physical IO adapter is either a parent of vlan adapters or is part of another bond adapter.

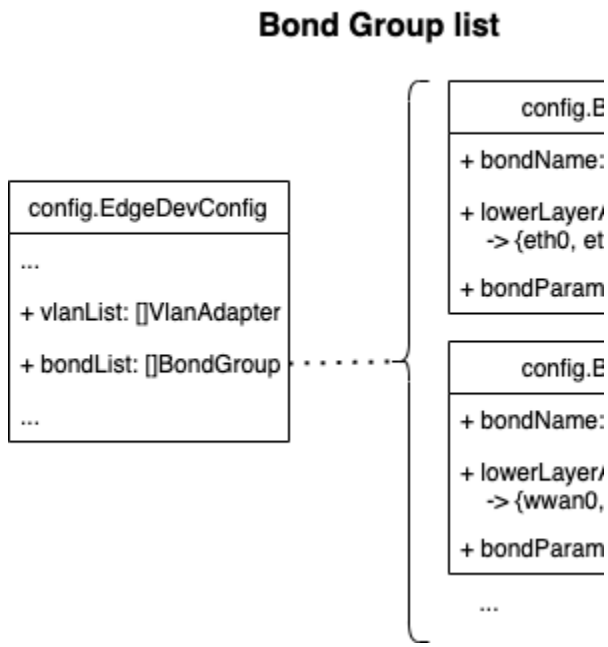
```
enum LogicalLinkType {
  LOGICAL_LINK_TYPE_NONE  = 0;
  LOGICAL_LINK_TYPE_VLANS = 1;
  LOGICAL_LINK_TYPE_BONDS = 2;
}

message PhysicalIO {
+         // aType - indicate if this adapter has been used for Vlans or Bonds
+         LogicalLinkType logicalType = 9;
}
```

A new proto enum type LogicalLinkType is defined above. It shall be used to indicate if a particular adapter is made a parent of other vlan adapters or has been made part of a bond interface.

The pictures below show examples for vlan adapter and bond adapter lists that will be part of EdgeDevConfig.

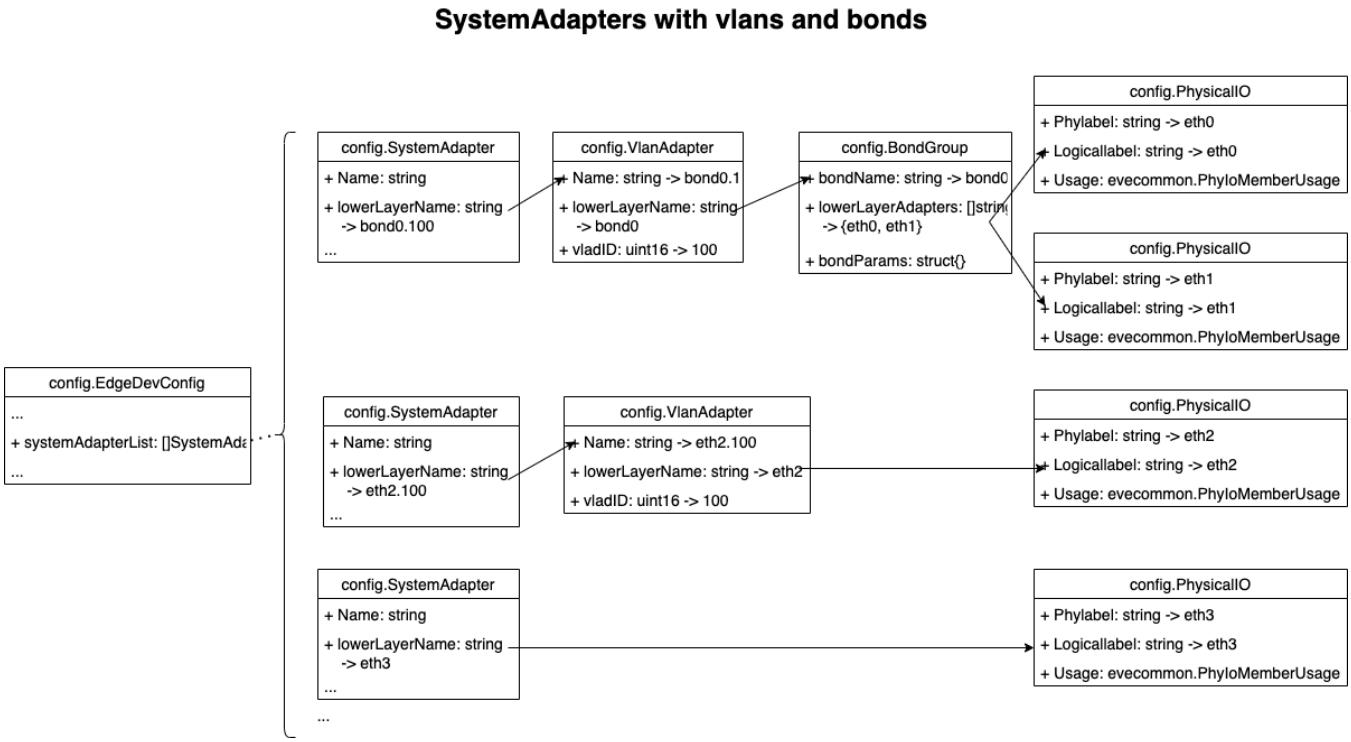


Each vlan adapter object contains a user given name (from device model file), its vlan id, and the parent physical adapter of this vlan adapter. Vlan adapters can have both physical adapter and bond interface types as parent adapter. It might make sense for EVE code to collapse the vlan related fields into the existing NetworkPortConfig (add the new vlan id field) object that is part of the DevicePortConfig object. Bond interfaces may need to be sent as a separate list inside DevicePortConfig. Also NetworkPortConfig will have a new indicator field that indicates if the current network port is a vlan interface or bond interface or a regular interface.



Domain manager service in EVE should skip PhysicalIO adapters that are used for system adapters of type vlan and bond. Physical IO adapters that are made part of vlans and bond interfaces from device model files can never be changed to app-direct interfaces.

NIM service should take care of not creating the kethx interfaces for physical IO adapters that are underlying adapters for other Vland or bond adapters. NIM should also have the additional logic of creating any non-existing vlan interfaces or bond interfaces in linux kernel when the edge node decides to change its DevicePortConfig. There should also be additional logic to remove any pre-created vlan/bond interfaces that are no longer part of the new DevicePortConfig that is being applied. Even though the vlan and bond interfaces being part of the device model cannot be added/deleted from user agents (UI/zcli), the edge node may in the event of prolonged network disconnection try to replace the running network configuration with last resort configuration. During such an event, NIM service should take care of creating new vlans/bonds that are required and remove any stale vlans/bonds that are not required according to the DevicePortConfig file being applied.



Example EdgeDevConfig that cloud controller backend would send to Edge node

```
EdgeDevConfig {
        ...
        ...
        SystemAdapterList: [
                SystemAdapter: {
                        ...
                        Name: "eth0",
                        NetworkUUID: "xyx", // UUID corresponding to net0 above
                        LowerLayerName: "eth0",
                        Cost: 0,
                        ...
                },
                SystemAdapter: {
                        ...
                        Name: "eth1.100",
                        NetworkUUID: "123", // UUID corresponding to net1 above
                        LowerLayerName: "eth1.100",
                        Cost: 0,
                        ...
                },
                SystemAdapter: {
                        ...
                        Name: "shopfloor",
                        NetworkUUID: "321", // UUID corresponding to net2 above
                        LowerLayerName: "shopfloor",
                        Cost: 0,
                        ...
                },
                SystemAdapter: {
                        ...
                        Name: "bond0.100",
                        NetworkUUID: "abc", // UUID corresponding to net3 above
                        LowerLayerName: "bond0.100",
                        Cost: 0,
                        ...
                },
        ],
        VlanList: [
                VlanAdapter: {
                        Name: "eth1.100",
                        LowerLayerName: "eth1",
                        VlanId: 100
                },
                VlanAdapter: {
                        Name: "shopfloor",
                        LowerLayerName: "eth1",
                        VlanId: 200
                },
                VlanAdapter: {
                        Name: "bond0.100",
                        LowerLayerName: "bond0",
                        VlanId: 100
                }
        ],
        BondList: [
                BondName: "bond0",
                LowerLayerAdapters: ["eth2", "eth3"],
                BondParams: {}
        ],
        DeviceIoList: [
                PhysicalIO {
                        ...
                        PhyLabel: "eth0",
                        LogicalName: "eth0",
                        LogicalType: LOGICAL_LINK_TYPE_NONE,
                        ...
                },
                PhysicalIO {
                        ...
                        PhyLabel: "eth1",
```

```
                        LogicalName: "eth1",
                        LogicalType: LOGICAL_LINK_TYPE_VLANS,
                        ...
                },
                PhysicalIO {
                        ...
                        PhyLabel: "eth2",
                        LogicalName: "eth2",
                        LogicalType: LOGICAL_LINK_TYPE_BONDS,
                        ...
                },
                PhysicalIO {
                        ...
                        PhyLabel: "eth3",
                        LogicalName: "eth3",
                        LogicalType: LOGICAL_LINK_TYPE_BONDS,
                        ...
                }
        ]
}
```