OpFlex Configuration

OpFlex Control Protocol or OpFlex is an open source protocol that supports policy exchange between network policy controller and smart devices that are capable of rendering abstract policies.

Restrictions for OpFlex Configuration

- Only two peers per domain are supported on opflex.
- Loopback or physical interface address can be used for opflex agent configuration on the device, but only physical interface ip address can be used for the ACI fabric.
- Avoid configuring VRFs on the device that are getting pushed through opflex.
- If multiple peers exist on the device, and the opflex agent receives the tenant update from multiple peers for the same tenant, the opflex agent picks the latest update from the opflex server.
- VLAN 4 is used only for peering between ASR1000 device and the ACI fabric.

Feature Information for OpFlex Configuration

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for OpFlex Configuration

Feature Name	Releases	Feature Information
OpFlex Configuration	Cisco IOS XE Release 16.5	The OpFlex Control Protocol or OpFlex Configuration is an open source protocol that supports policy exchange between the network policy controller and smart devices that are capable of rendering abstract policies. The following commands were introduced or modified by this feature: **opflex agent**, **service vxlan-evpn**.

Opflex Agent

Conbody

OpFlex is used to communicate policy from a border leaf to a router configured as a datacenter interconnect. In this configuration, the ASR1000 device is considered to have a ACI fabric facing side that peers with a border leaf over OpFlex and a WAN facing side that is configured either manually or through a separate network management system or controller.

With this configuration, we can automate frequently changing per-tenant configurations through OpFlex. The ACI information model does not support a complete end-to-end WAN configuration. For example, parameters that may be exchanged between a border leaf and DCI device include a tenant ID, VRF ID, ASN ID, and DCI-IP.

Note You can configure a DCI device and gather WAN statistics over opflex using abstract policy.

Opflex Agent in Polaris

The role of the Opflex Agent (OFA) is to handle policy updates, and render configuration as required. To handle updates, the agent needs to establish a connection with an Opflex peer. Then, as the updates start from

the peer(s), the agent operates in two phases: extracts data from the received policy and auto-generates the concrete configuration.

Note OFA is simultaneously used for multiple configuration models that control multiple functionalities. To facilitate this, OFA uses the notion of service, which handles a subset of incoming policies and renders concrete configuration appropriately.

For more information on the EVPN configuration, refer to the *EVPN VxLAN L3* section in the Carrier Ethernet Configuration Guide:

Peering and Opflex Control Protocol

Peering and core opflex protocol support is provided by a suite of third-party libraries: `libopflex, libuv, boost, and rapidjson`. The major roles are played by the `libopflex` library, which is responsible for the opflex protocol implementation, and `libuv`, which is responsible for the networking connections.

The opflex protocol uses TCP or IP for communication. Packets are sent and received through the data plane by leveraging the LFTS layer which allows BINOS processes to use TCP or IP sockets.. The peer ip-address and tcp-port are set through IOS configuration commands and used to provision the connection.

For each peer, once a connection is established, policy updates are pushed to the opflex agent from the Controller through the peer. At present time there is no way to send an indication to the peer or controller about the success of handling a policy update. That is, from the controller's view, all policy updates are successful. This is a limitation imposed by `libopflex`.

Policy data extraction

The policies are encoded in JSON format. It is the role of libopflex to parse JSON encoded policies and call an appropriate handler to process the data. This data pertained in the policy is service-specific, that is, it is used to configure a particular networking solution and thus requires service-specific handling.

Each service provides a dedicated BINOS side parser which is used by OFA to extract the policy data, used later for configuration rendering. The invocation of these parser is the responsibility of the Domain MGR.

Services

Different services are implemented as distinct sub-systems and can be packaged according to specific platform requirements. If a service is packaged, it is required to register with the Service MGR in OFA. During registration, the service provides several callbacks which are used by OFA to interact with the service.

The role of a service is to take the extracted policy data and apply the logic contained in it to produce the required concrete auto-generated configuration. The actual configuration rendering is performed in IOS. Similar to policy data extraction, rendering the concrete configuration requires service-specific implementation.

OFA also provides several event notifications to services. The handling of these events are within the service responsibility. The notified events are:

- Removal of a domain
- NVGEN request

- RP becoming active (following a switchover)

Policy Data Delivery to Services

As the data extraction and configuration rendering take place in two different components, with the former being on the BINOS side and the latter in IOS, the policy data needs to be delivered from the service-specific parser to the corresponding OFA service in IOS.

Therefore, while maintaining the generic nature of the OFA infrastructure, the policy data is transported as payload in a generic datagram. That is, the OFA infrastructure is not made aware of the service-specific data, which allows a simple addition of supported services.

Peering

For peering information OFA uses the notion of a domain. Domain is a logical collection of one or two peers which are synchronized. That is, two peers cannot send the same policy update.

Having two peers adds to fault resistance of the particular domain. Below is an example of an Opflex domain configuration.

```
Opflex agent
  Domain dom1
       Identity dci-[10.20.30.40]
       Peer p1 ip-address 156.2.21.10 tcp-port 8009 src-ip-address 148.2.15.1
       Peer p2 ip-address 156.2.21.10 tcp-port 8009 src-ip-address 148.2.15.1
```

Domain: At most two peers per domain.

Identity: The identity is a string that uniquely identifies a domain and needs to be matched with the Opflex peer definition of a domain identity, which is governed by the Opflex library in use.

The third party library (used by OFA), `libopflex`, uses the format `dci-[ip]`, where `ip` is the ip-address of the bgp ip loopback protocol.

Peers. The required information for each peer is the `ip-address` and `tcp-port` of the peer. The `src-ip-address` is the ip-address used by the OFA to send Opflex requests.

OFA Infra

The peering information and services configuration is synced to the standby through the common CONFIG_SYNC mechanism. The active peering connections are not established on the standby, but are re-created following a switchover.

There is no additional state that needs to be tracked.

OFA services

It is the responsibility of the service to support HA. To help facilitate this, OFA provides several workflows to support various HA scenarios.

Incremental sync following a policy update

1. The policy data is delivered to the service on the active
2. The policy data is handled on the active
3. The policy data is augmented by the service with auto-generated values and repackaged
4. OFA sends the augmented policy data to the standby
5. The augmented policy data is delivered to the service

Bulk Sync on Standby

- OFA requests the service for the current state in the form of a list of policy data items (which will recreate the state)
- OFA sends the list to the standby
- Each policy data item on the list is delivered to the service

Domain Removal CLI

- Domain is removed on the active (through CLI)
- Service is notified of a domain removal on the active
- Domain is removed on the standby (through CONFIG_SYNC)
- Service is notified of a domain removal on the standby
- Each service can choose the respective parts of the workflow

Active	Standby	Comments
Peering information	As on the active	Through CONFIG_SYNC
Peering connections	Disabled	Re-established on switchover
Individual services CLI configuration	As on the active	Through CONFIG_SYNC
Service state (e.g. updates history)	As supported by service	
Service generated configuration	As supported by service	

Configuring the Opflex Agent

OFA Configuration has two parts - the generic peering information and service-specific setup. The former creates logical policy update sources (2 peers per source), while the latter provides a platform for a service to have its particular configuration data.

Opflex Agent Services

Services are used to configure particular feature sets and the one service supported by OFA is the Service VXLAN-EVPN, or in short SVE.

Service VXLAN-EVPN (SVE)

Details about the actual service and the business logic it provides can be found in a document describing the VXLAN-EVPN solution. Here we focus on the service functionality from the Opflex perspective, i.e. service configuration and the rendering template.

In a nutshell, SVE receives policy updates which hold tenant information and render the corresponding configuration based on a template. The policy updates are encoded in the JSON format and convey several pieces of information:

- Policy space
- Routing domain
- Local VRF name
- A list of route targets (RTs)

Following updates, SVE builds an internal Tenant DB with the following properties for each tenant:

- Unique name (based on policy space and routing domain)
- Route targets (RTs)
- VRF name

Each tenant contributes its list of RTs to the VRF it references (name received in the policy update). Note that the same VRF can be referenced by multiple tenants, while each tenant references only a single VRF.

Each policy update carries information for a single tenant. The update can be of two types:

- **Tenant PUT** completely replaces current information in the Tenant DB with the new one.
- **Tenant DELETE** removes the tenant from the Tenant DB.

The rendered configuration is an up-to-date projection of the Tenant DB into VRF, BD, BGP, and NVE features. For each referenced VRF the rendering template in Fig. 3 is used

The rendered configuration is an up-to-date projection of the Tenant DB into VRF, BD, BGP, and NVE features. For each referenced VRF, the rendering template is used.

Services Configuration

SVE is configured as a subtree under opflex agent. It has two configurable items:

- nve-id: The interfaceis used for VNI generation
- bdi-ip: It is used for the configuration of the generated BDI
- Both the items are used in the rendering template

Manual versus Rendered Configuration

At times the rendered and manual configuration might have a conflict. The following set of rules outlines the current behavior of SVE in various scenarios where there is a potential collision between an auto-generated and manual config.

- SVE configuration may overwrite existing manual configuration for BDI attributes (ip address/forwarding vrf).
- Manual config is allowed to overwrite any SVE-generated configuration. Note that if done improperly service disruption is possible.
- Additional features, such as, NAT or QoS are manually configured on an SVE-generated BDI.
- SVE never deletes VRF/BD/BDI/NVE definitions (including those which are auto-generated following a tenant update)
- The only cleanup allowed by SVE is the RTs (normal or stitching) configured inside the VRF by SVE. That is, SVE will delete RTs during tenant updates. As a result we might have a rare scenario where a user configured RT is deleted by SVE due to tenant update if the tenant had a similar RT before the update.

```
Opflex agent
    Service vxlan-evpn
        Nve-id 1
        Bdi-ip 9.9.9.9 255.255.255.0
```

Policy Data Delivery to Services

As the data extraction and configuration rendering take place in two different components, with the former being on the BINOS side and the latter in IOS, the policy data needs to be delivered from the service-specific parser to the corresponding OFA service in IOS.

To achieve that, while maintaining the generic nature of the OFA infrastructure, the policy data is transported as payload in a generic datagram. That is, the OFA infrastructure is not made aware of the service-specific data, which allows a simple addition of supported services.

High Availability for Opflex

There are two main aspects for High Availability in SVE: tenant information and features (VRF/BD/BDI/NVE) configurational data. Features sync config data independently from SVE. Whereas SVE utilizes the OFA infra to sync Tenant DB information (see Section 0 on OFA High Availability)

In terms of data consistency: on the active, tenant information fully matches features' config data; on the standby, however, tenant information is not tied to features' config data as both are synced independently (for example a VRF might not yet be available on the standby when tenant information is synced).

After switchover each tenant is validated to have all the matching feature configuration. In case some pieces are missing (which might happen during the feature sync process), these are re-configured.

Handling of OFA events

As mentioned previously, each services is notified of a system-wide event. For the purposes of this user guide, it is of interest to discuss the consequences of domain removal and switchover in SVE.

Peering

When a domain is removed (unconfigured), SVE deletes the tenants which were last modified through that domain. For example, in the case of 4 tenants (as shown in the table below), if domain D1 is removed, this would lead to deletion of tenants T1, and T3.

Tenant deletion does not alter any of the existing configuration, except for route targets, as indicated in Section 0). Also, recall that the generated concrete configuration is a projection of the Tenant DB. Thus RTs will remain part of a VRF as long as there at least one tenant in the DB which still holds the RT in question.

Switchover

As mentioned earlier, the features are responsible for successful syncing of their respective configuration to the standby (VRF, BD, BGP, NVE).

On switchover, however, SVE attempts to re-apply the expected configuration. For each tenant, if successful, this results in no-op, otherwise, the tenant is deleted.

Configuring OpFlex Configuration

```
Device(config)# interface FortyGigabitEthernet1/1/1.4
 description "Connected to Spine"
 encapsulation dot1Q 4
 ip address 88.0.0.4 255.255.255.0
 ip ospf mtu-ignore
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
router ospf 100
 nsf ietf
 area 1 nssa
 area 100 nssa
```

```
 network 1.1.1.1 0.0.0.0 area 0
 network 1.0.0.0 0.0.0.255 area 100
!
vrf definition cust2
rd 1:1054
!
address-family ipv4
 route-target export 100:2
 route-target import 100:2
exit-address-family
!
bridge-domain 10368
 member vni 15000001
!
bridge-domain 10367
 member vni 15000017
!
interface bdi 10368
 vrf forwarding cust1
 ip address 1.1.1.1 255.255.255.0
 no shutdown
!
interface bdi 10367
 vrf forwarding cust2
 ip address 1.1.1.1 255.255.255.0
 no shutdown
!
opflex agent
service vxlan-evpn
  nve-id 1
  bdi-ip 1.1.1.1 255.255.255.0
domain Fabric1
  identity dci-[31.1.1.1]
peer 1 ip-address 88.0.0.3 tcp-port 8009 src-ip-address 88.0.0.4
!
interface nve1
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 vxlan udp port 48879
!
router bgp 101
 bgp router-id 31.1.1.1
 bgp log-neighbor-changes
 bgp listen limit 5000
 bgp graceful-restart
 timers bgp 120 360
 neighbor 102.102.102.102 remote-as 100
 neighbor 102.102.102.102 ebgp-multihop 255
 neighbor 102.102.102.102 update-source Loopback0
 neighbor 102.102.102.102 ha-mode graceful-restart
!
 address-family ipv4
  neighbor 102.102.102.102 activate
 exit-address-family
 !
 address-family l2vpn evpn
  import vpnv4 unicast re-originate
  neighbor 102.102.102.102 activate
  neighbor 102.102.102.102 send-community both
 exit-address-family
!
```

Example: Opflex Configuration

Example: OpFlex Configuration

OpFlex Configuration

```
Device# show opflex agent service vxlan-evpn all
Number of tenants: 2
Domain Name Tenant Name VRF
BD VNI
DC1 acivrf_cust1 cust1
10368 15000001
DC1 evpn2_cust2 cust2
10367 15000017
DeviceI# show opflex agent service vxlan-evpn tenant evpn2_cust2
Tenant: evpn2_cust2 (domain DC1)
VRF: cust2 (id 5, rd_nn 1054)
BDI: 10367 VNI: 15000017
Route Targets:
 RT:100:2 ipv4 import
 RT:100:2 ipv4 export
Device# show opflex agent service vxlan-evpn concrete-config all
Number of configs: 2
vrf definition cust1
 rd 1:1053
 address-family ipv4 unicast
 route-target import RT:100:1 stitching
 route-target import RT:100:1
 route-target export RT:100:1 stitching
 route-target export RT:100:1
 address-family ipv6 unicast
 route-target import RT:100:1 stitching
 route-target import RT:100:1
 route-target export RT:100:1 stitching
 route-target export RT:100:1
vrf definition cust2
 rd 1:1054
 address-family ipv4 unicast
 route-target export RT:100:2 stitching
 route-target export RT:100:2
 route-target import RT:100:2 stitching
 route-target import RT:100:2
 address-family ipv6 unicast
 route-target export RT:100:2 stitching
 route-target export RT:100:2
 route-target import RT:100:2 stitching
 route-target import RT:100:2
bridge-domain 10368
 member vni 15000001
interface bdi 10368
 vrf forwarding cust1
 ip address 100.1.1.1 255.255.255.0
 no shutdown
bridge-domain 10367
 member vni 15000017
interface bdi 10367
 vrf forwarding cust2
 ip address 100.1.1.1 255.255.255.0
no shutdown
router bgp 101
 address-family ipv4 vrf cust1
  advertise l2vpn evpn
 address-family ipv4 vrf cust2
  advertise l2vpn evpn

interface nve 1
```

```
 member vni 15000001 vrf cust1
 member vni 15000017 vrf cust2
Device# show ip route vrf cust2
Routing Table: cust2
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static
route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 a - application route
 + - replicated route, % - next hop override, p - overrides from PfR
Gateway of last resort is not set
 5.0.0.0/32 is subnetted, 1 subnets
S 5.1.2.2 is directly connected, Null0
 15.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
B 15.0.0.1/32 [20/0] via 30.1.2.2, 00:35:56
B 15.1.1.0/24 [20/0] via 30.1.2.2, 00:35:56
 30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 30.1.2.0/24 is directly connected, TenGigabitEthernet2/1/0.2
L 30.1.2.1/32 is directly connected, TenGigabitEthernet2/1/0.2
 50.0.0.0/24 is subnetted, 51 subnets
B 50.1.2.0 [20/0] via 10.0.0.34, 3d07h, BDI10367
B 50.2.1.0 [20/0] via 10.0.0.34, 3d07h, BDI10367
60.0.0.0/24 is subnetted, 1 subnets
B 60.1.2.0 [20/0] via 30.1.2.2, 00:35:56
 64.0.0.0/24 is subnetted, 1 subnets
B 64.1.2.0 [20/0] via 30.1.2.2, 00:35:56
 65.0.0.0/24 is subnetted, 1 subnets
B 65.1.2.0 [20/0] via 30.1.2.2, 00:35:56
 100.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 100.1.1.0/24 is directly connected, BDI10367
L 100.1.1.1/32 is directly connected, BDI10367
```

Verifying How to Display or Cleanup Tenants

Verifying OpFlex Configuration

The **show** or **clear** commands are used to display to clean-up tenants.

```
Device# show opflex agent service vxlan-evpn all
Prints brief tenant information for the tenants in the Tenant DB. Information includes:
    Domain name - the most recent source for tenant configuration
    Tenant name - the name of the tenant which is a combination of Opflex Policy Space and
 Routing Domain
    VRF - the tenant VRF
    BD/VNI - the bridge domain/VNI information associated with the tenant
show opflex agent service vxlan-evpn tenant tenant-name
    Prints detailed a specific tenant which includes route targets.
    Device# show opflex agent service vxlan-evpn concrete-config all
    Prints the concrete configuration associated with the tenant

Device# clear opflex agent service vxlan-evpn all
Deletes all tenants currently in the Tenant DB. Note that the tenants do not get automatically
 re-configured. To re-create the tenants, the user has to flap recreate the domain(s). Also,
 the deletion only takes place only on the RP where it is executed (e.g. deleting tenants
on the active does not delete the tenants on the standby).
Debug commands
debug opflex agent service vxlan-evpn [all|debug|error|info]
Enable different levels of debugging for SVE.
```

Additional References for OpFlex Configuration

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards and RFCs

Standard/RFC	Title
Standard	*Title*

MIBs

MIB	MIBs Link
• CISCO-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html